Sara Migliorini

Supporting Distributed Geo-Processing: A Framework for Managing Multi-Accuracy Spatial Data

Ph.D. Thesis

June 5, 2012

Università degli Studi di Verona
Dipartimento di Informatica

Advisor:
prof. Alberto Belussi

Series N°: **TD-08-12**

Università di Verona
Dipartimento di Informatica
Strada le Grazie 15, 37134 Verona
Italy

to Mauro

Abstract. In the last years many countries have developed a Spatial Data Infrastructure (SDI) to manage their geographical information. Large SDIs require new effective techniques to continuously integrate spatial data coming from different sources and characterized by different quality levels. This need is recognized in the scientific literature and is known as data integration or information fusion problem. A specific aspect of spatial data integration concerns the matching and alignment of object geometries. Existing methods mainly perform the integration by simply aligning the less accurate database with the more accurate one, assuming that the latter always contains a better representation of the relevant geometries. Following this approach, spatial entities are merged together in a sub-optimal manner, causing distortions that potentially reduce the overall database quality. This thesis deals with the problem of spatial data integration in a highly-coupled SDI where members have already adhered to a common global schema, hence it focuses on the geometric integration problem assuming that some schema matching operations have already been performed. In particular, the thesis initially proposes a model for representing spatial data together with their quality characteristics, producing a multi-accuracy spatial database, then it defines a novel integration process that takes care of the different positional accuracies of the involved source databases. The main goal of such process is to preserve coherence and consistency of the integrated data and when possible enhancing its accuracy. The proposed multi-accuracy spatial data model and the related integration technique represent the basis for a framework able to support distributed geo-processing in a SDI context. The problem of implementing such long-running distributed computations is also treated from a practical perspective by evaluating the applicability of existing workflow technologies. This evaluation leads to the definition of an ideal software solution, whose characteristics are discussed in the last chapters by considering the design of the proposed integration process as a motivating example.

Acknowledgement

First of all, I would like to thank my advisor, prof. Alberto Belussi, for his support, that it is keeping on also after the end of the PhD program. A great thanks also goes to prof. Carlo Combi for his help and involvement in my PhD course.

I am also grateful to dr Marcello La Rosa and prof. Arthur ter Hofstede who hosted me and collaborated with me during and after my visit at Queensland University of Technology, Brisbane, Australia.

I would also like to thank my PhD thesis referees prof. Manfred Reichert, prof. Peter van Oosterom, and prof. Eliseo Clementini for their precious advices and comments on my studies.

The greatest thank goes to Mauro, who shared with me the joys and difficulties of these years. Many parts of this thesis should not be here without his collaboration. The last thanks goes to my parents and sisters for their support.

Contents

1 Introduction 1
- 1.1 Context Overview 1
  - 1.1.1 Spatial Data Integration 2
  - 1.1.2 Distributed Geo-Processing 4
- 1.2 Motivating Example 5
- 1.3 Contribution 10
- 1.4 Thesis Structure 12

Part I Geo-Processing in Theory

2 Background: Spatial Data Management 17
- 2.1 Spatial Data Representation 17
- 2.2 Spatial Relations 20
  - 2.2.1 Topological Relations 21
  - 2.2.2 Cardinal Directional Relations 22
  - 2.2.3 Distance Relations 24
- 2.3 Spatial Data Integration 25
  - 2.3.1 Ontology-Based Integration 27
  - 2.3.2 Schema Integration 28
  - 2.3.3 Feature or Point Matching 28
- 2.4 Uncertainty in Spatial Data 29
- 2.5 Representation of Uncertain Spatial Data 33
- 2.6 Representation of Uncertain Topological Relations 34
- 2.7 Integration of Uncertain Spatial Data 35
- 2.8 Summary and Concluding Remarks 36

3 A Multi-Accuracy Spatial Data Model 39
- 3.1 Representing Metric Observations 39
- 3.2 Representing Logical Observations 46
- 3.3 MACS Database Accuracy Estimators 55
- 3.4 MACS Implementation Model 56
- 3.5 Summary and Concluding Remarks 57

4 Integration of two MACS databases 59
4.1 Possible Integration Scenarios 60
4.2 Integrating Metric Observations 62
4.3 Integrating Logical Observations 69
4.4 Integrating Metric and Logical Observations Together 72
4.5 Properties of the Integration Process 84
4.6 Distributed Integration of two MACS databases 88
4.6.1 Distributed Integration of Metric Information 89
4.6.2 Distributed Integration of Logical Information 90
4.7 Efficient Covariance Matrix Representation 90
4.8 Integration Applied to a Real-World Case 96
4.8.1 Role of Accuracy During Metric Integration 97
4.8.2 Preservation of Relative Distances 99
4.8.3 Preservation of Topological Relations 100
4.9 Summary and Concluding Remarks 102

Part II Geo-Processing in Practice

5 Background: Workflow Management Systems 109
5.1 Business Process Management 110
5.2 Service Oriented Architecture (SOA) 111
5.3 Workflow Management Systems 112
5.4 Control-Flow Oriented Modeling Languages 114
5.4.1 Web-Services Business Process Execution Language (BPEL) 115
5.4.2 Business Process Modeling Notation (BPMN) 116
5.4.3 The YAWL System 118
5.5 Data-Flow Oriented Modeling Languages 121
5.5.1 The Kepler System 122
5.6 Geo-Processing Workflow Systems 125
5.6.1 OGC Standards 126
5.6.2 Humboldt Project 127
5.6.3 ArcGIS Workflow Manager 129
5.7 Summary and Concluding Remarks 132

6 A Comparison of the Existing WfMSs 135
6.1 Workflow Pattern Evaluation of the Selected WfMSs 135
6.1.1 Workflow Control-Flow Patterns Evaluation 136
6.1.2 Workflow Data Patterns Evalutation 145
6.1.3 Workflow Resource Patterns 149
6.2 Requirements of a Geo-Processing WfMS 152
6.2.1 Integration Process Implementation with BPMN 152
6.2.2 Integration Process Implementation with YAWL 155
6.2.3 Integration Process Implementation with Kepler 156
6.3 The Ideal Solution 157
6.4 Summary and Concluding Remarks 159

7 The NestFlow Solution 161
7.1 Control-Flow Aspects 162
7.2 Data-Flow Aspects 163
7.3 Well-Formed Models 165
7.4 Run-time Behavior 166
7.5 Design of the Integration Process 167
7.5.1 The Main Integration Process 168
7.5.2 The ObjectInt Process 171
7.5.3 The LogRelInt Process 172
7.5.4 The SolveInconsistency Process 174
7.5.5 The DistributedIntegration Process 174
7.6 Summary and Concluding Remarks 179

8 Conclusions 181

References 185

Part III Appendices

Statistical Background 197
A.1 Probability Theory 197
A.2 Fuzzy Theory 199

Workflow Patterns 201
B.1 Control-Flow Patterns 201
B.2 Data Patterns 207
B.3 Resource Patterns 209

1

Introduction

In recent years the amount of available spatial data has grown considerably: technological advances in acquisition methods and tools have made it possible to collect an unprecedented amount of high-resolution and high-quality spatial data, while the proliferation of Internet-based technologies has allowed different organizations to share these data for paying off acquisition and maintenance costs. For instance, modern satellites produce about one terabyte of data per day that can be used by many organizations in addition to the collecting one [56].

A set of organizations interested in maintaining and processing a certain portion of spatial data can build a so virtual organization. The coordination of these new entities is possible thanks to the development of a common spatial data infrastructure, that is becoming a solid reality in many countries. A *Spatial Data Infrastructure* (SDI) is a technological infrastructure through which several organizations with overlapping goals can share data, resources, tools, and competencies in an effective way. Due to its nature, an SDI is usually government-related and regulated by precise rules. In Europe, the development of a global SDI is driven by the INSPIRE project [2] that has been translated into a European directive. These unavoidable changes pose many additional problems regarding the integration, storage and processing of spatial data, as well as regarding the development of supporting software systems. This thesis faces two challenges that usually affect an SDI: (i) the integration of spatial data coming from different sources and characterized by different quality levels, in particular for what regards positional accuracy, (ii) the development of a framework for supporting the construction and maintenance of a distributed and integrated global SDI database.

1.1 Context Overview

This section introduces the context in which the thesis has been developed. In particular, Sec. 1.1.1 describes the concepts behind the integration of spatial data characterized by different quality levels, while Sec. 1.1.2 highlights the technological issues related to the development of an infrastructure for supporting a distributed integration framework.

1.1.1 Spatial Data Integration

As geographical agencies start to collaborate for sharing their acquired data and developed competencies, the integration of spatial data, coming from different sources and acquired using different technologies and instruments, has become a primary need. On the one hand, the integration problem is related to the different data models (schemas) and formats used for representing and storing information: each agency in an SDI can adopt those data structures fitting best to the purposes for which the information was originally collected. On the other hand, the integration activity should consider the different accuracies induced by the specific instruments and technologies used to acquire data, in particular as regards to metric information. Several research efforts, documented in the scientific literature, deal with the first aspect of the integration problem, some of them will be summarized in Sec. 2.3. On the contrary, this thesis concentrates on the second aspect of the integration problem. It considers the case of a highly coupled SDI where members have adhered to a common global schema, and assumes that a previous merging operation on the source schemas has already been performed.

The choice to concentrate on the second aspect of the integration problem has been inspired by a real-world project regarding the creation of the SDI in the Italian Lombardy region. Such SDI represents one of the first cases of spatial data integration between a central reference agency and several municipalities (or aggregation of them) in Italy. Notice that Lombardy has about 1,500 municipalities and the project has been started by experimenting the cooperation among the central regional administration and some selected municipalities. During this project, traditional rubber-sheeting techniques have been adopted for aligning source geometries. The problems encountered during this activity have motivated the search for more sophisticated integration techniques that are presented in this thesis.

Spatial data is characterized by an inherent uncertainty because the measurements needed to survey the shape, extension and position of an object with maximum accuracy are often too expensive, or maximum accuracy is not necessary to satisfy application requirements. Hence, a certain amount of errors in the representation of a spatial object always exists. In literature, the term *accuracy* is defined as a measure of how closely the reported values represent the true values, and *uncertainty* is a statistical estimate of the accuracy of a value modeled with probability theory [58, 66, 114, 129].

The accuracy concept considered here is related to the position of spatial objects, not of their attributes in general. Regarding to positional accuracy, we can observe that the importance of uncertainty is perceived in different ways by the different communities that work in the geographical field. In particular, *computer scientists* working with Geographical Information Systems (GISs) tend to perceive the absolute geometric coordinates as the primary data concerning object locations and to treat them as deterministic values. The measurements from which these coordinates were obtained are seen as unnecessary data once absolute point locations have been determined and no record is kept about them. In this perspective, each relative geometry measure (e.g. distance, angle, etc) and all the other information (e.g. spatial relationships between objects) can be derived from absolute coordinates. On the contrary, *surveyors* typically perceive the measurements concerning geographical objects and relative object distances as being the primary data, while

the computed coordinates are treated as derived variables. The coordinate values are simply seen as a view of the data: the one that best fits the measurements at that time. Considering that in practice the accuracy of relative geometry is higher than the accuracy of absolute positions, it can be concluded that absolute coordinates and relative measures are not equivalent as often believed. The concepts of uncertainty and accuracy of spatial data will be further discussed in Sec. 2.4.

In literature, some authors proposed the introduction of measurements-based cadastral systems [22, 50, 67, 92]. Although it is possible to store measurements rather than derived coordinates, into a database and compute the coordinates as required using all the stored measurement information, this operation is computationally intensive and thus not practical in many applications. As a consequence, in spatial databases, only derived coordinates are usually stored without any information about their accuracy or the original measurements from which they stem. Having discarded the solutions based on measurements management, some aggregated accuracy information are still needed in order to deal with spatial data in a correct way, since the derivation of coordinates from observations is a unique but not reversible operation [49]. In particular, we suppose that a set of metadata describing accuracy of absolute positions and accuracy of relative distances have been assigned to each database. This is a very common case in practice, for which at least average errors about absolute positions and relative distances can be often recovered or derived by the cartographic scale characterizing the survey process. Moreover, with the SDI growing the specification of such metadata is becoming a recognized practice, e.g. the ISO standard for Land Administration [67] explicitly includes a quality attribute `DQ_Element` for object classes.

Information about accuracy of spatial data should be used in every operation involving these uncertain data; in particular, this is important for integrating new observations coming from different sources, or for correctly interpreting the result of a query. Moreover, the result of an integration of spatial data coming from different sources is a dataset containing spatial data with different accuracies; therefore, it is crucial that accuracy becomes a stable part of spatial data representation.

The explicit representation of accuracy information, together with the spatial objects themselves, leads to the notion of multi-accuracy spatial database, which is the key concept of the approach proposed in this thesis. A *multi-accuracy spatial database* is a spatial database in which objects are characterized by different *positional* accuracy values; eventually each single object, or even each single point, in the database can have its own positional accuracy. Moreover, the creation of a multi-accuracy spatial database highlights the need for new data integration methods to consolidate the huge amount of spatial data belonging to different data sources and characterized by different quality levels. In particular, those methods have to fuse different observations regarding the same specific and identified geographical object (or set of objects) or different objects among which a particular relation holds, producing a new integrated dataaset. Such methods have to consider the metadata describing the quality of both the datasets to be integrated and the resulting one. Notice that this thesis deals with vector representations of spatial data, namely spatial datasets are represented by set of geometries including points, polylines and polygons specified using a list of coordinates in a reference space.

1.1.2 Distributed Geo-Processing

A few decades ago, paper maps were the principal means to synthesize and represent geographic information. Clearly, the manipulation of this information was limited to manual and batch activities. Since then, the rapid development of new technologies to collect and digitize geographic data, together with an increasing demand for both interactive manipulation and analysis of this data, has generated a need for dedicated software applications, namely Geographic Information Systems (GISs). One of the major tasks of GISs is to efficiently manage huge geographical databases of complex information. In an SDI context the amount of spatial data at disposal of each member is growing continuously. This inevitably increases the amount of processing power needed to manage such data, which soon exceeds the capabilities of a traditional monolithic geographical application that runs on a single server or tightly coupled machines. As a consequence, SDI agencies start to share services, tools and processing resources [104], in addition to spatial information.

In this thesis, the term *geo-processing* is used to denote long-running interactive computations that rely on self-contained, specialized and interoperable services. The new approach proposed in this thesis for integrating multi-accuracy spatial data is an example of geographical process. The main characteristics of such kind of processes can be summarized as follows:

- A geo-process is a *distributed* application. Not only spatial data could be geographically and logically distributed in different repositories, but also functionalities can be provided as distributed services [56]. This requirement has determined the investigation of new technologies for supporting distributed data processing, as well as distributed data access, in a transparent way for users.
- A geo-process is an *interoperable* application, which must be able to deal with heterogeneous data represented in multiple data formats. At now many different data models and data formats have been specified and adopted for capturing and managing spatial data (see Sec. 2.1). Each organization should choose its preferred format in addition to a particular model, since data can be used inside an organization for a specific purpose. For addressing this problem the Open Geospatial Consortium (OGC) [4] has developed a set of interoperability standards and many techniques have been studied about the definition and management of ontologies for integrating spatial data in an automatic way, some of them are summarized in Sec. 2.3.
- A geo-process is *collaborative* in nature. It has become evident that the needs for a large variety of spatial datasets and services cannot be satisfied by a single organization. As organizations start to collaborate, they need new mechanisms and tools for defining, executing and coordinating collaborative processes [104].
- A geo-process is an *interactive* activity. Indeed, not all operations can be automated due to technological limits or economical reasons, and human factors play a central role in the automation of such processes, i.e. the knowledge of domain experts is as important as the provided data. Indeed, many processing activities cannot be completely automated and the human intervention is required during computation. In other cases, the development of fully auto-

matic procedures may not be reasonable, even if technologically feasible, for instance, to manage exceptional situations that rarely occur. For example, the resolution of certain situations during the harmonization of two datasets may require several interventions of a domain expert.

- A geo-process often includes *real-time* requirements. In order to be more integrated in human work, geographical applications should respond to requests from users, process distributed data, and perform sophisticated analysis in acceptable time: some information become useless if it is not provided in time. These requirements force an high level of sophistication, because the management of computing resources should be transparent for GIS users, that may not be able to deal with low level optimizations [57].
- A geo-process can require *flexibility*. A geo-process may be more or less stable, it can require ad-hoc computations that are used only once, or it can need to be enhanced because far from being perfect, or because changes have happened in the application context. As a result, geo-processes have to be flexible enough to meet the new emerging requirements as soon as possible and with less effort as possible. An overview of methods, concepts and terminology concerning process flexibility is given in [106].

The implementation of a system for this kind of geo-processing can be extremely challenging: there is the need to integrate existing systems and to expose them over the network, causing new problems related to security and reliability. In addition, these systems have also to be flexible for easily adapting to new environmental conditions and lowering maintenance costs. Workflow Management Systems (WfMSs) can be good candidates for tackling these emerging challenges. This class of systems will be analyzed in Part II as candidate solution for supporting the exposed requirements.

1.2 Motivating Example

The integration framework proposed in this thesis has been inspired by a real-world project regarding the creation of a regional SDI in Lombardy, a north-east region of Italy. In particular, the construction of such SDI has been started in 2008 and has involved several consolidated integration techniques, also presented in Sec. 2.3. From this project the development of a new integration framework has been designed, considering the spatial data of a pilot municipality, called Cremona. A small representation of such data is given in Fig. 1.1. This section illustrates some issues that can arise during an integration of spatial data when metadata about positional accuracies are not considered.

The considered scenario requires to integrate together a Regional DataBase containing less accurate data, called RDB, with a Local DataBase containing more accurate and up-to-date information, called LDB. Notice that in real-world situations, it is often cost-prohibitive to start again and collect new datasets of higher positional accuracy from scratch. In particular, in the considered case there is the need to upgrade the content of RDB using the available higher accuracy data contained in LDB, and to maintain the existing information in RDB when no updated data are available. We assume that the two datasets contain information

represented with the same level of details and at the same scale. Moreover, in certain cases the real data have been slightly modified in order to better highlight the found problems and illustrate the properties of the proposed integration technique.

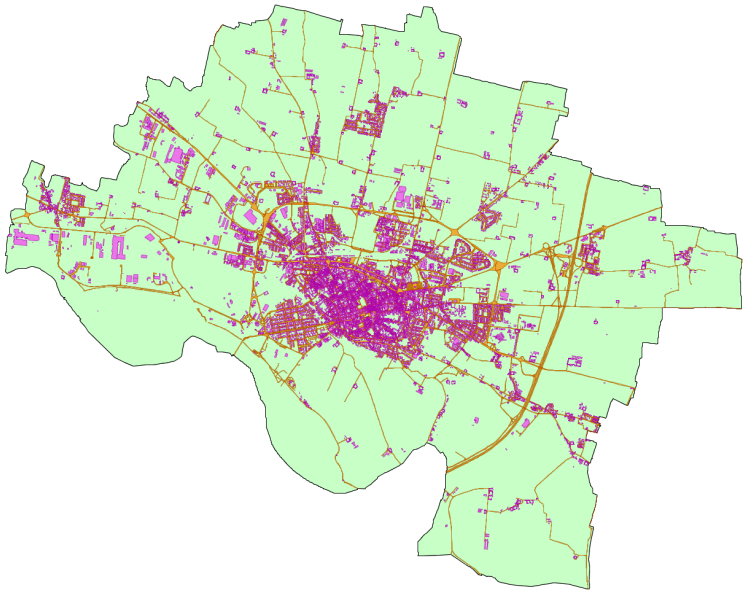


Fig. 1.1. Territory of the Cremona municipality. The pink polygons represents buildings, while the orange ones are street areas.

Since the two datasets have different quality characteristics, when they are overlaid corresponding features may not perfectly align and also the spatial relationships that exist between their features may be violated, this is illustrated by the following examples.

Example 1.1 (Resulting database accuracy). Let us consider the situation in Fig. 1.2 where (a) represents the content of RDB regarding a cross between two streets: the horizontal one, labeled as "M012AT',' is owned by the Cremona municipality, while the vertical one, labeled as "P123SX", is under the region competency. Fig. 1.2.b depicts the updated information collected by the municipality; in particular, in recent years the existing cross has been replaced by a roundabout. If this updated information is integrated in RDB by simply substituting the old representation of the municipality street with the new one, the situation shown in Fig. 1.3 will be generated and the regional street will result disconnected from the roundabout. Such situation is usually solved by applying some rubber-sheeting transformation, producing a geometry that does not strictly adhere with the real street shape. Anyway, the most important issue generated by this kind of integration regards the quality level of the obtained result. Surely, it does not have the accuracy of any of the source databases: we cannot state that it has the accuracy of the more accurate LDB, nor of the less accurate RDB. Soon, after some of this integrations, the quality

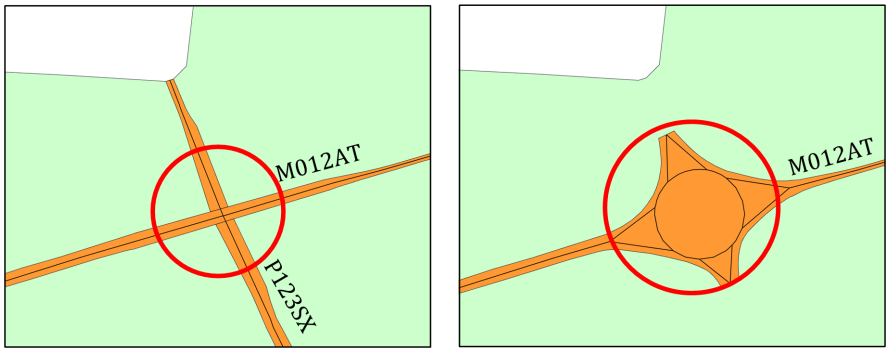


Fig. 1.2. Content of the two databases before the integration: (a) represents the content of RDB, while (b) represents the content of LDB. The vertical street labeled as "P123SX" is under the region competency, while the street labeled as "M012AT" is owned by the municipality.

of certain parts of the obtained database can become very low, independently from the accuracy of the integrated data. □

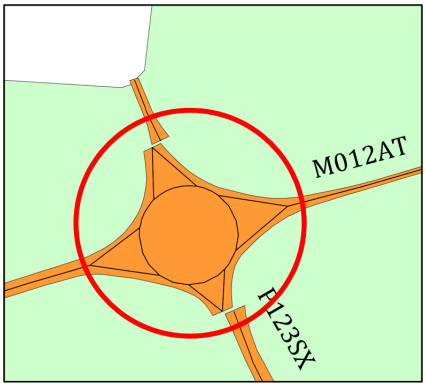


Fig. 1.3. Result of an integration that simply replaces the old geometry of the municipality street in RDB with the new one contained in LDB: the regional street results disconnected from the roundabout.

Example 1.2 (Relative distances). Fig. 1.4.a shows some buildings and street areas contained in RDB. Inside this area the new hospital depicted in Fig. 1.4.b has been built and collected in LDB. We assume that some relative distance information is available between certain buildings and a reference point representing a bus stop. This bus stop is contained in both databases but with different absolute positions.

Regardless the integration is performed by placing the new hospital in (a), or the other buildings in (b), different distances are generated between the buildings and the reference point. In particular, if the new hospital is placed into RDB its distance with respect to the reference point changes from 75 meters to 68 meters,

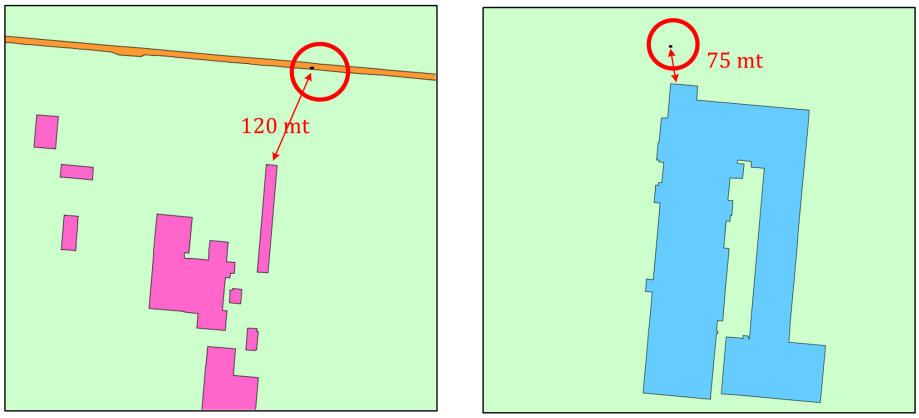


Fig. 1.4. A content fragment of the two source databases before the integration: (a) shows the RDB buildings and streets, a distance of 120 meters exits between a building and a reference point representing a bus station; (b) shows the new hospital contained in LDB, a distance of 75 meters exists between it and the reference point.

while if we maintain the absolute position of the reference point in LDB, the distance between the existing buildings and the reference point changes from 120 meters to 127 meters; moreover, maintaining the LDB position of the reference point, its location is no longer contained into the street area.

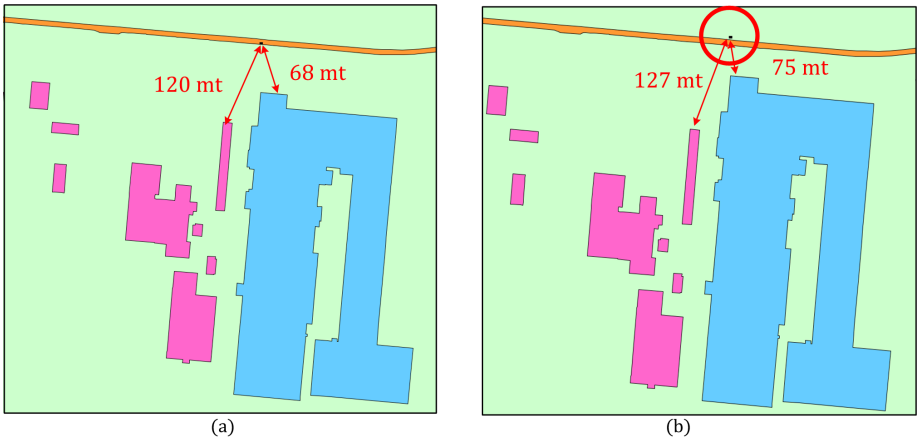


Fig. 1.5. (a) Result of placing the new hospital building into RDB: the final distance between it and the reference point changes from 75 meters to 68 meters. (b) Result of placing the old buildings into LDB: the distance between them and the reference point changes from 120 meters to 127 meters, while the reference point is no longer contained into the street area.

This is due to the fact that the accuracy of object coordinates is usually less than the accuracy of relative distances among object vertices. This may lead to the

situations in Fig. 1.5 where the available representation is very accurate in terms of object shapes but less accurate with respect to absolute positions and relative distances. □

Example 1.3 (Topological relations). Fig. 1.6 illustrates the result of inserting into RDB a new building contained in LDB, which is depicted in green and labeled as new_A. If the new building is overlaid on the RDB content, an overlap is obtained between it and the surrounding ones. □

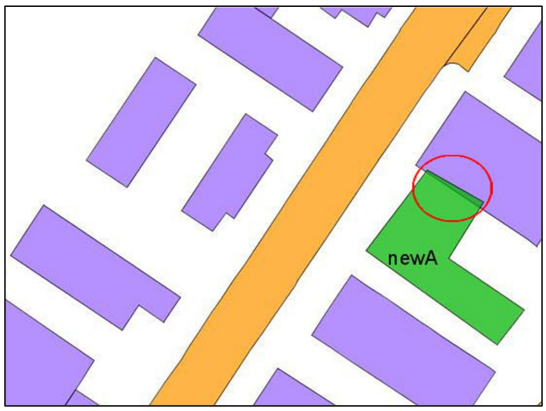


Fig. 1.6. Result of a naïve integration of new polygons into RDB: an overlap is obtained between the building with label new_A and the surrounding ones.

In order to overcome the problems previously exemplified, this thesis proposes as a core contribution a statistical method that is able to generate an effective integration between new observations and the current database content.

Finally, we consider the problem of successfully applying the integration procedure in a distributed context, such as an SDI environment.

Example 1.4 (Distributed integration). Let us suppose that the territory in Fig. 1.1 is subdivided between two SDI members, as depicted in Fig. 1.7 using two different colours. A new building, labeled as new_B, has to be added to the local dataset of Mb_1 very close to the boundary between Mb_1 and Mb_2. If the integration procedure is only applied to this local database, some inconsistencies on the boundary between Mb_1 and Mb_2 can occur. For instance, as depicted in Fig. 1.8, the road area crossing the two datasets, that is connected before the integration, can become disconnected. These inconsistencies can be determined by two kinds of problems:

(i) Some points of the boundary objects are shared by adjacent datasets and they cannot be moved only on one side of the border. In other words, the integration effects have to be propagated also on Mb_2.
(ii) The local accuracy information is partial with respect to the objects surrounding new_B. In the case considered in Fig. 1.8, we know only some measures of relative distance between new_B and the road area close to it, depicted as red

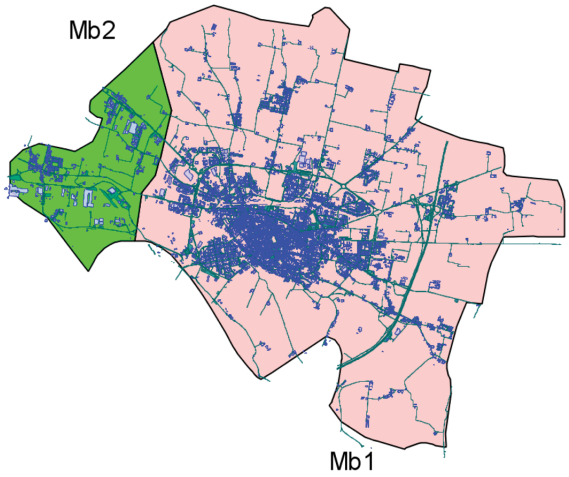


Fig. 1.7. Subdivision of the considered region between two SDI members Mb_1 and Mb_2.

lines. In order to obtain a correct integration, the accuracy information for objects of Mb_2 that are near to new_B are also needed. □

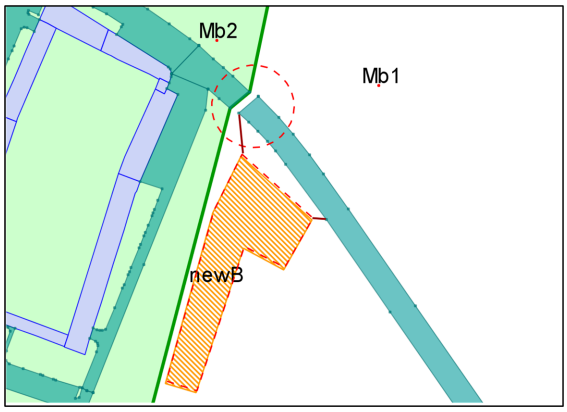


Fig. 1.8. Example of problem that may arise when an integration is performed locally without propagating its effects to the adjacent datasets. In particular, the road crossing the two dataset becomes disconnected, since only the portion laying in Mb_1 is modified.

1.3 Contribution

The aim of this thesis is to solve the problems exemplified in the previous section by proposing a framework for dealing with multi-accuracy spatial databases in a distributed context. More precisely our contribution is articulated into two major

parts: Part I presents the theoretical statistical framework for the representation and integration of multi-accuracy spatial data, while Part II discusses the implementation of such framework in a distributed context using workflow technologies. In more details, Part I deals with the following problems:

1. The definition of a data model for describing a Multi-ACuracy Spatial database, called MACS model. In such model, metric observations, (i.e. the object coordinates) are represented using probability theory, while logical observations (i.e. topological relations among objects) are described with a logical approach.
2. The definition of a procedure to derive accuracy information for coordinates starting from the available metadata. In particular, this thesis considers the case, in which only coordinates are stored inside a spatial database and just few aggregate accuracy metadata are available, i.e. a maximum granted error for absolute positions and a maximum grated error for relative distances among objects.
3. The definition of a procedure for integrating two MACS databases and producing a new MACS database. More specifically, such procedure has to take into account the accuracy of both source databases and to produce new accuracy information for the integrated one. The integration procedure is composed of three steps: the first one integrates together the metric information contained in the two source datasets, the second one integrates together their logical observations; finally, the last step checks the coherence between metric and logical information, and eventually adjusts the resulting metric observations for satisfying the logical ones. Some formal properties of the integration procedure are also discussed.
4. The definition of a distributed version of the integration procedure in order to make it applicable in a SDI. Such distributed version is characterized by two steps: in the first one each local agency performs a partial integration on its own data, then in the second one the locally integrated data are transmitted back to a central agency, which applies a global integration by propagating the effects of the locally performed ones. Moreover, the central agency is responsible for notifying the other involved local agencies with the integrated data of their interest.
5. The definition of a procedure for reducing the number of information that has to be transferred to and from the central agency, in order to make the technique effective in a distributed context, even in presence of a huge amount of data.
6. Finally, the application of the proposed framework in a real world scenario is illustrated.

The proposed integration process presents many of the characteristics highlighted in Sec. 1.1.2. Its implementation can become extremely challenging not only for the complexity of the involved operations, but also for the need to involve different agents that concurrently participate to its realization. For this reason, Part II evaluates the possibility to implement such process exploiting workflow technologies. In particular, the following contributions are presented:

1. An overview of the rationale underlying existing workflow technologies and an introduction to some representative WfMSs.

2. A comparison between the considered existing workflow technologies first from a more general point of view, using the widely accepted method of workflow patterns, and secondly from the specific geographical point of view. In particular, for this second comparison, the implementation of the integration process presented in Part I is considered, and the strengths and limits of existing WfMSs are highlighted. These two comparisons leads to the definition of the requirements for an ideal geographical workflow system.
3. The introduction of a novel business process modeling language, called NestFlow, which has been developed by our University as result of another PhD thesis. More specifically, besides the language presentation, we discuss the specific extensions we have introduced for allowing the modeling of geographical processes, and we present the design of the integration process using this modeling language.

1.4 Thesis Structure

This section summarizes the structure of the thesis and the content of each chapter, eventually reporting its corresponding publications.

Part I Geo-Processing in Theory

- Chap. 2 summarizes some background notions and related researches about the representation spatial data and topological relations, the integration of spatial data coming from different sources, the modeling and managing of multi-accuracy spatial data.
- Chap. 3 introduces the proposed Macs model for representing multi-accuracy spatial data. It illustrates the theoretical background and explain how accuracy parameters can be derived starting from the available metadata.
- Chap. 4 introduces the statistical integration framework based on the MACS model described in Chap. 3. More specifically, three integration aspects are presented: (i) the integration of metric observations, (ii) the integration of logical information, and (iii) the integration of metric and logical information together. Such framework is then extended in order to apply it in a distributed context, such as an SDI environment. Finally, the application of the integration framework is discussed using the real-world scenario introduced in Sec. 1.2.

The content of these chapters has been published in [13–15].

Part II Geo-Processing in Practice

- Chap. 5 presents the technological infrastructure considered in this thesis and describes some representative WfMSs.
- Chap. 6 compares the considered WfMSs from two distinct points of view. Firstly, they are compared in general, using the well-known technique of workflow patterns, a tool widely used by the business process community for evaluating the expressiveness and suitability of WfMSs. Secondly, they are compared

from a more specific geographical viewpoint, by trying to design the proposed integration framework. From these comparisons, we delineate the essential requirements of an ideal spatial WfMSs.
- Chap. 7 introduces a novel language for modeling geographical processes, called NestFlow that meets the requirements highlighted in Chap. 6. This presentation will highlight the extensions we have made for managing geo-processes. Moreover, we will discuss the NestFlow implementation of the integration framework proposed in Part I.

The content of Chap. 6 related to the comparison between business and scientific WfMSs in terms of workflow patterns has been published in [88], while the comparison related to geographical requirements has been published in [87]. Finally, the description of the NestFlow modeling language contained in Chap. 7 has been published in [29, 30, 47, 48].

Appendix

- App. A summarizes some probability and statistical notions that are useful for understanding this thesis.
- App. B contains a brief description of the mentioned workflow patterns.

Part I

Geo-Processing in Theory

2

Background: Spatial Data Management

This chapter provides an overview of several basic concepts and introduces some related work useful for understanding the first part of the thesis. In particular, it starts by recalling some notions that are well-known in the geographical context, such as the different existing methods for representing spatial data (Sec. 2.1) and the definition of the most widespread relations between spatial objects (Sec. 2.2). Subsequently, it introduces the integration problem in the general case and summarizes some existing integration techniques that represent the starting point for this thesis development (Sec. 2.3). It then formalizes the concept of uncertainty and accuracy in spatial data (Sec. 2.4) and it presents some research efforts about the representation of uncertain spatial objects (Sec. 2.5) and uncertain topological relations (Sec. 2.6). Finally, it treats the spatial data integration problem when metadata about accuracy are considered (Sec. 2.7). These last contributions can be considered as alternative approaches to the one given in this thesis; therefore, similarities and differences between them are also discussed.

2.1 Spatial Data Representation

A *geographic object* is a real world entity related to a specific location on the Earth surface. In particular, each geographic object is characterized by the following components:

- several *thematic* or *descriptive attributes*: alphanumerical attributes that defines characteristics and nature of the object, for instance, the name of a town or the number of its inhabitants;
- a *spatial component* or *geometric attribute* which defines the location, shape, orientation and size of the object in 2D or 3D space;
- a set of *topological relations* which describes the relations exiting among the object and the surrounding ones.

Geographical objects can be described using two fundamental models: object-based or field-based [89,90]. The *object-based model*, also referred to as *entity-based* or *feature-based model*, considers the real world as a surface populated by several

uniquely identified objects. As mentioned above, geographic objects are characterized by as set of spatial and non-spatial attributes. In particular, the basic types for representing geometries in the 2D space are: point (zero-dimensional objects), line (one-dimensional objects), and surface (two-dimensional objects). *Points* are normally used for representing the location of entities whose shape is not considered useful, or when the area is quite small with respect to the embedding space size. *Lines* or linear objects are commonly used for representing networks (road, hydrography, and so on). The basic linear geometric type considered in real systems is the polyline; namely, a set of linear segments, such that each segment endpoint (called vertex) is shared by exactly two segments, except for two endpoints (called extreme points) which belong to exactly one segment. A special case of polyline is the closed one, in which the two endpoints are identical. Such type of polyline is used to describe the boundary of a polygon. *Polygons* are the most common implementation of surfaces and are used for representing entities with a significant extension or a particular relevance.

Object-based models are suitable for describing the content of a map: the set of buildings, streets, lakes, rivers in a given region, the set of government entities in a territory (municipalities, province, region, etc.), and so on. Such models are similar to those normally used in traditional information systems, since the attributes are stored as entity properties and one of these properties describes the object relation with the space. An example of representation using an object-based model is illustrated in Fig. 2.1.a, where each Italian region is identified by a particular integer value.

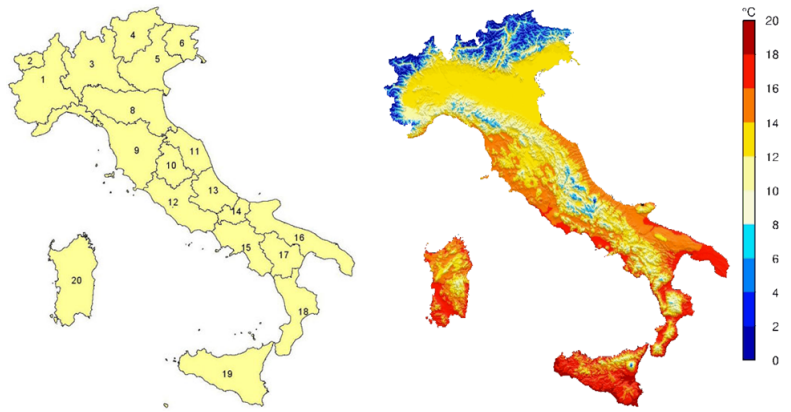


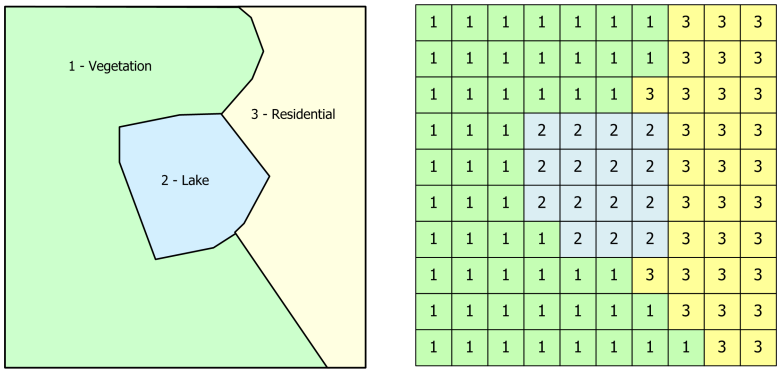
Fig. 2.1. (a) Object-based representation of the Italian regions: each polygon representing a region is identified by a unique integer identifier. (b) Field-based representation of the mean annual temperatures in Italy: red regions are those with higher temperatures, while the blue ones are those with lower temperatures.

In the *field-based model* the real world is considered as a continuous surface on which features vary also in a continuous manner. The geographical information is represented by several functions from the reference space to the set of possi-

ble values that the represented property can assume. Such functions describe the distribution of the attribute values on the space. Field-based models are suitable for describing spatial information such as the average temperature registered in a particular month, the average amount of rainfall in a year, the land use classification, the altimetry, and so on. An example of representation using a this model is reported in Fig. 2.1.b, where is described the mean annual temperatures in Italy.

In the remainder of this thesis an object-based model is assumed for the description of geographical objects, in particularly during the MACS model definition. As regards to the representation of their spatial component, geometry can be described and stored using a vector or raster format [107]. In a *vector format* spatial data is represented by the coordinates of points, lines or polygons which constitute its *geographical extent*. In particular, in a 2D space a point is represented by a pair of coordinates $p = (x, y)$, a polyline is represented by a list of points which are the endpoints of its constituent segments $l = (p_1, p_2, \ldots, p_n)$, and finally a polygon is represented as a closed polyline that represents its boundary $pl = (p_1, p_2, \ldots, p_n)$ where $p_1 = p_n$. An example of vector representation is illustrated in Fig. 2.2.a.

In a *raster format* the continuous space is approximated by a discrete one through the construction of a regular grid, in which each cell is associated to an alphanumeric value representing a property value. An area or region is then obtained by the set of adjacent cells with the same value. An example of raster representation is illustrated in Fig. 2.2.b.



(a) (b)

Fig. 2.2. (a) Example of representation using a vector format. (b) Example of representation using a raster format.

In this thesis the vector format is assumed for the representation and storage of spatial components, in particular a statistical approach will be proposed for describing the coordinates of each location. As regards to this kind of representation, a technique is needed in order to assign a value to each coordinate. The terms *geo-reference*, *geo-locate* or *geo-code* are used to denote the act of assigning locations to atoms of information [80]. The primary requirements of a geo-reference are: (1) it has to be unique, namely there is only one location associated with a given geo-reference, and (2) its meaning shall be shared among all the peo-

ple who wish to work with the given information. Several geo-reference systems have been developed [107], among them this thesis considers two coordinate systems that enable every location on the Earth surface to be specified by a set of numbers: the Cartesian coordinate system and the geographic latitude and longitude system. The *Cartesian coordinate system* uses two perpendicular axes (x and y) whose intersection generates the origin of the system and localizes the point $(x = 0, y = 0)$. All positions on the plane are determined by two (positive or negative) values, denoted as coordinates x and y, which specifies the horizontal and vertical position of the point with respect to the system origin. On the contrary, in a reference system based on *longitude and latitude*, the coordinates are define in terms of angles with respect to well-defined references: the Royal Observatory of Greenwich, the center of mass, and the axis of rotation. Latitude and longitude constitute the most comprehensive system of geo-referencing. It supports a wide range of analysis, including the computation of the distance between two points on the curved surface of the Earth. However, many technologies for working with spatial data are inherently "flat", including paper maps, and GISs that usually deal with a flattened projection of the Earth surface. Therefore, some projection techniques are necessary in order to transform a latitude and longitude system into a Cartesian one. Several projection techniques are available, each of them preserves certain distinctive properties, such as distance, direction, shape or area. More specifically, projection techniques are classified on the basis of the preserved property [80]: *conformal* projections ensure that the shape of small features on the Earth surface is preserved, in other words that the projection scale in the x and y directions are always equal; *equivalent* projections ensure that areas measured on the map are always in the same proportion to areas measured on the Earth surface, and *equidistant* projections preserve the relative distance between points. None of the available projection techniques allow to simultaneously preserve all these properties and there is not a better projection system for all the cases, but the best projection method depends on the future use of the spatial data.

Besides their distortion properties, another common way to classify projection techniques is by analogy to the physical model describing the relation between positions on a flat map surface and positions on the curved Earth surface. Three major classes can be distinguished [80]: cylindrical projections, azimuthal or planar projections and conic projections. *Cylindrical* projections are analogous to wrap a cylinder paper around the Earth, projecting the Earth features onto it, and then unwrapping the cylinder. *Planar* projections are analogous to touch the Earth with a sheet of flat paper. Finally, *conic* projections are analogous to wrap a sheet of paper around the Earth in a cone.

2.2 Spatial Relations

In literature several kinds of relations have been defined between spatial objects. In this thesis the notion of spatial relation is considered as a mean for describing the mutual relation that exists between the position of spatial objects. In particular, since these properties are not subject to a measurement process, but they are observed to exist or not, they can be expressed as true or false propositions and treated using a logical approach.

This section summarizes the principles underlying three well-known types of spatial relations: topological, cardinal and distance relations. Anyway, in the next chapters we focus on topological relations, since they are the mostly used and frequently available in the existing geographical systems.

2.2.1 Topological Relations

Topological relations are a subset of spatial relations characterized by the property of being preserved under topological transformations, such as translation, rotation and scaling. These relations are commonly described using the *9-intersection model* originally defined in [38, 39]. In the 9-intersection model, the geometry of each object A is represented by 3 point-sets: its interior A°, its exterior A^-, and its boundary ∂A. The definition of the binary topological relation existing between two spatial objects A and B is based on the 9 possible intersections of each object component. Thus, a topological relation $R(A, B)$ can be represented as a 3×3-matrix, called *9-intersection matrix*, defined as:

$$R(A,B) = \begin{bmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{bmatrix}$$

For each element i, j in the matrix, the value empty ($R_{i,j}(A, B) = \varnothing$), or not empty ($R_{i,j}(A, B) \neq \varnothing$) is considered. In such way, many relations can be distinguished between surfaces, curves and points. In particular, the boundary of a geometry is defined as follows: a surface boundary is the ring defining its border, the boundary of a curve is composed of its end points and the point boundary is empty.

Starting from this model, several refined definitions of topological relations have been proposed. In particular, since the objects considered in this thesis can have geometries of different types (point, curve and surface), the set of adopted topological relations is the one defined by Clementini et al. in [25]. In their work the authors extend the original model by considering for each 9-intersection its dimension (i.e., 0 for points, 1 for curves and 2 for surfaces), giving raise to the *extended 9-intersection model*. The number of such relations is quite high; hence, different partitions of the extended 9-intersection matrices have been defined, grouping together similar matrices and assigning a name to each group. In [25] the authors proposed the following set of binary topological relations $\{disjoint, touch, in, contain, overlap, cross, equal\}$, defined as:

- $A\ disjoint\ B \iff A \cap B = \varnothing$
- $A\ touch\ B \iff (A^\circ \cap B^\circ = \varnothing) \wedge (A \cap B \neq \varnothing)$
- $A\ in\ B \iff (A \cap B = A) \wedge (A^\circ \cap B^\circ \neq \varnothing)$
- $A\ contain\ B \iff B\ in\ A$
- $A\ overlap\ B \iff dim(A^\circ) = dim(B^\circ) = dim(A \cap B) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$
- $A\ cross\ B \iff dim(A^\circ \cap B^\circ) = max(dim(A^\circ), dim(B^\circ)) - 1 \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$
- $A\ equal\ B \iff A = B$

This is a complete set of mutually exclusive topological relations, namely a set of topological relations in which for each pair of objects there is one and

only one possible relation. Such relations have been implemented in many GIS systems and included in OCG and ISO standards. Notice that some relations can be realized only in presence of particular object types. Tab. 2.1 summarizes for each topological relation the pair of types that can realize it. For instance, an *equal* relation can exist only between two objects of the same type, while a point can never contains a polygon or a curve, and so on.

	S/S	C/C	P/P	S/C	S/P	C/S	C/P	P/S	P/C
disjoint	✓	✓	✓	✓	✓	✓	✓	✓	✓
touch	✓	✓		✓	✓	✓	✓	✓	✓
in	✓	✓				✓		✓	✓
contain	✓	✓		✓	✓		✓		
overlap	✓	✓							
cross		✓		✓		✓			
equal	✓	✓	✓						

Table 2.1. This table reports for each topological relation the pair of geometric objects that can realize it. S denotes the surface type, C the curve type, and P the point type.

2.2.2 Cardinal Directional Relations

Cardinal directional relations provide a way to determine what is the relative position of a target object with respect to a reference object by considering some cardinal directions. The idea behind a qualitative representation of cardinal rela-

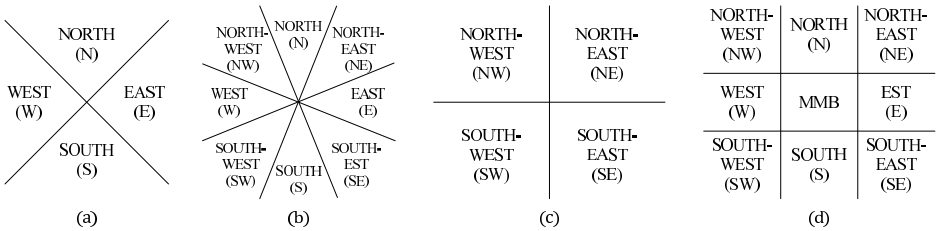


Fig. 2.3. Different space divisions: (a) four cones, (b) eight cones, (c) four half-planes, (d) nine tiles.

tions is to map quantitative directional information (i.e. the degrees of an angle) into a set of symbols. More precisely, given a *reference* point P_1, which is used to define directions, and a *target* point P_2, whose direction with respect to the reference point has to be detected, the directional relationships are binary functions that map the pair (P_1, P_2) into a symbolic direction d.

The number of available direction symbols depends on the used cardinal directions model. The basic model divides the space around a reference point into

triangular areas or into half planes, as shown in Fig. 2.3.a, Fig. 2.3.b, and Fig. 2.3.c. If the reference object is not a point, the most used model, called D_9 model [52], is based on the space decomposition presented in Fig. 2.3.d. It approximates the reference object with its minimum bounding box (MBB), and subdivides the space around the reference object into distinct areas, called *tiles*, using the infinite extensions of its MBB sides. The tiles result in the usual set of directions $\{N, NE, E, MBB, SE, S, SW, W, NW\}$. Let us notice that all the tiles are unbounded and their union coincides with $\mathbb{R}^2$. Based on the D_9 model, cardinal directions between two regions can be represented as 3×3 matrices containing the value empty or non-empty for each intersection between the target object A and the tiles generated by the reference object B:

$$dir_{RR}(A,B) = \begin{bmatrix} NW_B \cap A & N_B \cap A & NE_B \cap A \\ W_B \cap A & MBB_B \cap A & E_B \cap A \\ SW_B \cap A & S_B \cap A & SE_B \cap A \end{bmatrix}$$

In [51, 53], this basic model has been extended to deal with lines, points, and regions without holes. In this case, a cardinal directional relation is represented again as a 3×3 matrix, but now each cell contains a 9-cells vector, called *neighbor code*, which records information about the intersections between the boundary of a particular tile and the target object A, using 9 bits. Bit 0 (x_0) records the value of the intersection between A and the direction tile the vector refers to, called DT, and bits $x_1 - x_8$ record the values of the intersection between A and the left (L), bottom-left (BL), bottom (B), bottom-right (BR), right (R), top-right (TR), top (T) and top-left (TL) boundaries of DT, respectively.

x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
TL	T	TR	R	BR	B	BL	L	DT
256	128	64	32	16	8	4	2	1

Each neighbor code corresponds to a number between 0 and 256. Therefore, each matrix can be seen as a 3×3 matrix of integer numbers: different matrix configurations correspond to different cardinal relations. However, by considering only connection objects, not all possible configuration represents a correct cardinal directional relation.

The information contained in neighbor codes can also be represented by using a 5×5 matrix, called *directional matrix*.

$$D(A,B) = \begin{bmatrix} NW_B \cap A & n\text{-}nwl_B \cap A & N_B \cap A & n\text{-}nel_B \cap A & NE_B \cap A \\ w\text{-}nwl_B \cap A & nwp_B \cap A & nl_B \cap A & nep_B \cap A & e\text{-}nel_B \cap A \\ W_B \cap A & wl_B \cap A & MBB_B \cap A & el_B \cap A & E_B \cap A \\ w\text{-}swl_B \cap A & swp_B & sl_B \cap A & sep_B \cap A & e\text{-}sel_B \cap A \\ SW_B \cap A & s\text{-}swl_B & S_B \cap A & s\text{-}sel_B \cap A & SE_B \cap A \end{bmatrix}$$

In such matrix, a row and a column exist for each tile interior and each boundary between two tiles, according to the space subdivision in Fig. 2.4.

Each matrix element can assume the value empty or non-empty, depending on whether the target object A intersects or does not intersect the corresponding portions of space. A formal model for cardinal relations based on a 5×5 matrix is presented in [120, 121].

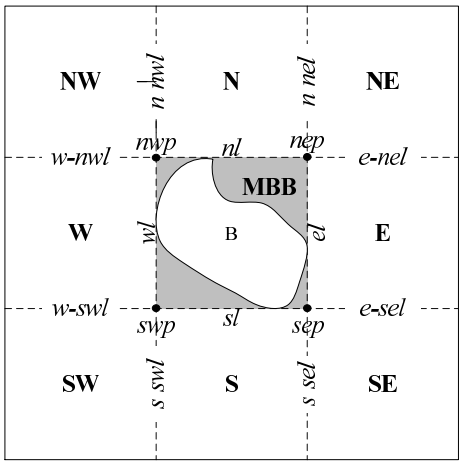


Fig. 2.4. Space subdivision corresponding to the direction matrix.

2.2.3 Distance Relations

Distance relations are based on the definition of a distance function in the reference space. For instance, given two points $P = (x_p, y_p)$ and $Q = (x_q, y_q)$ the Euclidean distance between them is computed as follows:

$$d(P,Q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

This distance function is well suited in presence of point objects, contrarily when geographical objects with a significant extension are considered, the concept of distance shall be properly redefined. For example, let O_1 and O_2 be two generic geographical objects, the distance between them can be extended as follows:

$$distance(O_1, O_2) = min(\{d(P,Q) \mid P \in O_1 \wedge Q \in O_2\})$$

namely the distance between them is the minimum distance between any pair of points composing their geometry. This definition of distance between polygons can become quite unsatisfactory for some applications. Indeed, in many cases the intended notion of small distance between polygons requires that no point of one polygon is far from any point of the other polygon, while the previous definition considers only the smallest distance between any two points, independently from the distance between the other ones. In order to avoid such issue, the Hausdorff distance has been defined which considers the maximum distance of a set of points to the nearest points in the other set. More formally, the Hausdorff distance is defined as follows:

$$h(O_1, O_2) = \max_{a \in O_1}(\min_{b \in O_2}(d(a,b)))$$

Let *dist* a chosen distance function between geometric objects, and (d_1, d_2) a pair of distance values, a distance relation R can be defined between two objects O_1 and O_2 as follows:

$$R_{(d_1,d_2)}(O_1, O_2) \Leftrightarrow (d_1 < dist(O_1, O_2) < d_2) \tag{2.1}$$

In other words, a relation exists between two objects O_1 and O_2, with respect to a pair of distances (d_1, d_2), if and only if the distance between O_1 and O_2 is greater than d_1 and smaller than d_2. Let us consider the situation in Fig. 2.5 and suppose $d_1 = 0$ and $d_2 = d$, from Eq. 2.1 it results that A is in relation with C, E and F, but not with B, D, G and H.

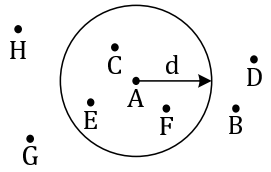


Fig. 2.5. Example of distance relation with $d_1 = 0$, $d_2 = d$ and $O_1 = A$.

In some cases, the distance relation between two objects is not defined using a precise metric observation, but using abstract concepts of proximity and distance. Such relations are called *approximated distance relations*. A set of approximated distance relations has a finite cardinality, for instance the following is a valid set:

$$\{same\_location, very\_near, near, medium, far, very\_far\}$$

The definition of such elements is usually based on the specification of some distance intervals and a particular ordering among them can be derived. For instance, relatively to the previous set, the following ordering can be defined:

$$same\_location < very\_near < near < medium < far < very\_far$$

Besides to the definition of particular distance relations, several queries based on the distance notion can be used in real-world applications. A *distance* or *buffer query* selects all the geographical objects which lay on a distance between a minimum value d_1 and a maximum value d_2 from a reference point or another geometric value. For instance, let us consider an object A with geometric attribute g, denoted as $A.g$, a reference point P, and two distance values $d_1 = 0$ and $d_2 = d$. A selection based on the distance can be the following:

$$\sigma_{buffer(P,0,d)\ overlap\ A.g}(A)$$

which selects all the objects A whose geometry g overlaps a buffer with ray d centered on the reference point P. As regards to the situation depicted in Fig. 2.6, the previous query produces as result polygons P_1 and P_5, while if the relation *in* is considered in place of the *overlap* one, the query returns only polygon P_2.

2.3 Spatial Data Integration

Information fusion is the process of integrating information coming from different sources to produce new information with added value, reliability, or usefulness [33, 138]. Geographical information fusion is an important function of any interoperable

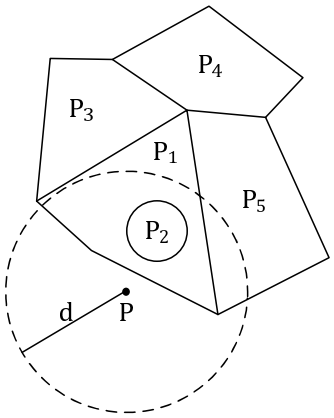


Fig. 2.6. Example of distance query with $d_1 = 0$ and $d_2 = d$.

GIS application. In the remainder of this thesis the terms integration and fusion are used as synonymous.

The integration of two spatial datasets essentially requires to identify different objects having a semantic link, and solve structural (schematic) differences existing between linked objects. Schematic conflicts between two objects with a semantic link may arise when equivalent concepts are represented differently in local data sources. Those conflicts can be associated with names, data types, units, attributes, and so on. Another kind of conflict can arise in the representation of the same attribute; for instance, an address can be represented using only one or multiple attributes. In the GIS community many efforts have been focused on interoperability aspects. Most of these approaches propose to enrich local data models in order to conform to a unified model, and the creation of the Open Geospatial Consortium (OGC) [4] together with the ISO TC211 standard series are the most visible outcome of such efforts.

The major issues that characterize the data integration process are: (1) developing a correct understanding of existing data semantics, (2) establishing an accurate correlation structure, and (3) choosing a well-suited integrated description, based on some integration goals and the available data conversion techniques. Such issues determine the development of three phases in the data integration process [34]:

1. *Schemas preparation* – this phase includes all preliminary activities which aim to obtain the convergence of existing descriptions towards a uniform pattern. Fall into this phase: the enrichment and completion of existing descriptions with additional information to achieve a uniform level of understanding of data semantics, the establishment of a global and local thesauri to ease information exchange, and so on.
2. *Correspondences investigation* (including conflicts detection) – this phase aims at the identification and precise description of all correlations among existing schemas and among existing data objects. Such phase has been deeply treated in literature, and some results are reported in Sec. 2.3.1.
3. *Integration* – this phase aims to solve possible conflicts, and creating the integrated description as a virtual database on top of the existing data sources.

The integration is performed both at schema and at instance level, producing a new database where the content has been properly transformed to adhere to a common schema. Such phase is treated in details in Sec. 2.3.2 and Sec. 2.3.3. In particular, Sec. 2.3.3 deals with an important and characteristic aspect of spatial data integration: the harmonization of geographical feature components.

As discussed in the introduction, this thesis assumes that a previous integration phase has been performed, using some of the techniques cited before and summarized in the following sections. In other words, the content of the two databases are comparable and specified with respect to a common schema.

2.3.1 Ontology-Based Integration

The main problem facing any geographical information fusion is the *semantic heterogeneity*, where the concepts and categories used in different geographic information sources have incompatible meanings.

The semantics of an information source may be described using an *ontology*. The task of fusing information compiled through different ontologies is a classical problem in information science [144], and continues to be a highly active research issue with many topics, including databases, interoperability, semantic Web, medical information systems, knowledge representation, data warehousing, and of course, geographical information systems.

A clear distinction is usually made between the process of identifying the relationships between corresponding elements in two heterogeneous ontologies, called *ontology integration*, and the process of constructing a single combined ontology based on these identified relationships, called *ontology alignment*.

Fonseca et al. in [44–46] introduces the concept of *ontology-driven GIS* which aims to augment conventional GISs with formal representation of geographical ontologies, leading to tools that enable improved ontology-based information integration. A wide variety of related work has addressed the issue of geographical information integration using ontologies [11,19,34,122,134,135]. However, all these researches focuses on the integration itself, assuming that the semantic relation between two ontologies is already known.

A relatively small amount of work has begun to provide techniques for geographical ontology alignment. The majority of these works adopts an *intensional approach* which aims to analyze the definitions of the concepts and categories used in the input information sources. Intensional techniques usually analyze heterogeneous ontologies to identify lexical similarities, structural similarities, or some combination of them. Kavouras, Kokla et al. use FCA [125] as the basis for their approach to geographical ontology alignment [70,71,73]. Manoah et al. in [84] apply the intensional machine learning technique to geographical data. Duckham and Worboys investigate in [36] the use of description logics and in [148] the application of a formal algebraic approach. Anyway, the diversity of geographical terms and concepts makes ontology alignment a semi-automated process which still requires human domain expert interactions at critical stages in the alignment process.

The study of Duckham and Worboys in [37] is one of the few attempts to apply an *extensional approach* for automating information fusion. An extensional approach uses specific instances to determine how concepts are actually used. In

particular, at the core of the proposed extensional approach, called ROSETTA, is the process of inferring semantic relationships from spatial relationships. Geographical information is richly structured and voluminous source of instances are available, favoring the use of an extensional approach. However, uncertainty is an endemic issue of geographical information, which leads to the problem of unreliable inference. For these reasons, the authors propose some initial technique to adapt the automated fusion process in order to operate in presence of imperfect and uncertain geographical information.

2.3.2 Schema Integration

One of the main issues faced during the integration of spatial data is due to the schema heterogeneity. Indeed, spatial data can be represented in different formats and can use the data model that best fits the purposes for which they have been collected. Schema integration is the process by which several local schemas are integrated to form a single virtual (global) schema. A *mediation system* is a software system that provides to users a uniform view of the different data sources via a common data model. There are two main approaches to data mediation: in the *global as view* approach, the global schema is defined as a set of views over local schemas, while in the *local as view* one, local sources are defined as a set of views over a given global schema, applying sometimes some domain ontologies. In the first approach query rewriting is straightforward, while adding a new data source is easier in the second one.

Regardless to the adopted data mediation approach, a framework for data transformation is always needed, that is a language for schema mapping. A *mapping* is a rule that allows the specification of one-to-one schema transformations under some constraints. In particular, each mapping is composed of a right term, a left term, and a restriction. The left term is the global schema construct, while the right one consists of one or more paths on the source schema constructs. Such mapping means that the global schema element on the left corresponds to the local schema path(s) under the given restriction. VirGIS [19] is an example of geographical mediation system that complies with OGC recommendations with respect to some of its core software components, that is GML and WFS.

2.3.3 Feature or Point Matching

An important aspect of spatial data integration is the identification of corresponding geometries contained into two distinct databases. Conflation techniques [112] have been widely used for integrating two vector spatial databases. These methods essentially involves two phases: (1) corresponding features in the two source datasets are recognised through the identification of matching control points; (2) the two source datasets are then aligned using rubber-sheeting transformations based upon the identified matching control points. These phases are repeated iteratively, with further control points being identified as the data sources are brought into alignment.

As regards to the first phase, matching algorithms can be classified into three different categories, on the basis of the main considered characteristics:

- *Geometric methods* – they list the objects belonging to the two datasets and compare them on the basis of geometric criteria, such as distances, angles, position relations and shape characteristics. Distance methods state that if the distance between two objects contained into the two considered datasets belongs to a certain value, then the probability they represent the same real-world object is greater. The most widespread distance criteria are: (1) the Euclidean distance, for computing the distance between two points or a point and a curve, and (2) the Hausdorff distance (see Def. 2.1), for computing the distance between two curves or polygons. Other geometric methods consider information about the angles between two linear objects, the direction of a line, the position relation between two objects (e.g. a point is contained or not inside a polygon), or the shape characteristics of the objects.
- *Topological methods* – they analyse the topological characteristics of as set of objects in order to identify the correlation among them. For instance, considering the components of two road networks, two intersections are most likely to represent the same object if their degree, computed as the number of incident roads, is the same, or if they connect objects that have been previously identified as the same road.
- *Semantic methods* – they consider the value of various object attributes during the integration process: if two objects present the same (or equivalent) value in a set of attributes, then they are most likely to represent the same object.

The second phase considers instead rubber sheeting transformations, through which a set of objects, called as a whole a layer, is distorted to allow it to be seamlessly joined to another layer.

Important problems can arise when different representations of the same data are characterized by different level of granularity, details (LoDs) or map scales. This situation can lead to partial or incorrect correspondences between objects.

2.4 Uncertainty in Spatial Data

The term *uncertainty* is frequently used for denoting different concepts, such as vagueness, ambiguity, or anything that is undetermined. In the field of geographical information science, the uncertainty of a measure can be described by a range of values that possibly includes the true value of the measured object. Three main interpretations of uncertainty can be distinguished: imprecision, ambiguity, or vagueness, that correspond to different mathematical theories.

- *Imprecision* refers to the level of variation associated with a set of measurements or to a lack of quality precision. Imprecision is usually described through probability and statistical theory; for instance, the confidence region model [118]. This is the uncertainty aspect considered in the thesis.
- *Ambiguity* can be associated with a form of lack of clarity, which implies one or more meaning, or the existence of one or more relationships. For example, ambiguity can make difficult to determine which thematic class an object contained into a satellite image belongs to. Ambiguity can be quantified by discordant measures, confusion measures, and non-specificity measures. A measurement

of information was introduced by Hartley [54], known as *Hartley's measure*, to described ambiguity in a crisp set. Moreover, Shannon's entropy [115] measures uncertainty in information theory, based on the probability theory.

- *Vagueness* refers to a lack of clarity in meaning and is normally associated with a difficult of making a sharp or precise distinction in relation to an object in the real world. Vagueness differs from ambiguity, because in the second one there are some specific and distinct interpretations that are all permitted, whereas with a vague information it is difficult to make any interpretation. Fuzzy set theory [72] is usually adopted to describe vagueness. In particular, fuzzy topology theory is used to model uncertain topological relationships between spatial objects [126]. Some of this models will be briefly described in Sec. 2.6.

In many cases, error is used a synonym for uncertainty. However, even if uncertainty can be caused by mistakes, it can also be caused by incomplete information. More specifically, the uncertainty in spatial data can result from the following major factors [117]:

- Inherent uncertainty in the real world – The natural world is complex, huge, and nonlinear; this implies that the described spatial entities usually possess a certain level of uncertainty. For example, spatial entities may not have determined, distinct, or easily identifiable boundaries. These boundaries are frequently a transition zone with a gradual change in class types, such as forest and grassland. Furthermore, as regards to scale dimension, the measurement of identical objects in the real world, using the same measurement technology but at a different level, may produce very different results.
- The limitation of human knowledge in cognition of the real world – Spatial data represented into a GIS is normally the result of an abstraction and approximation of the spatial features in the real world, not a full and complete description. Therefore, representations in a spatial database are less detailed than the whole objects in the real world.
- The limitation of measurement technologies for obtaining spatial data – The accuracy level of spatial data is increasing, but it is restricted by the limitation of measurement technologies.
- The potential generation and propagation of uncertainty in spatial data processing and analysis – Errors generated from spatial analysis operations will significantly affect the quality of the resulting datasets. This leads to the development of methods for quantifying such error propagation. Essentially, error propagation depends upon (1) errors in source datasets, (2) the applied spatial analysis operations, and (3) the presentation methods used for rendering the result. Among all the spatial processing and analysis operations, the one that mostly increases the uncertainty of spatial data, is the integration of datasets coming from different sources and characterized by different accuracies.

This thesis focuses on both modeling uncertainty in spatial data derived from the first three aspects, and managing the propagation of uncertainty during spatial data integration. In particular, a framework is presented for managing such situation and generating accuracy information for the integrated result.

Spatial Data Acquisition Methods

Spatial data capture methods are typically divided into two categories [117]: (1) direct method, which includes total station, distance measurement, GPS and laser scanning, and (2) indirect method, which includes interpolated data from models, remote sensing data, digitization of existing graphics, and direct digitization from photogrammetric instruments.

The *total station* is currently the most widely used instrument for ground surveying. It has the ability to measure both angle and distance by integrated optical and electronic technologies. The measurement accuracies are affected by both instruments characteristics, operator skills, measurement schemes, and instrumental and weather condition during measurement.

The *global positioning system* (GPS) is a highly accurate positioning technology based on pseudo range measurements between the ground receiver and the tracked satellites. Factors affecting GPS accuracy may includes: the atmosphere, which can cause signal delay, the multi-path effect, when a signal from a satellite reaches the antenna over more than one path due to different surface reflection, the number of available GPS satellites, GPS receiver clock errors, environmental disturbance, and human factors. GPS positioning accuracy is around 10 to 20 meters, with a standalone GPS positioning mode that uses pseudo-range measurement. A differential GPS (DGPS) mode uses a reference station to reduce errors in GPS measurements and the positional accuracy can reach a level of meters. A positioning accuracy of centimeters can be achieved using a GPS Real-Time Kinematic (RTK) system, which is a special GPS positioning technique that applies a carrier phase.

Laser scanning is a new technology for capturing three-dimensional spatial data. The principle of capturing ground surface data with a laser scanner involves the estimation of the distance from a laser source to a target point by recording the echo time of the emitted laser beams. The beams are then converted into the three-dimensional coordinates (x, y, z) for the target point. During this laser transmission process, the reflective intensity (I) of the target is also detected and recorded. Laser scanning technologies are usually classified into two categories: terrestrial laser scanner and airbone laser scanner. The accuracy of a point scanned by a terrestrial laser scanner can reach an accuracy that varies from millimeters to centimeters; while the accuracy of a point scanned by an airbone laser scanner reaches decimeters.

Remote sensing refers to an image technology that captures images of the Earth or other planet surfaces by using satellite sensors. In particular, thematic information can be classified or extracted from these captured images either manually or automatically. The accuracy of the captured data is subject to the spatial resolution of the satellite images, the atmosphere conditions and the instability of the remote sensors and platforms. The spatial resolution of a civilian remotely sensed image is continuously improved: from 60m for Landsat, 30m for TM, 10m for SPOT, 5m for SPOT-5, 1m for IKONOS, 0.67m for QuickBird, and up to 0.5m for WorldWiew-1. More satellite images can be produced based on the original collected ones; however, their accuracies are affected also by the quality of the mathematical models, the accuracy of the control points, and the reliability of the classification methods.

Aerial photogrammetry is a mapping technology based on stereo aerial photographs. The technology is appropriate for mapping large areas and comprises three consecutive stages: analog photogrammetry, analytical photogrammetry, and digital photogrammetry. The input of photogrammetry mapping includes stereo aerial photographs with a certain level of overlap and group control points of the area to be mapped. The outcomes are digital topographic maps, digital orthophotos, and digital elevation models. The accuracy of the produced data depends on the accuracy and scale of the aerial photographs, accuracy of the ground control points, reliability of the algorithms used for the topographic mapping, and human factors. At present, the accuracy reaches a decimeter level.

Map digitization is a common method from capturing digital spatial data from maps. Three map digitization methods are used: manual digitization from paper maps, on-screen digitization for scanned raster maps, and automated raster to vector conversion of scanned maps. The accuracy of digitized maps is related to many factors: quality of the used raw paper map, the density and complexity of the spatial features, the skill and the operational methodology of the operator, the instrument quality, the data processing software functions used to handling errors, and the accuracy of control points used for coordinate transformation.

Quality of Spatial Data

The quality of spatial data is usually described by six components: accuracy, precision, resolution, reliability, logical consistency, and completeness [64].

Accuracy usually refers to the degree to which a measurement is free from bias. It is the extent to which an estimated value approaches its true value or the reference value. In [66] accuracy is defined as the closeness of the agreement found between test results and acceptable reference values, where the test results can be either measurements or observations. Two kinds of accuracy are recognized in literature: positional and attribute accuracy.

Positional accuracy refers to the accuracy of the location of a spatial objects. Positional accuracy can be affected by three categories of errors: random error, systematic error, and gross error. *Random errors* are due to occasional factors affecting the measurement instruments, such as a change in the measurement environment, and are characterized by irregularity in magnitude and sign. *Systematic errors* may be due to functional errors in measurement equipment, they can be noticed when the magnitude and sign of the error follow a regular pattern. Finally, *gross errors* are mistakes made during measurement or data processing. For instance, they can be caused by the observer misidentifying the target to be measured, or the introduction of human error in the computational process.

Different objects in the same spatial data can be characterized by different positional accuracy, generating a so called *multi-accuracy spatial database*. In particular, positional accuracy can be classified as: *absolute accuracy*, which refers to closeness of reported coordinate values to the reference values or the value accepted as true, and *relative accuracy*, which refers to the closeness of one position measurement to the position of other measurements.

Attribute accuracy regards instead the value of a thematic or descriptive attribute. It is determined by the closeness of its current value to its actual or reference value.

Precision parameter is usually adopted in presence of an object-based model and it describes the ability of a measure to be consistently repeated. In statistics, it measures the dispersion of an observation around its mean, and it is normally used as a quality parameter for the measurement. On the contrary, *reliability* represents the consistency of a set of measurements. *Resolution* is the counterpart of precision in case of a field-based model. It refers to the level of detail that can be distinguished in an image.

Logical consistency refers to the degree of adherence to logical data structure rules, attributes and relationships [64]. It can be further classified as: *conceptual consistency*, adherence to rules of the conceptual schema, *domain consistency*, adherence of values to the value domains, *format consistency*, degree to which data are stored in accordance with the physical structure of the dataset, and *topological consistency*, correctness of the explicitly encoded topological characteristics of a dataset. In this thesis the term logical observations is used for denoting spatial relations.

Finally, *completeness* denotes the degree to which the entity objects of a dataset cover all the entity instances of the abstract universe. In other worlds, it refers to the presence or absence of features, attributes or relationships in comparison with those contained in the data model or in the natural world.

2.5 Representation of Uncertain Spatial Data

The need to consider the uncertainty of spatial data is widely recognized. In [17, 78, 93] Bhanu et al. propose a probability-based method for modeling and indexing uncertain spatial data. In this model each object is represented by a probability density function and the authors discuss how to perform spatial database operations in presence of uncertainty. In particular, in [93] they present a method for performing the probabilistic spatial join operation, which, given two uncertain datasets, finds all pairs of polygons whose probability to overlap is larger than a given threshold. In [17, 78] Bhanu et al. present a different indexing structure, called Optimized Gaussian Mixture Hierarchy (OGMH) that supports both uncertain/certain queries on uncertain/certain data, in particular they consider the k nearest neighbors (kNN) search operation. The proposed model allows the representation of multi-accuracy spatial databases, because the uncertainty of an object is described by associating to each vertex of its extent a probability density function. Therefore, an object can be intended as a d-dimensional random variable (i.e. a vector of d random variables) and the similarity between two objects is given by the probability that the two corresponding random variables are the same.

Another model for representing uncertainty in spatial database is introduced by Tossebro et al. in [128–132]. In [129] the authors propose a representation of spatial data through uncertain points, uncertain lines and uncertain regions. The basic idea is that all uncertain objects, regardless of their type, are known to be within a particular crisp region, it may also be known where an object is most likely to be. Therefore, they define the concepts of *core* and *support*: each object is represented by two regions, one inside the other: the innermost region is the area in which the object is certain to be, it is called core and it is the area where the probability

of finding the object is 1; the outermost region is the area in which the object may be, it is called support and in this area the probability of finding the object is above zero. Moreover, it is known that the object is not outside the outermost region. In [131] this model is refined in order to reduce the required storage space and to simplify the computation of the core and support regions. In [130] the authors extend their model with some constructs for representing also temporal uncertainty into a spatial database. Finally, in [132] the model is completed with the representation of topological relationships between uncertain spatial objects, since they cannot be directly inferred from the object representations.

2.6 Representation of Uncertain Topological Relations

As anticipated in Sec. 2.4 fuzzy theory is frequently used for representing topological relations among imprecise or vague objects. In order to deal with indeterminate boundaries, in [26] Clementini and Di Felice define a *region with a broad boundary* A by using two simple regions A_1 and A_2, such that $A_1 \subseteq A_2$. The boundary ∂A_1 is called *inner boundary* of A, while ∂A_2 represents the *outer boundary* of A. The broad boundary ΔA of the region A is the closed subset of $\mathbb{R}^2$ comprised between the inner boundary and the outer boundary. Given these definitions of boundary, the topological relation between two regions is defined by extending the 9-intersection model as follows:

$$\begin{bmatrix} A^\circ \cap B^\circ & A^\circ \cap \Delta B & A^\circ \cap B^- \\ \Delta A \cap B^\circ & \Delta A \cap \Delta B & \Delta A \cap B^- \\ A^- \cap B^\circ & A^- \cap \Delta B & A^- \cap B^- \end{bmatrix}$$

Based on the empty and nonempty value, this algebraic model provides a total of 44 relations between two spatial regions with a broad boundary.

Another model for dealing with nonexact spatial objects is due to Cohn and Gotts. In [28] the authors propose a model, called *egg-yolk model*, based on the use of two concentric subregions, which indicate the degree of membership in a vague or fuzzy region. In particular, the *egg* represents the precise part, and the *yolk* represents the vague (fuzzy) part. The egg-yolk model is an extension of the region connection calculus theory to fuzzy regions. Using this model, 46 relations between vague regions can be identified. The two models presented in [26] and [28] have been defined independently and mostly simultaneously. However, while [28] is based on a previous logical formulation of the authors about spatial representation and reasoning, the work in [26] is based on the point-set theoretical approach inspired by Egenhofer. When 2D regions are considered, the 44 relations of [26] coincide with the 46 relations of [28], the two additional relations are due do the fact that the first model does not further distinguishes two cases. The most relevant difference between the two models is the formal definition of the notion "one region is a crisper version of another one", which is contained in [28], but not in [26].

In [126] the authors extend the notion of fuzzy topological relation by considering the relation existing between two fuzzy regions in a fuzzy topological space, instead of in an ordinary space. This extension to a fuzzy topological space determine the existence of more topological parts. In particular, a fuzzy region A can be decomposed into the following parts:

- the core $A^{\oplus}$, which is the fuzzy interior part with a value equal to one,
- the c-boundary $\partial^c A$, representing the fuzzy subset of ∂A, s.t. $\partial A(x) = \bar{A}(x)$,
- the b-closure $A^{\perp}$, representing the fuzzy subset of $\bar{A}$, s.t. $\bar{A}(x) > \partial A(x)$, and
- the outer $A^{=}$, namely the fuzzy complement of A with value equal to one.

Given such decomposition, a 9-intersection matrix and a 4×4 intersection matrix are formalized. The 9-intersection matrix becomes:

$$\begin{bmatrix} A^{\oplus} \wedge B^{\oplus} & A^{\oplus} \wedge \ell B & A^{\oplus} \wedge B^{=} \\ \ell A \wedge B^{\oplus} & \ell A \wedge \ell B & \ell A \wedge B^{=} \\ A^{=} \wedge B^{\oplus} & A^{=} \wedge \ell B & \ell A \wedge \ell B \end{bmatrix}$$

where ℓA is called *fridge* and is defined as the union of the c-boundary and the b-closure. This intersection matrix can be used when the boundary cannot be distinguished in detail. Conversely, when the separation between the boundary of the interior and the boundary of the closure of the fuzzy region cannot be hold, the following 4×4 intersection matrix has to be used. It can be obtained by not combining the c-boundary with the b-closure:

$$\begin{bmatrix} A^{\oplus} \wedge B^{\oplus} & A^{\oplus} \wedge B^{\perp} & A^{\oplus} \wedge \partial^c B & A^{\oplus} \wedge B^{=} \\ A^{\perp} \wedge B^{\oplus} & A^{\perp} \wedge B^{\perp} & A^{\perp} \wedge \partial^c B & A^{\perp} \wedge B^{=} \\ \partial^c A \wedge B^{\oplus} & \partial^c A \wedge B^{\perp} & \partial^c A \wedge \partial^c B & \partial^c A \wedge B^{=} \\ A^{=} \wedge B^{\oplus} & A^{=} \wedge B^{\perp} & A^{=} \wedge \partial^c B & A^{=} \wedge B^{=} \end{bmatrix}$$

The main contribution of this work is the extension to a topological space, that makes the approach applicable for the identification of topological relation between fuzzy sets and crisp sets.

2.7 Integration of Uncertain Spatial Data

In Sec. 2.3 some works concerning the integration of spatial data coming from different sources have been briefly presented. In particular, conflation techniques have been cited as methods for matching corresponding geometries between two datasets. However, they typically align the dataset with lower accuracy to the more accurate one, called target dataset, without any consideration about accuracy. The positional information related to the control points within the less accurate dataset is ignored, assuming that the target one is correct. In this way, corresponding features in the two datasets are aligned but in a sub-optimal manner. Moreover, no updated quality information are provided for the adjusted dataset.

In [22, 50, 92] the authors introduce the concept of measurement-based GIS as an alternative to the usual notion of coordinate-based GIS. While in the latter systems the stored coordinate values are the primary source of data and they provide answer to both metric and topological queries; in the proposed systems, measures between higher-quality points (i.e. control points), target object boundary measurements and measurements of other objects of interest are stored together with their accuracy information. This solution provides some advantages during the integration process, because any new measure can be easily added to the database, since old or inaccurate measurements can coexists with better ones or deleted

without difficulty. However, any time a query has to be answered or the spatial information has to be visualized, the coordinates of each point have to be derived. In order to overcome this problem, in [22] the authors propose to store also the obtained coordinates and to periodically process the available measures in order to make coordinates reliable and consistent. As stated in the introduction, this thesis considers the more usual case where measurements are not available and only coordinates are stored.

A more sophisticated approach to the integration problem has to consider the accuracies of both source datasets in order to produce a more accurate integrated database, as done in [49,58–60]. These approaches use techniques based on weighted least-squares method to obtain the best fit between the source datasets. The advantage of such approaches is that resultant positions are determined taking into account all the available information, including the positional accuracy of the source positions. Moreover, updated quality parameters are generated, enabling detailed quality reporting for the resultant dataset. The integration method proposed in this thesis is also based on a least-squares estimation of the new coordinates, but it exploits the Kalman filter to perform an incremental computation of such estimation, namely the integration has not to be performed at once and there is no need to maintain all the previously integrated information for obtaining the final result. In [58–60] the authors consider also the problem of preserving topological relations between objects by representing them as inequalities that are included in the least-squares method. This thesis proposes a different approach for preserving topological relations during the integration process, similarities and differences between the two approaches will be discussed in Sec. 4.4.

In [124] the authors discuss how to use the Kalman filter into a static context for sequentially improving the best least-squares estimate as soon as new observations are integrated. The key concept above the use of the Kalman filter for sequentially computing a least-square estimate, is the idea of updating the solution: the new estimate is expressed as the linear combination of the previous one and the new observations, in a recursive manner, hence it is not required to store the previously integrated observations. In [10] the author uses the Kalman filter approach to estimate the coordinate positions of atoms within a molecule. He assumes a static structure and he does not introduce any time-dependent model of change. More details about the Kalman filter and its application in a static content can be found in Sec. 4.2.

Notice that, all these solutions for updating spatial data rely on measures with known accuracy; therefore, they are not directly applicable to existing spatial databases storing only coordinates values. A method has to be defined for determining the accuracy of these coordinates from the commonly available quality information, as discussed in the following chapter. observations for denoting spatial relations.

2.8 Summary and Concluding Remarks

This chapter has summarized some basic notions that are useful for understanding the remainder of the thesis. In particular, the model proposed in Chap. 3 for

describing a multi-accuracy spatial database uses a field-based representation of geographical objects, where the spatial component of each object is described in a 2D vector-based format. This thesis concentrates on basic geometric types, such as point, line, and polygons, and leaves more complex data structures structures, like the topological ones, to a further investigation. We also abstract from some other relevant attributes of geographical objects, such as the temporal aspects, because they are not strictly relevant for the thesis purposes.

Several kind of spatial relations have been introduced in Sec. 2.2. However, in the following we concentrate on the set of topological relations defined by Clementini et al. in [25], since they are the most widely used and implemented in GIS systems. The concept of topological relation has been also extended to represent relations between two uncertain objects, as described in Sec. 2.6. These representations usually apply the fuzzy set theory and are suitable for describing situation in which the shape and extension of the objects cannot be exactly determined, as for seas or forests. This thesis will adopt a different approach for the representation of topological relations, because in this case the object uncertainty is due to measurement errors, not to the vagueness of the involved object boundaries.

Finally, as regards to the integration of spatial data, all results presented in Sec. 2.3 are considered as a prerequisite for schema matching and corresponding object identification, but a different approach is taken for the last phase regarding the geometry alignment. In particular, an approach similar to those presented in Sec. 2.7 is adopted, which is based on an application of a least-square method. Actually, the proposed integration procedure is quite different, because it allows one to perform the integration in multiple steps and eventually in a distributed way. The differences regard also the role covered by topological relations during the integration, as it will be clear in Chap. 4.

3

A Multi-Accuracy Spatial Data Model

A multi-accuracy spatial database is a database in which objects are characterized by different accuracy parameters, in the extreme case each single position in the database can have a different accuracy. This chapter introduces an abstract data model for representing a Multi-ACcuracy Spatial database, called Macs *model*.

According to this model spatial information can be classified into two major groups: *metric observations* and *logical observations*. Metric observations represent quantitative properties of spatial objects, in particular their position and extension. These observations are subject to uncertainty and have to be treated with a statistical approach in order to express their different accuracies. Logical observations describe qualitative properties of spatial objects, like spatial relations or shape characteristics. These observations represent certain information, namely they can be only known or unknown, and they are treated with a logical approach. In geographic applications the most important category of spatial relations is the set of topological ones, whose theoretical model has been summarized in Sec. 2.2.1.

This thesis assumes that metric observations and topological relations are stored inside a Macs database and they are considered jointly during the integration process. The following sections presents how this two kinds of observations are represented inside a Macs database. In particular, Sec. 3.1 illustrates how metric observations are represented inside a Macs database, while Sec. 3.2 discusses the representation of topological information. Finally, Sec. 3.3 presents two accuracy estimators for a Macs database.

3.1 Representing Metric Observations

In accordance with the ISO TC 211 international standards for geographic information [65] and the Open GeoSpatial Consortium [4] terminology, the objects inside a Macs database are called features. A *feature* represents a real geographic entity and has a fundamental property which is the geometry describing its extension, shape and position on the Earth surface. This section considers 2D datasets embedded in a Euclidean space E^2; its extension to a 3D context is straightforward.

In a Macs database each *real position* P is represented as a pair of random variables (x_p, y_p) and its accuracy information is given by the joint probability

density function: $f_p(x_p, y_p) : E^2 \rightarrow [0, 1]$. This function describes where the position P could be located, its type depends on the survey process and can vary considerably. This thesis assumes that random variables representing real positions have a Gaussian distribution, since statistically it is the distribution obtained by any experimental process. Following this approach, for each position P to be stored in the database, it should be necessary to store its $f_p(x_p, y_p)$ by means of a set of parameters that approximate such function. This set of parameters could be very large; moreover, visualizing complex probability density functions or using them in query processing could be very difficult and computationally expensive. Therefore, a synthetic description of $f_p(x_p, y_p)$ has to be defined. Considering the context of geographical applications of recent years, where very few information about spatial accuracy is available, the following representation of positions is proposed.

Definition 3.1 (Soft Absolute Position). The absolute position of a point $P = (x_p, y_p)$ with probability density function $f_p(x_p, y_p)$, is given by a position index and a dispersion index. The **position index** of P, also called *representative point* and denoted by $\underline{\mathrm{P}}$, is the point (μ_{x_p}, μ_{y_p}), where μ_{x_p} and μ_{y_p} are the averages of x_p and y_p with respect to $f_p(x_p, y_p)$. The **dispersion index** of P represents the dispersion of the probability around $\underline{\mathrm{P}}$ and is given by the variance-covariance matrix of the x_p and y_p variables.

$$C_p = \begin{bmatrix} \sigma^2_{x_p} & \sigma_{x_p y_p} \\ \sigma_{y_p x_p} & \sigma^2_{y_p} \end{bmatrix} \tag{3.1}$$

□

In many real situations, as the building of a shared SDI database, the only available metadata describing the metric quality of coordinates in each surveyed area are: an error estimate *abs_err* for absolute positions, namely the maximum granted error between real coordinates and measurements, and a validity percentage of that error *abs_fr*, which is the percentage of cases that have to satisfy this error. These metadata are included in any tender for the production of geographical datasets (e.g. in survey processes based on areal or satellite photos) and are also required by recent ISO standards [67], hence we assume that they can be easily recovered. In [27] the authors illustrate how variance of coordinates can be computed from these metadata using the circular error formula; in this thesis we adopt their approach, as shown in Eq. 3.2. In particular, since there is no reason for considering different the variance of x from the variance of y, we can suppose that:

$$\sigma^2_{x_p} = \sigma^2_{y_p} = \sigma^2_p = \frac{-abs\_err^2}{2 \cdot \log(1 - abs\_fr)} \tag{3.2}$$

Given the variance of a position, the correlation between different positions can be estimated by introducing the covariance between their point coordinates. This correlation is defined in a way such that it is greater for near positions and it decreases as distance increases. Given two positions $P = (x_p, y_p)$ and $Q = (x_q, y_q)$, their variance and covariance values can be represented in a matrix, called C, as follows:

$$C = \begin{bmatrix} \sigma_p^2 & \sigma_{x_p y_p} & \sigma_{x_p x_q} & \sigma_{x_p y_q} \\ \sigma_{y_p x_p} & \sigma_p^2 & \sigma_{y_p x_q} & \sigma_{y_p y_q} \\ \sigma_{x_q x_p} & \sigma_{x_q y_p} & \sigma_q^2 & \sigma_{x_q y_q} \\ \sigma_{y_q x_p} & \sigma_{y_q y_p} & \sigma_{y_q x_q} & \sigma_q^2 \end{bmatrix}$$

This matrix could be fully populated only when all measurements collected during surveys are known, as done in [22]. This is not the case considered in this thesis, since we suppose to know only some aggregate metadata about the metric accuracy of positions at hand. Under these conditions some hypotheses have to be introduced in order to simplify the model and reduce the number of unknown parameters in the matrix.

Definition 3.2 (Independence hypotheses). The following hypotheses can be reasonable when considering surveyed spatial data for which no detailed information about ground measurements are available:

1. The x and y coordinates of a position P can be considered mutually independent, thus their covariance can be set to zero: $\sigma_{x_p y_p} = \sigma_{x_q y_q} = 0$.
2. The correlation among point positions is assumed to take effects only between coordinates of the same axis, i.e. the x coordinate of a position P does not influence the y coordinate of any other position Q, and vice versa: $\sigma_{x_p y_q} = \sigma_{y_p x_q} = 0$
3. The correlation between the x coordinate of P and the x coordinate of Q is equal to the correlation between the y coordinate of P and the y coordinate of Q: $\sigma_{x_p x_q} = \sigma_{y_p y_q} = c_{pq}$.

Any other hypotheses lead to an inconsistent state of C or removes the propagation effect. □

Clearly, if some additional information about the accuracy of a particular position or object, or about the correlation among specific locations is known, it can be used to properly initialize the corresponding elements of the variance-covariance matrix. Anyway, the integration technique proposed in the following sections is independent from the matrix initialization.

Applying the hypotheses contained in Def. 3.2 and considering the covariance property $\sigma_{ab} = \sigma_{ba}$, the matrix C can be rewritten as follows:

$$C = \begin{bmatrix} \sigma_p^2 & 0 & c_{pq} & 0 \\ 0 & \sigma_p^2 & 0 & c_{pq} \\ c_{pq} & 0 & \sigma_q^2 & 0 \\ 0 & c_{pq} & 0 & \sigma_q^2 \end{bmatrix} \tag{3.3}$$

where c_{pq} represents the correlation between positions P and Q and is the only unknown parameter. In order to obtain an estimation of this parameter, the following approach is considered: c_{pq} represents somehow the "attraction" that P exerts on Q and vice versa; therefore, it can be computed by considering the accuracy of the relative distance among positions in a map. Indeed, this is another piece of metadata that is often available for surveyed spatial datasets, since the accuracy of the relative distance among surveyed objects is usually higher than the one derivable from the accuracy of their absolute coordinates. Now supposing

that $\sigma^2_{d_{pq}}$ is the variance of the relative distance between two positions P and Q, it can be computed using Eq. 3.2 where *abs_err* is replaced with the maximum granted error for the relative distance between absolute positions, and *abs_fr* with its percentage of validity. The value c_{pq} can be obtained as shown in the following lemma.

Lemma 3.3 (Covariance estimation). Given the variance $\sigma^2_{d_{pq}}$ of the relative distance between two positions P and Q and the variance of their coordinates σ^2_p and σ^2_q, the covariance $\sigma_{x_P x_Q} = \sigma_{y_P y_Q} = c_{pq}$ can be computed as follows:

$$c_{pq} = \frac{\sigma^2_p + \sigma^2_q - \sigma^2_{d_{pq}}}{2} \tag{3.4}$$

Proof. - Eq. 3.4 can be obtained by applying the variance propagation law to the random variable d_{pq}, representing the distance $\overline{PQ}$, and the vector of random variables $\mathbf{v} = (x_p \; y_p \; x_q \; y_q)$, representing the coordinates of the positions P and Q. More specifically, between the random variable d_{pq} and the vector of random variables $\mathbf{v}$ the following relation exists: $d_{pq} = g(\mathbf{v}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, where g is the well-known Euclidean distance between two points. Notice that g is a non-linear function, but it can be easily linearized as $d_{pq} \simeq J \cdot \mathbf{v}$, where J is the Jacobian, namely the matrix containing the partial derivatives of g with respect to each component of $\mathbf{v}$.

$$J = \left[\frac{\partial g}{x_1} \; \frac{\partial g}{y_1} \; \frac{\partial g}{x_2} \; \frac{\partial g}{y_2} \right] \tag{3.5}$$

Let C_v the variance-covariance matrix for the random variable $\mathbf{v}$ defined as in Eq. 3.3, and let C_d the variance-covariance matrix for the random variable d_{pq}. Matrix C_d is composed of a single value: the variance of the distance between positions P and Q, computed starting from the accuracy of the relative distances (i.e. the value $\sigma^2_{d_{pq}}$). Such matrix is defined as follows:

$$C_d = \left[\sigma^2_{d_{pq}} \right] = E\left[(d_{pq} - E(d_{pq})) \cdot (d_{pq} - E(d_{pq}))^T \right] \tag{3.6}$$

where $E[X]$ is the expected value of a random variable X. Given Eq. 3.6, the relation $d_{pq} \simeq J \cdot \mathbf{v}$, and the expected value properties, the following can be derived:

$$\begin{aligned} C_d &= E[(d_{pq} - E[d_{pq}]) \cdot (d_{pq} - E[d_{pq}])^T] \\ &= E[(J \cdot \mathbf{v} - J \cdot E[\mathbf{v}]) \cdot (J \cdot \mathbf{v} - J \cdot E[\mathbf{v}])^T] \\ &= E[J \cdot (\mathbf{v} - E[\mathbf{v}]) \cdot (\mathbf{v} - E[\mathbf{v}])^T - J^T] \\ &= J \cdot E[(\mathbf{v} - E[\mathbf{v}]) - (\mathbf{v} - E[\mathbf{v}])^T] \cdot J^T \\ &= J \cdot C_v \cdot J^T \end{aligned}$$

The components of the Jacobian matrix are:

$$\frac{\partial g}{\partial x_1} = \frac{1}{2} \cdot \frac{1}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}} \cdot 2(x_1 - x_2) = \frac{(x_1 - x_2)}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}}$$

$$\frac{\partial g}{\partial x_2} = \frac{1}{2} \cdot \frac{1}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}} \cdot 2(x_2 - x_1) = \frac{(x_2 - x_1)}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}}$$

$$\frac{\partial g}{\partial y_1} = \frac{1}{2} \cdot \frac{1}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}} \cdot 2(y_1 - y_2) = \frac{(y_1 - y_2)}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}}$$
$$\frac{\partial g}{\partial y_2} = \frac{1}{2} \cdot \frac{1}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}} \cdot 2(y_2 - y_1) = \frac{(y_2 - y_1)}{\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}}$$

Starting from such Jacobian components and the relation $C_d = J \cdot C_v \cdot J^T$, the value $\sigma^2_{d_{pq}}$ can be computed as follows:

$$C_d = \left[\sigma^2_{d_{pq}}\right] = J \cdot C_v \cdot J^T$$
$$= \begin{bmatrix} \frac{\partial g}{\partial x_1} \cdot \sigma_p^2 & \frac{\partial g}{\partial y_1} \cdot \sigma_p^2 & \frac{\partial g}{\partial x_1} \cdot c_{pq} & \frac{\partial g}{\partial y_1} \cdot c_{pq} \\ + & + & + & + \\ \frac{\partial g}{\partial x_2} \cdot c_{pq} & \frac{\partial g}{\partial y_2} \cdot c_{pq} & \frac{\partial g}{\partial x_2} \cdot \sigma_q^2 & \frac{\partial g}{\partial y_2} \cdot \sigma_q^2 \end{bmatrix} \cdot J^T$$
$$= \begin{bmatrix} \left(\frac{\partial g}{\partial x_1} \cdot \sigma_p^2 + \frac{\partial g}{\partial x_2} \cdot c_{pq}\right) \cdot \frac{\partial g}{\partial x_1} + \\ \left(\frac{\partial g}{\partial y_1} \cdot \sigma_p^2 + \frac{\partial g}{\partial y_2} \cdot c_{pq}\right) \cdot \frac{\partial g}{\partial y_1} + \\ \left(\frac{\partial g}{\partial x_1} \cdot c_{pq} + \frac{\partial g}{\partial x_2} \cdot \sigma_q^2\right) \cdot \frac{\partial g}{\partial x_2} + \\ \left(\frac{\partial g}{\partial y_1} \cdot c_{pq} + \frac{\partial g}{\partial y_2} \cdot \sigma_q^2\right) \cdot \frac{\partial g}{\partial y_2} \end{bmatrix}$$

The value c_{pq} can be obtained by solving the equation:

$$\sigma^2_{d_{pq}} = \left(\frac{\partial g}{\partial x_1} \cdot \sigma_p^2 + \frac{\partial g}{\partial x_2} \cdot c_{pq}\right) \cdot \frac{\partial g}{\partial x_1} + \left(\frac{\partial g}{\partial y_1} \cdot \sigma_p^2 + \frac{\partial g}{\partial y_2} \cdot c_{pq}\right) \cdot \frac{\partial g}{\partial y_1} +$$
$$\left(\frac{\partial g}{\partial x_1} \cdot c_{pq} + \frac{\partial g}{\partial x_2} \cdot \sigma_q^2\right) \cdot \frac{\partial g}{\partial x_2} + \left(\frac{\partial g}{\partial y_1} \cdot c_{pq} + \frac{\partial g}{\partial y_2} \cdot \sigma_q^2\right) \cdot \frac{\partial g}{\partial y_2} =$$
$$= \left(\frac{(x_1 - x_2) \cdot \sigma_p^2 + (x_2 - x_1) \cdot c_{pq}}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}\right) \cdot \frac{(x_1 - x_2)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} +$$
$$\left(\frac{(y_1 - y_2) \cdot \sigma_p^2 + (y_2 - y_1) \cdot c_{pq}}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}\right) \cdot \frac{(y_1 - y_2)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} +$$
$$\left(\frac{(x_1 - x_2) \cdot c_{pq} + (x_2 - x_1) \cdot \sigma_q^2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}\right) \cdot \frac{(x_2 - x_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} +$$
$$\left(\frac{(y_1 - y_2) \cdot c_{pq} + (y_2 - y_1) \cdot \sigma_q^2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}\right) \cdot \frac{(y_2 - y_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}} =$$
$$= \frac{(x_1 - x_2)^2 \cdot \sigma_p^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \frac{(x_2 - x_1)(x_1 - x_2) \cdot c_{pq}}{(x_1 - x_2)^2 + (y_1 - y_2)^2} +$$
$$\frac{(y_1 - y_2)^2 \cdot \sigma_p^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \frac{(y_2 - y_1)(y_1 - y_2) \cdot c_{pq}}{(x_1 - x_2)^2 + (y_1 - y_2)^2} +$$

$$
\begin{aligned}
&\frac{(x_1 - x_2)(x_2 - x_1) \cdot c_{pq}}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \frac{(x_2 - x_1)^2 \cdot \sigma_q^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \\
&\frac{(y_1 - y_2)(y_2 - y_1) \cdot c_{pq}}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \frac{(y_2 - y_1)^2 \cdot \sigma_q^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \\
&= \frac{((x_1 - x_2)^2 + (y_1 - y_2)^2) \cdot \sigma_p^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \frac{((x_1 - x_2)^2 + (y_1 - y_2)^2) \cdot \sigma_q^2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \\
&\frac{-2 \cdot ((x_1 - x_2)^2 + (y_2 - y_1)^2) \cdot c_{pq}}{(x_1 - x_2)^2 + (y_1 - y_2)^2} \\
&= \sigma_p^2 + \sigma_q^2 - 2 \cdot c_{pq}
\end{aligned}
$$

obtaining the value:

$$c_{pq} = \frac{\sigma_p^2 + \sigma_q^2 - \sigma_{d_{pq}}^2}{2}$$

□

Notice that a connection exists between the accuracy of absolute positions and the accuracy of their relative distance. For example, if two positions P and Q have an absolute accuracy corresponding to a circular error of e_p and e_q, respectively, with a validity percentage of 95%, then their relative distance will be affected at most by an error of $e_p + e_q$ in the 95% of the cases. Moreover, in the context of real spatial data integration, only positive values of covariance are acceptable in order to preserve relative distances among positions.

Observation 3.1 (Positive covariance constraint). *In order to preserve the relative distance between two positions P and Q during the integration process presented in the following chapter, the covariance value c_{pq} between P and Q has to be positive (greater than zero), namely from Eq. 3.4:*

$$\sigma_{d_{pq}}^2 < \sigma_p^2 + \sigma_q^2$$

□

In other words the variance-covariance matrix can contain only non-negative covariance elements: a positive covariance is defined between a pair of objects that influence each other, while a zero covariance is defined between a pair of independent objects.

Starting from this observation, it follows that every time a value greater than this limit is obtained for $\sigma_{d_{PQ}}^2$ using Eq. 3.2, it has to be substituted with the value $\sigma_P^2 + \sigma_Q^2$.

Lemma 3.4. The covariance matrix C defined in Eq. 3.3 is positive-definite.

Proof. – A symmetric matrix is positive defined if:

1. all diagonal elements are positive;

2. each diagonal element is greater than the sum of the absolute values of all the other entries in the corresponding row/column.

The first condition is satisfied by definition, because the variance is a non-negative value, and thanks to the initialization presented in Def. 3.2, in our case it is also different from zero. As regards to the second condition, it can be easily proved by considering the hypotheses of Def. 3.2 (in particular the second one) and the constraint in Obs. 3.1.

The reasoning illustrated above regards only two positions, but its extension to the network of all points contained in a database is straightforward. In particular, this procedure shall be applied to all possible pair of positions in the database, altogether there are $m = \frac{n \cdot (n-1)}{2}$ distinct pairs of positions, where n is the total number of positions.

Clearly, the dimension of the matrix for a real database grows rapidly and its complete storage into a database can become difficult. Given a network of n positions, it has a dimension of $n \times n$, where $n \cdot (n-1)$ is the number of covariance values, and the remaining n elements are the variance values stored in the diagonal position. We can observe that the variance-covariance matrix is symmetric; therefore, only its upper (or lower) triangle has to be stored, halving the required space. Moreover, the correlation between a position P and another position Q decreases with the distance, becoming equal to zero after a certain distance; hence, for each position the number of covariance values that are different from zero and have to be actually stored is less than $(n-1)$. Anyway, some other optimizations can be adopted, they are deeply discussed in Sec. 4.7.

Given the notion of absolute position, a geometric object in a MACS database is defined as follows.

Definition 3.5 (MACS Object or feature). A MACS object or feature $\mathtt{o}$ is defined as a tuple: $\mathtt{o} = \langle \mathtt{id}, \mathtt{cl}, \mathtt{geo} \rangle$ where:

- $\mathtt{id}$ is an integer representing the unique identifier for the object.
- $\mathtt{cl}$ is the thematic class to which the object belongs to, e.g. Building or Road.
- $\mathtt{geo}$ is the geometry of the object, that is composed of:
  (i) the set of absolute positions $\mathtt{geo.pos} = \{P_1, \ldots, P_n\}$ describing the geometry and its uncertainty,
  (ii) the type of geometry $\mathtt{geo.type} \in \{\mathit{point}, \mathit{curve}, \mathit{surface}\}$, and
  (iii) the representative geometry $\mathtt{geo.rep} = \{\mu_{x_1}, \mu_{y_1}, \ldots, \mu_{x_n}, \mu_{y_n}\}$ which is the point, polyline or polygon used during object visualisation and querying.

□

Notice that the representative geometry $\mathtt{geo.rep}$ can be derived from the representative of each absolute position in $\mathtt{geo.pos}$. In order to handle the case in which only spatial relations among objects are represented (see next section), with no geometries, the empty value for $\mathtt{geo}$ is admitted; it is denoted as $\varnothing_{geo}$ and we suppose that $\varnothing_{geo}.\mathtt{pos} = \varnothing_{geo}.\mathtt{rep} = \varnothing$ and $\varnothing_{geo}.\mathtt{type} = \mathtt{null}$. Notice that on each object geometry the following constraints hold:

- if $\mathtt{geo.type} = \mathit{point}$, then $|\mathtt{geo.pos}| = |\mathtt{geo.rep}| = 1$,

- if geo.type = *curve*, then |geo.pos| = |geo.rep| > 1,
- if geo.type = *surface*, then |geo.pos| = |geo.rep| > 2.

3.2 Representing Logical Observations

For representing geographical information, another kind of observation is necessary, namely the spatial relations existing among the dataset objects. Several types of spatial relations have been defined in literature, as described in Sec. 2.2.1. This thesis focuses on topological relations, since they have been deeply studied in literature starting from the paper of Egenhofer [38] and they are available in every current GIS product and also open source software, like the well known Java APIs called JTS Topology Suite [136]. More specifically, the considered set of mutually exclusive topological relations is $\{disjoint, touch, in, contain, overlap, cross, equal\}$ to which the relations *covered_by* and *covers* are added, since they are specializations of *in* and *contains* that require a specific treatment during the integration process. The reference set of topological relations considered here becomes:

$$R_{topo} = \{disjoint, touch, in, covered\_by, contains, covers, cross, overlap\}$$

The semantics of topological relations in R_{topo} is provided in Table 3.2: for each topological relation, the last column reports the pattern grouping all the corresponding 9-intersection matrices. Notice that as highlighted in Sec. 2.2.1, the definition of such relation depends on the geometric type of the involved objects. For example, the *touch* relation can only be applied to a pair of surfaces or a surface and a curve or a point and curve, and so on. Fig. 3.1 shows the decision tree for the set R_{topo} which demonstrates that it is a set of mutually exclusive topological relations.

In the currently available GIS systems, the topological relation existing between two objects is usually derived from their geometries. However, in a MACS database absolute positions, composing the objects geometries, are *soft data*, since they are subject to measurement errors. As a consequence, from absolute positions only *soft topological relations* can be derived, namely topological relations that are not precisely defined. Sec. 2.6 summarizes some existing models for describing uncertain topological relations, this thesis proposes a different approach that is justified by the following claim.

Claim. Topological relations can also be considered as observations useful for representing spatial information. This claim has two important consequences: (i) observed topological relations among objects of a dataset have to be stored independently from the objects geometries; (ii) observed topological relations have to be integrated with objects geometries solving possible inconsistency. □

Given this claim, observed topological relations cannot be considered data subject to measurement error, since they cannot be measured like the width of a building, but they can only be true or false. Therefore, topological relations are called *hard data*, to distinguished them from absolute positions that are *soft data*, as explained before. The lack of knowledge about the precise topological relation

existing between two objects can be represented by a disjunction of topological relations, that we know might exist between them. If no relation can be excluded, then the disjunction is composed of all relations of the considered reference set.

Relation Name	**Relation Definition**	**Geometry type (S: surface, C: curve, P: point)**	**Corresponding patterns of the 9-int. matrix**
disjoint (d)	$A \cap B = \varnothing$	S/S, C/C, S/C, C/S	$FFT - FFT - TTT$
		S/P, C/P	$FFT - FFT - TFT$
		P/S, P/C	$FFT - FFF - TTT$
		P/P	$FFT - FFF - TFT$
touch (t)	$(A^\circ \cap B^\circ = \varnothing) \wedge (A \cap B) \neq \varnothing)$	S/S	$FFT - FTT - TTT$
		C/C	$F*T - *T* - T*T$
			$F*T - T** - T*T$
			$FTT - *** - T*T$
		S/C	$FFT - T** - **T$
			$FFT - FTT - T*T$
		C/S	$FT* - F** - T*T$
			$FFT - FT* - TTT$
		S/P, C/P	$FFT - TFT - FFT$
		P/C, P/S	$FTF - FFF - TTT$
in (i)	$(A \cap B^\circ = A) \wedge (A^\circ \cap B^\circ) \neq \varnothing)$	S/S, C/C, C/S	$TFF - TFF - TTT$
		P/S, P/C	$TFF - FFF - TTT$
coveredBy (b)	$(A \cap B = A) \wedge (A^\circ \cap B^\circ) \neq \varnothing) \wedge (A \cap B^\circ \neq A)$	S/S, C/C	$TFF - TTF - TTT$
		C/S	$T*F - *TF - TTT$
contains (c)	$(A \cap B^\circ = B) \wedge (A^\circ \cap B^\circ) \neq \varnothing)$	S/S, C/C, S/C	$TTT - FFT - FFT$
		S/P, C/P	$TFT - FFT - FFT$
covers (v)	$(A \cap B = B) \wedge (A^\circ \cap B^\circ) \neq \varnothing) \wedge (A^\circ \cap B \neq B)$	S/S, C/C	$TTT - FTT - FFT$
		S/C	$T*T - FTT - FFT$
			$T*T - TFT - FFT$
			$T*T - TTT - FFT$
equal (e)	$A = B$	S/S, C/C	$TFF - FTF - FFT$
		P/P	$TFF - FFF - FFT$
cross (r)	$dim(A^\circ \cap B^\circ) = (max(dim(A^\circ), dim(B^\circ)) - 1) \wedge (A \cap B) \neq A \wedge (A \cap B) \neq B$	C/S	$TTT - *** - TTT$
		S/C	$T*T - T*T - T*T$
		C/C	$0*T - *** - T*T$
overlap (o)	$dim(A^\circ) = dim(B^\circ) = dim(A^\circ \cap B^\circ) \wedge (A \cap B) \neq A \wedge (A \cap B) \neq B$	S/S	$TTT - TTT - TTT$
		C/C	$1*T - *** - T*T$

Table 3.1. Definition of the reference set of topological relations between two objects A and B. The pattern is a string "$c_{1,1}c_{1,2}c_{1,3} - c_{2,1}c_{2,2}c_{2,3} - c_{3,1}c_{3,2}c_{3,3}$", where element $c_{i,j}$ corresponds to cell (i, j) in the 9-intersection matrix. If $c_{i,j} = *$ then this position is not relevant for defining the topological relation, $c_{i,j} = F/T$ means that the intersection is (or is not) empty, $c_{i,j} \in \{0, 1, 2\}$ means that the intersection has the specified dimension. Finally, $dim(g)$ computes the dimension of the geometry g.

Relatively to point (ii) in the claim, we will explain in the following chapter that in some cases the knowledge about a particular topological relation between two objects can justify an increment of the accuracy of the relative distance between some of their positions, in order to preserve a particular configuration.

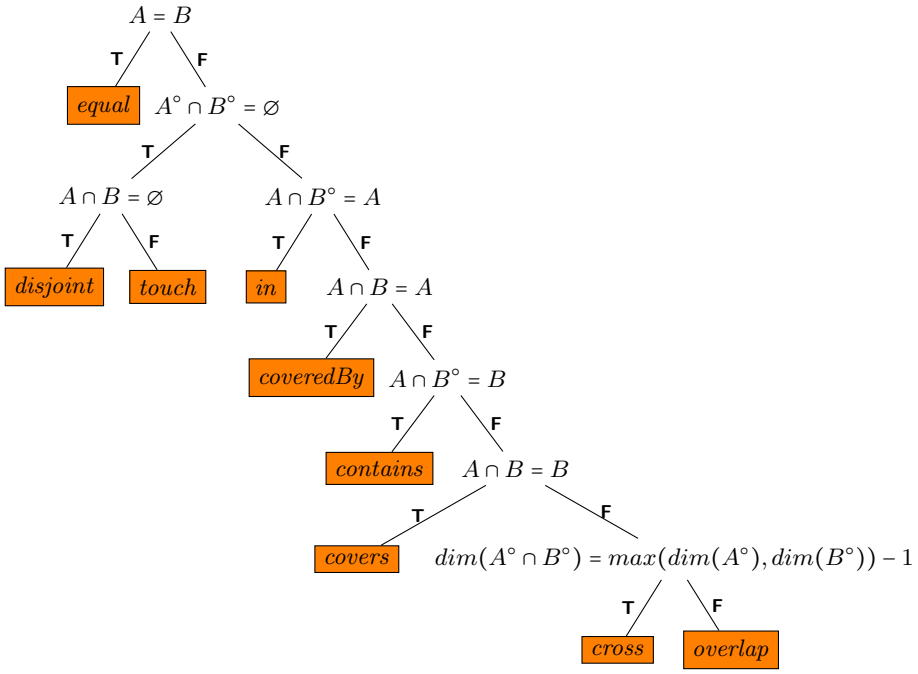


Fig. 3.1. Topological relation decision tree.

Definition 3.6 (Hard Topological Relation). Given a complete set of mutually exclusive topological relations R_{topo}, an instance of hard topological relation is the tuple

$$\langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R} \rangle$$

where:

- $\mathtt{o}_1, \mathtt{o}_2$ are objects
- $\mathtt{R} \in 2^{R_{topo}}$ is the set of topological relations that might exist between $\mathtt{O}_1$ and $\mathtt{O}_2$ (e.g. $\{Disjoint\}$, $\{In, Equal\}$, $\{Touch, In, Overlap\}$, etc.).

In particular, sets with more than one relation represent disjunction of topological relations between $\mathtt{o}_1$ and $\mathtt{o}_2$. The set containing all the topological relationships, called universal relation and denoted with R_U, represents the situation in which the topological relation between $\mathtt{o}_1$ and $\mathtt{o}_2$ is completely unknown. □

From this definition of hard topological relation, it follows that three situations may occur:

1. if $|\mathtt{R}| = 1$, the relation is known;
2. if $\mathtt{R} = R_U$, the relation is completely unknown;
3. if $|\mathtt{R}| > 1 \wedge \mathtt{R} \neq R_U$, the relation is unknown and could be one of the relations $r \in \mathtt{R}$.

In the following, where there is no ambiguity, a hard topological relation will be denoted simply as topological relation.

Even if topological relations cannot be derived from absolute positions, a *coherence constraint* has to be imposed between hard and soft topological relations.

Definition 3.7 (Soft Topological Relation). Given two objects $\mathtt{o}_1$ and $\mathtt{o}_2$ the soft topological relation r_{soft} that exists between them is the topological relation that can be computed by considering as geometries their representatives. □

For obtaining an effective integration between soft and hard data, r_{soft} has to be compatible with the hard topological relation $\mathtt{R}$ explicitly stored, as stated by the following rule.

Definition 3.8 (Hard2Soft Coherence Constraint). Given two objects $\mathtt{o}_1$ and $\mathtt{o}_2$ such that an hard topological relation $\mathtt{R}$ has been explicitly defined between them, and a soft topological relation r_{soft} can be derived from their representatives, the following constraints shall be verified: $r_{soft} \in \mathtt{R}$. □

The integration of two MACS databases can determine the violation of the coherence constraint: Sec. 4.4 will discuss in details how to solve this kind of conflicts.

Let us notice that the number of hard topological relations to be stored in a MACS database can be very large; indeed, if the database contains n objects, the total number of hard topological relations to be stored is $n(n-1)/2$, because one topological relation has to be defined between each pair of objects. This could be a large number in real databases, hence some optimizations shall be applied in order to reduce the amount of information that have to be stored. The idea is to represent hard topological relations among objects using soft topological relations when possible and store them explicitly only when they are completely or partially unknown (i.e., when $1 < |R| \leq |R_U|$). In order to apply such optimization, the notion of confidence region (or support) has to be introduced.

Definition 3.9 (Point Confidence Region). The confidence region for a position P, denoted as $\mathcal{CR}_p(\alpha)$, is the region around $\underline{P}$ within which the true location of P is contained with a probability larger than a predefined confidence level α: $\Pr\{P \in \mathcal{CR}_p(\alpha)\} > \alpha$. □

Fig. 3.2.a illustrates an example of confidence region for a position P. The orientation of the ellipse (relatively to the Cartesian coordinate system) depends on the correlation between errors in x and y directions. If errors in x and y directions are uncorrelated, as in the considered case where for any position P it holds that $\sigma_{x_p y_p} = 0$, the two semi-axes of the error ellipse are parallel to the x and y axes; therefore, the error ellipse has the shape illustrated in Fig. 3.2.b. Finally, if the error in the two directions is the same, as in the considered case where $\sigma^2_{x_p} = \sigma^2_{y_p}$ for any position P, the error ellipse becomes an error circle, as illustrated in Fig. 3.2.c. In particular, if $\alpha \simeq 0.95$, the circle radius is equal to $2\sigma^2_p$.

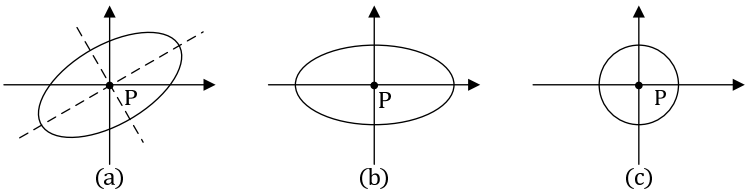


Fig. 3.2. (a) Generic confidence region for a position P, (b) confidence region for a position P when $\sigma_{xy} = 0$, (c) confidence region for a position P when $\sigma_{xy} = 0$ and $\sigma_x^2 = \sigma_y^2$.

Given the definition of confidence region for a point, several error models have been proposed in literature for describing the positional error of spatial features. As regards to line segments, this thesis considers the *error-band model* (or G-band model), originally proposed by Perkal [102] and subsequently extended by Shi [116] in order to apply it when line endpoints have different error distributions. The name of the error model suggests that the confidence region is a band around a measured location of the line, within which the true location of the line is located, with a probability larger than a predefined confidence level α.

Definition 3.10 (Line Confidence Region). The confidence region for a line segment PR, denoted as $\mathcal{CR}_{pr}(\alpha)$, is the region containing the true location of the line segment with a probability larger than a predefined confidence level. This confidence region is obtained as the union of the confidence regions of all points Q_k on the line segment, for $k \in [0,1]$. This construction ensures that the true location of all points on the line segment are contained within $\mathcal{CR}_{pr}(\alpha)$ with a probability larger than a predefined confidence level α: $\forall k \in [0,1]\ .\ \Pr\{Q_k \in \mathcal{CR}_{pr}(\alpha)\} > \alpha$. □

A line segment PR with endpoints $P = (x_p, y_p)$ and $R = (x_r, y_r)$ has an infinite set of intermediate points Q_k, for $k \in [0,1]$, that can be derived by the following formula:

$$Q_k = (1-k) \cdot P + k \cdot R = \begin{bmatrix} (1-k) \cdot \mu_{x_p} + r \cdot \mu_{x_r} \\ (1-k) \cdot \mu_{y_p} + r \cdot \mu_{y_r} \end{bmatrix} \qquad k \in [0,1] \tag{3.7}$$

The cumulative distribution function of a line segment PR can be defined by the joint distribution function of the stochastic vectors of points P, R and Q_k for all $k \in [0,1]$. However, such distribution function cannot be used to define the positional error of a line segment directly. For this purpose, the following symmetric matrix is defined between any pair of (vertex and intermediate) points Q_i, Q_j on the line segment, for any $i, j \in [0,1]$.

$$\Sigma_{st} = \begin{bmatrix} \sigma^2_{x_i x_j} & \sigma_{x_i y_j} \\ \sigma_{y_i x_j} & \sigma^2_{y_i y_j} \end{bmatrix} \tag{3.8}$$

Such matrix describes the linear correlation of the (vertex and intermediate) points on the segment. In particular, given Eq. 3.7 the elements of the matrix are given as:

$$\begin{cases} \sigma^2_{x_i x_j} = (1-i)(1-j)\cdot\sigma^2_{x_p} + [i(1-j)+j(1-i)]\cdot\sigma_{x_p x_q} + ij\cdot\sigma^2_{x_q} \\ \sigma_{x_i y_j} = (1-i)(1-j)\cdot\sigma_{x_p y_p} + i(1-j)\sigma_{x_q y_p} + j(1-i)\sigma_{x_p y_q} + ij\cdot\sigma_{x_q y_q} \\ \sigma_{y_j x_i} = (1-i)(1-j)\cdot\sigma_{y_p x_p} + j(1-i)\sigma_{y_q x_p} + i(1-j)\sigma_{y_p x_q} + ij\cdot\sigma_{y_q x_q} \\ \sigma^2_{y_i y_j} = (1-i)(1-j)\cdot\sigma^2_{y_p} + [i(1-j)+j(1-i)]\cdot\sigma_{y_p y_q} + ij\cdot\sigma^2_{y_q} \end{cases}$$

In the case considered in this thesis, the positional error in the x and y components of a same endpoint are independent and equal: $\sigma^2_{x_1} = \sigma^2_{y_1} = \sigma^2_1$, and $\sigma^2_{x_2} = \sigma^2_{y_2} = \sigma^2_2$, while $\sigma^2_1 \neq \sigma^2_2$. Therefore, the matrix in Eq. 3.8 can be simplified as:

$$\Sigma_{q_i q_j} = [(1-i)(1-j)\sigma^2_1 + ij\cdot\sigma^2_2]\cdot I_2 \tag{3.9}$$

where I_2 is the two-dimensional identity matrix. Such matrix is independent of the rotation angle, because given the following two-dimensional rotation matrix ρ:

$$\rho = \begin{bmatrix} cos\gamma & sin\gamma \\ -sin\gamma & \cos\gamma \end{bmatrix}$$

where γ is any rotation angle, the following relation holds:

$$\rho\cdot\Sigma_{q_i q_j}\cdot\rho = \Sigma_{q_i q_j}$$

Let us consider the case $i = j$, Eq. 3.9 can be simplified as:

$$\Sigma_{q_i q_i} = [(1-i)^2\sigma^2_1 + i^2\cdot\sigma^2_2]\cdot I_2 \tag{3.10}$$

This indicates that the error ellipse at any implicit point between two endpoints becomes an error circle, whose radius can vary, since the error circles of the endpoints are unequal. In order to determine the minimum radius, let $(d\Sigma_{q_i q_i})(dt)|_{t_\Delta} = 0$, it follows that $t_\Delta = (\sigma^2_1)/(\sigma^2_1+\sigma^2_2)$. When $i = t_\Delta$, the diagonal elements of the matrix $\Sigma_{q_i q_i}$ will equal to their minimum values. This corresponds to the minimum radius of the error circles.

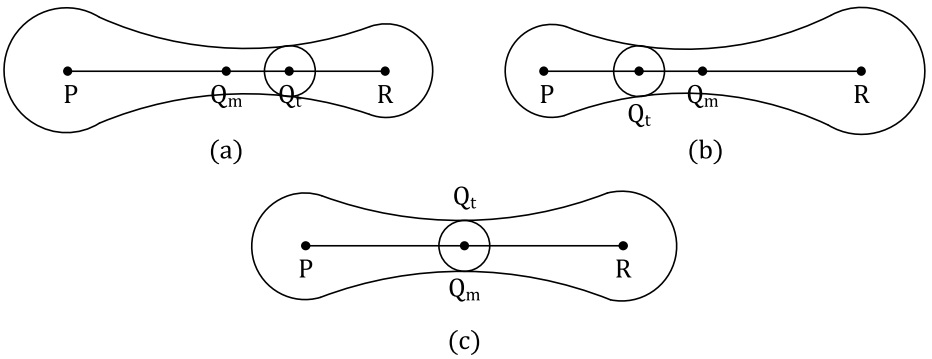


Fig. 3.3. Confidence region for a line segment when (a) $t_\Delta > 1/2$, (b) $t_\Delta < 1/2$, and (c) $t_\Delta = 1/2$.

The location of the minimum error circles on the line segment varies as follows:

- If $\sigma_1^2 > \sigma_2^2$, and thus $t_\Delta > 1/2$, then the minimum error circle on the line segment approaches to endpoint R (see Fig. 3.3.a).
- If $\sigma_1^2 < \sigma_2^2$, and thus $t_\Delta < 1/2$, then the minimum error circle on the line segment approaches to endpoint P (see Fig. 3.3.b).
- If $\sigma_1^2 = \sigma_2^2$, and thus $t_\Delta = 1/2$, then the minimum error circle on the line segment is located at the middle point of the line segment PR (see Fig. 3.3.c).

It follows that the minimum error on the line segment is always closer to the endpoint with the smallest error circle.

A polygon is obtained by linking line segments to form a closed area. Positional error of these line segments can be described using the band-error model explained above. Chun et al. propose in [24] to adopt this model also for describing the positional error of a polygon, since it is caused by the positional error of its boundary elements.

Definition 3.11 (Polygon Confidence Region). Joining several line segments forms a polyline. A special case of polyline, with identical beginning and end vertices is a closed polygon. A confidence region $\mathcal{CR}_{p_1 \ldots p_n}(\alpha)$, for the polyline $(P_1, \ldots, P_n)$, is defined as the region containing the true location of all its points with a probability larger than a predefined confidence level α: $i = 1, 2, \ldots, m-1, r \in [0,1]$. $\Pr\{P_{ir} \in J\} > \alpha$. □

Fig. 3.4 shows a square A with four edges surrounded with the corresponding error bands.

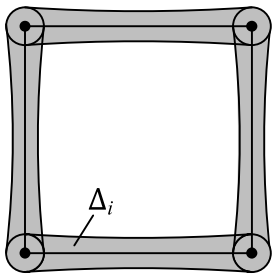


Fig. 3.4. Confidence region of a polygon computed as the union of the confidence regions of its boundary line segments.

An *index of maximum dispersion* α_m can be defined for the whole database, which has to be considered during the computation of the support of each database position. Therefore, any point outside $\mathcal{CR}_p(\alpha_m)$ cannot be considered an eligible position for P. However, the computation of the exact confidence region of all objects in a MACS database can be impracticable. The following definition explains how the confidence region of each MACS feature is estimated.

Definition 3.12 (Confidence region estimation). Given an object `o` = ⟨`id`, `cl`, `geo`⟩, its confidence region with respect to α_m, denoted as $\mathcal{CR}(\texttt{o}, \alpha_m)$, can be approximated by considering the smallest buffer region of `o.geo.rep` that contains the confidence region of all its defining positions `o.geo.pos`.

The real position of an object is located inside its confidence region estimation with a probability larger than or equal to α. □

The buffer operation is a well-known operation available in any GIS system that, given a geometry g and a distance d, computes the region representing the set of points having a distance less or equal to d from g. Fig. 3.5 illustrates an example of estimated confidence region for a line segment and a polygon.

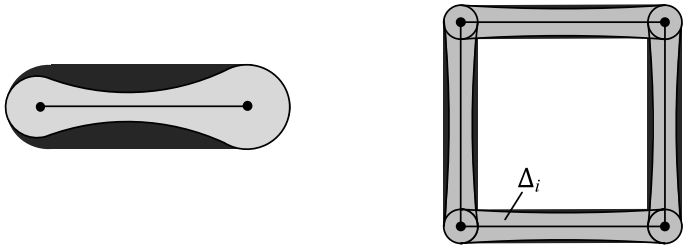


Fig. 3.5. Approximation of a line segment (a) and a polygon (b) confidence region using a buffer containing the exact confidence region.

Thanks to the notion of object confidence region, only topological relations between pairs of objects $\langle \mathtt{o}_1, \mathtt{o}_2 \rangle$ that interact (i.e. whose confidence regions are not disjoint) have to be explicitly stored. In other words, the only possible topological relation between two objects whose supports are disjoint is the disjoint one.

Besides this consideration, we can observe that in practical cases, the topological relation existing between two features is known rather than unknown, hence given the coherence constraint previously mentioned, we can decide to store only hard topological relations that contain more than one element and derive the other ones from the representatives of the objects. Therefore, given a pair of objects $\langle \mathtt{o}_1, \mathtt{o}_2 \rangle$ the possible cases are shown in Table 3.2.

Condition on objects support	Soft top. relation	Hard top. relation	Stored hard top. relation
$\mathcal{CR}(\mathtt{o}_1, \alpha_M) \cap \mathcal{CR}(\mathtt{o}_2, \alpha_M) = \varnothing$	$\mathtt{o}_1\ dj\ \mathtt{o}_2$	$\langle \mathtt{o}_1, \{dj\}, \mathtt{o}_2 \rangle$	-
$\mathcal{CR}(\mathtt{o}_1, \alpha_M) \cap \mathcal{CR}(\mathtt{o}_2, \alpha_M) \neq \varnothing$	$\mathtt{o}_1\ r\ \mathtt{o}_2$	$\langle \mathtt{o}_1, \{r\}, \mathtt{o}_2 \rangle$	-
$\mathcal{CR}(\mathtt{o}_1, \alpha_M) \cap \mathcal{CR}(\mathtt{o}_2, \alpha_M) \neq \varnothing$	$\mathtt{o}_1\ r_i\ \mathtt{o}_2$	$\langle \mathtt{o}_1, \{r_1, .., r_i, .., r_k\}, \mathtt{o}_2 \rangle$	$\langle \mathtt{o}_1, \{r_1, ..., r_k\}, \mathtt{o}_2 \rangle$
$\mathcal{CR}(\mathtt{o}_1, \alpha_M) \cap \mathcal{CR}(\mathtt{o}_2, \alpha_M) \neq \varnothing$	$\mathtt{o}_1\ r_i\ \mathtt{o}_2$	$\langle \mathtt{o}_1, R_U, \mathtt{o}_2 \rangle$	$\langle \mathtt{o}_1, R_U, \mathtt{o}_2 \rangle$

Table 3.2. Possible cases in the representation of the hard topological relations between two objects $\mathtt{o}_1$ and $\mathtt{o}_2$ (dj = disjoint).

Given the definition of soft and hard data, a MACS database can be defined as follows.

Definition 3.13 (MACS database). A Multi ACcuracy Spatial database (MACS database) is a 6-tuple: $\mathtt{macs} = \langle \mathtt{DB}, \mathtt{C}_{\mathtt{DB}}, \mathtt{TY}, \mathtt{OBJ}, \mathtt{REL}, \alpha_m, \mathcal{CR}_{\mathtt{DB}} \rangle$ where:

- $\mathtt{DB}$ is the set of position indexes of the absolute positions contained in the MACS database. For each position index $\underline{\mathrm{P}}$ the following tuple is stored: $\langle \mathtt{id}_p, x_p, y_p \rangle$, where $\mathtt{id}_p$ is the identifier of P, and $\underline{\mathrm{P}}{=}\ (x_p, y_p)$.
- $\mathtt{C}_{\mathtt{DB}}$ is the matrix of dispersion indexes (variance and covariance of coordinates) of $\mathtt{db}$; the problem of storing $\mathtt{C}_{\mathtt{DB}}$ will be discussed in the following chapter.
- $\mathtt{TY}$ is a set of available feature classes for the objects.
- $\mathtt{OBJ}$ is a set of objects $\langle \mathtt{id}, \mathtt{cl}, \mathtt{geo} \rangle$ (see Def. 3.5) belonging to the classes of $\mathtt{TY}$ and whose geometry is described through the positions in $\mathtt{DB}$: $\mathtt{cl} \in \mathtt{TY}$ and $\mathtt{geo.pos} \in \mathtt{DB}$.
- $\mathtt{REL}$ is a set of hard topological relations, which are explicitly stored, since they are not derivable from soft topological relations.
- α_m is the maximum dispersion index.
- $\mathcal{CR}_{\mathtt{DB}}$ is the region representing the support of the database, which is obtained as the union of the objects supports. □

From the optimization given above about the storing of topological relations, it follows that the complete set of topological relations that are known for a MACS database is obtained by considering the union of the hard topological relations explicitly stored, with the soft topological relations that can be derived by object geometries. Such operation is represented by the function $ext(\mathtt{REL}, \mathtt{OBJ})$.

Let us notice that if two objects inside a MACS database have intersecting geometries, then in order to exactly propagate the integration effects, they shall share some positions representing their common intersection points (for surfaces this constraint is referred to their boundary).

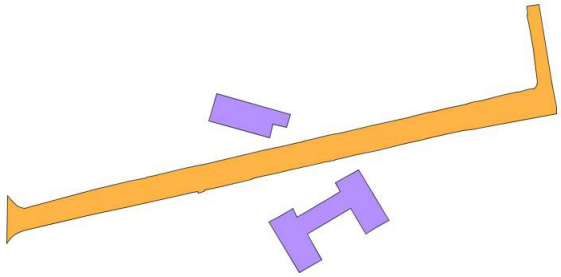


Fig. 3.6. Dataset considered in Ex. 3.14: the violet polygons represent two buildings, while the yellow polygon is a road.

Example 3.14 (Example of MACS *database).* Let us consider the database presented in Fig. 3.6, denoted here as $\mathtt{macs}_1$. Supposing that for $\mathtt{macs}_1$ the error *abs_err* for absolute positions is $0.8m$ with a percentage of validity of 95%, and the error *rel_err* for relative distances is $0.6m$ with a percentage of 95%, while its maximum dispersion index α_m has value of 0.75 and its confidence region is briefly denoted as $\mathtt{cr}$. The representation of this MACS database is reported below. Let us notice that $\mathtt{DB(id)}$ stands for the elements of the vector $\mathtt{DB}$ related to the position with identifier $\mathtt{id}$; similarly, $\mathtt{C_{DB}(id)}$ denotes the elements (variance and covariances) of the $\mathtt{C}_{\mathtt{DB}}$ matrix related to the position with identifier $\mathtt{id}$.

- $\texttt{macs}_1.\texttt{DB} = \{\langle \texttt{id}_{001}, 2456, 9783\rangle, \ldots \langle \texttt{id}_{023}, 2456, 7684\rangle, \ldots\}$
- $\texttt{macs}_1.\texttt{C}_{\texttt{DB}} = (0.25, 0.18, \texttt{cr})$
- $\texttt{macs}_1.\texttt{TY} = \{\textit{Road}, \textit{Buildings}\}$
- $\texttt{macs}_1.\texttt{OBJ} = \{\langle \texttt{obj}_1, \textit{Road}, \texttt{obj}_1.\texttt{geo}\rangle, \langle \texttt{obj}_2, \textit{Building}, \texttt{obj}_2.\texttt{geo}\rangle, \langle \texttt{obj}_3, \textit{Building}, \texttt{obj}_3.\texttt{geo}\rangle\}$
  - $\texttt{obj}_1.\texttt{geo.pos} = \{\langle \texttt{DB}(\texttt{id}_{001}), \texttt{C}_{\texttt{DB}}(\texttt{id}_{001})\rangle, \ldots, \langle \texttt{DB}(\texttt{id}_{023}), \texttt{C}_{\texttt{DB}}(\texttt{id}_{023})\rangle, \ldots\}$
  - $\texttt{obj}_1.\texttt{geo.type} = \textit{surface}$
  - $\texttt{obj}_1.\texttt{geo.rep} = \{577907, 5000248, \ldots, 577905, 5000221\}$
- $\texttt{macs}_1.\texttt{REL} = \{\langle \texttt{obj}_1, \{\textit{touch}, \textit{disjoint}\}, \texttt{obj}_2\rangle, \langle \texttt{obj}_1, \{\textit{touch}, \textit{disjoint}\}, \texttt{obj}_3\rangle\}$
- $\texttt{macs}_1.\alpha_{\texttt{m}} = 0.75$,
- $\texttt{macs}_1.\mathcal{CR}_{\texttt{db}} = \texttt{cr}$

□

As discussed in 3.1 the dimension of the variance-covariance matrix $\texttt{C}_{\texttt{DB}}$ can be very huge. The problem of efficiently representing the variance-covariance matrix will be discussed in Sec. 4.7.

3.3 MACS Database Accuracy Estimators

In order to evaluate the overall accuracy of a MACS database, an index of accuracy for metric observations and an index of certainty for logical ones are introduced. An estimation of certainty for logical observations has been chosen, instead of uncertainty, for having an index with the same behaviour of the metric accuracy.

Given a position P inside a MACS database `macs`, the metric accuracy of its absolute position is defined as the inverse of its variance. Since according to Eq. 3.3 the variance of the x and y coordinates of a position is the same and, as we will see in Sec. 4.5, remains the same also after the integration procedure, the metric accuracy of the position P is defined as:

$$acc_m(P) = \frac{1}{\sigma_P^2}$$

From this, the *average global accuracy estimator* of a MACS database `macs` concerning metric observations can be computed as:

$$acc_m(\texttt{macs}) = \frac{\sum_{P_i \in \texttt{macs.DB}} acc_m(P_i)}{|\texttt{macs.DB}|}$$

Similarly, the certainty of a set of topological relations R defined between two objects $\texttt{o}_1$ and $\texttt{o}_2$ can be estimated as:

$$acc_t(R) = \frac{(|R_U| - |R|)}{((|R_U| - 1) \cdot |R|)}$$

Considering the reference set of topological relations proposed in Sec. 3.2, we obtain: $acc_t(R) = (9 - |R|)/(8 \cdot |R|)$. Therefore, the certainty is the highest when $|R| = 1$, namely when the relation is known ($acc_t(R) = 1$), and it is the lowest when $R = R_U$, namely when the relation is unknown ($acc_t(R) = 0$).

The *average global certainty estimator* of a MACS database `macs` concerning logical observations can be computed as follows:

$$acc_t(\texttt{macs}) = \frac{|\texttt{macs.OBJ}|^2 - |\texttt{macs.REL}| + \sum_{R_i \in \texttt{macs.REL}} acc_t(R_i)}{|\texttt{macs.OBJ}|^2}$$

Each known topological relation (i.e. not explicitly stored in `macs.REL`) has a unit certainty value, hence the first term $|\texttt{macs.OBJ}|^2 - |\texttt{macs.REL}|$ computes the overall certainty of all known relations. To this value the certainty of all unknown topological relations is added ($\sum_{R_i \in \texttt{macs.REL}} acc_t(R_i)$). This sum is normalized with respect to the total number of possible relations ($|\texttt{macs.OBJ}|^2$). Therefore, the certainty is the highest when all the relations are known and decreases when more relations are unknown.

3.4 MACS Implementation Model

Given an existing spatial database, the implementation of the MACS model requires to store some additional tables that will contain the position accuracy information and the hard topological relations explicitly stored. Fig. 3.7 depicts such additional tables, notice that geometric types have been specified with respect to the OGC Simple Feature Model [8].

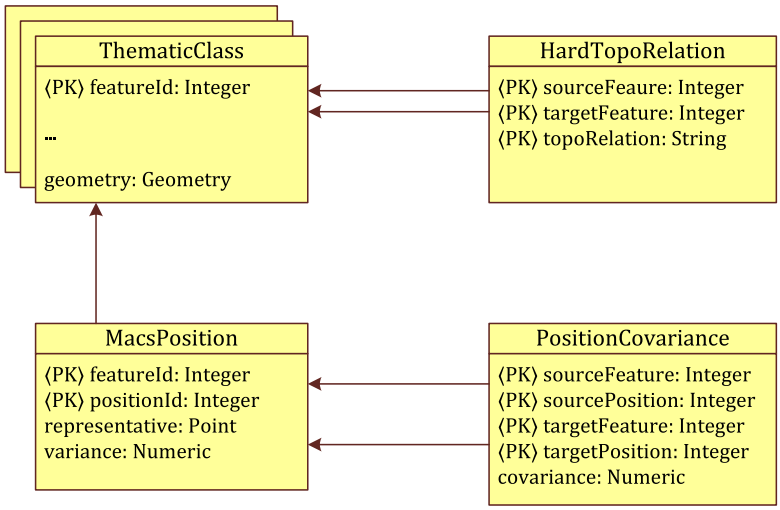


Fig. 3.7. Database schema for storing of the MACS additional information.

Existing features are supposed to be stored inside a set of tables with a name corresponding to its thematic class, as represented by the set of overlapping tables called `ThematicClass`. These tables contains a set of attributes including an identifier, which is assumed unique among all these tables, and a generic geometric attribute, which is used for querying and visualization purposes and can be specialized in each table.

Another table, called `MacsPosition`, stores the absolute positions of each feature, together with their variance. Notice that the geometry of each absolute position can be derived from the corresponding feature geometry; however, this table stores only positions that will be used during the integration process and assign to each of them a unique identifier and a variance value. We will highlight in the following chapter that in some cases only a selection of object positions has to be considered during the integration, while some other positions are discarded. Conversely, the covariance between two positions is stored into a separate table: given a source position, a covariance value is stored for any other interacting position; namely, if two positions do not interact (i.e. do not influence each other) no covariance values are stored between them.

The last presented table is called `HardTopoRelation` and is used to explicitly store the set of topological relations that may exist between two objects. Notice that the topological relation is part of the primary key, indeed between the same pair of objects different relations can be stored, which constitute a disjunction of possible relations. Similarly to the previous case, one or more tuples are stored for the same pair of objects, only if the relation cannot be subsumed for the object representatives.

3.5 Summary and Concluding Remarks

This chapters has introduced a model for representing together spatial data characterized by different positional accuracies. The model assumes a field-based representation of geographical objects and concentrates on how positional accuracy information can be stably associated to each object or even to each position representation. In particular, a statistical description of absolute positions in terms of random variables has been proposed, while accuracy information is described through a variance-covariance matrix. A method is also presented for deriving accuracy information starting from the few metadata usually available: an error for absolute positions and an error for relative distances. Such procedure is only a way for initialize the variance-covariance matrix when limited information is available, but the integration method proposed in the following chapter is independent from the starting matrix initialization. Theoretically, the proposed representation requires to define a correlation (i.e. a covariance value) between each possible pair of positions. Actually, in real situations the correlation between two positions decreases as the distance between them increases, becoming soon or later equal to zero. Therefore, the number of covariance values that have to be really stored is much less than he number of possible position pairs.

Another type of spatial information that has been considered is the set of topological relations existing between two objects. In the proposed model, topological relations cannot be derived from object representations, because the latter ones are subject to measurement errors; conversely, topological relations are considered data that can be only known or unknown. Therefore, the topological relation known between two objects has to be explicitly stored, and eventually the lack of knowledge about an exact topological relation can be modeled using a disjunction of relations. Again this approach theoretically requires to store a (set of)

topological relation(s) between any pair of objects, in reality we establish that a topological relation has to be stored between two objects only if they have an intersecting confidence region (otherwise the only possible relation is the disjoint one) and the relation cannot be derived from the relation induced by the object representatives (i.e. the topological relation is a disjunction of relations). We also propose some ways to derive the set of possible topological relations between two objects when no information is available, starting from the relation between their confidence regions.

Finally, the mathematical definition of MACS database is given together with some accuracy estimators, which can be used to evaluate its overall quality, and a description of its database implementation model. The next chapter will discuss the problem of integrating two MACS databases.

4

Integration of two MACS databases

This chapter deals with the integration of two existing MACS databases. First of all, it proposes a three steps approach that takes care of the accuracies of both source databases and produces quality information for the resulting one. In particular, the first step deals with the integration of metric observations: a statistical approach is defined that applies the Kalman filter to incrementally obtain the estimate that best fits all the available information and return also updated quality values for the result. The second step regards the integration of logical information, topological relations contained in the two source databases are combined for determining the set of relations that have to be satisfied by the resulting database. Finally, the last step treats the problem of integrating together metric and logical information, by proposing a method for solving generated inconsistencies that can be generated between soft and hard topological relations during the first two phases.

Subsequently, the proposed integration framework is extended in order to make its application feasible in a distributed context, such as an SDI, even in presence of an huge amount of data. In order to apply the proposed technique in such environment, each phase of the integration process has to be partitioned, so that it can be performed locally by each local agency. The partial computed results are then sent back to an central agency which is responsible for combining them to obtain the global result and propagating the necessary information back to the involved local agencies. Moreover, some optimization techniques are presented for reducing the amount of space that is necessary to handle accuracy information.

Finally, the proposed integration technique is validated against the real world case presented in Sec. 1.2, regarding the construction of a regional SDI in Lombardy (Italy). The aim is to analyze how the various problems highlighted in the introduction can be solved by the proposed integration process.

The structure of the chapter is the following: Sec. 4.1 discusses in details the possible scenarios in which the proposed integration framework can be applied. Sec. 4.2 deals with the statistical integration of measures in the various presented scenarios, Sec. 4.3 treats the integration of topological relations, while Sec. 4.4 explains how metric and logical observations can be combined together. Some properties of this integration process are illustrated in Sec. 4.5. The extension of the proposed technique to a distributed environment is treated in Sec. 4.6, while Sec. 4.7 presents some compression technique for reducing the amount of covariance

information that has to be transferred to and from the SDI manager. Finally, Sec. 4.8 discusses the application of the proposed technique to the motivating example presented in the introduction.

4.1 Possible Integration Scenarios

During the integration of two MACS databases different situations may occur, since the databases to be integrated can be completely different or can share absolute positions and/or objects and/or relations. More specifically, the following different application scenarios can be recognized that are further specified in Tab. 4.1:

(A) The integration of two independent databases having a comparable number of objects and positions.
  (i) The integration of two size-comparable spatial databases describing different geographic themes but sharing a large part of territory.
  (ii) The integration of two databases describing the same geographic features but on adjacent, eventually overlapping, regions.

(B) The update of a reference database $\texttt{macs}_1$ with new metric and/or logical observations represented into another database $\texttt{macs}_2$.
  (i) The integration of a massive spatial database with some new soft or hard observations about known positions or objects.
  (ii) The update of the geometries of some known objects in a reference dataset.

Notice that, in Table 4.1 some combinations are not admissible and are not shown, since the following conditions have to be satisfied:

1. $\texttt{OBJ}_1.\texttt{ID} \cap \texttt{OBJ}_2.\texttt{ID} \neq \varnothing \Rightarrow ext(\texttt{REL}_1, \texttt{OBJ}_1) \cap ext(\texttt{REL}_2, \texttt{OBJ}_2) \neq \varnothing$
2. $\texttt{OBJ}_1.\texttt{ID} \cap \texttt{OBJ}_2.\texttt{ID} \neq \varnothing \Rightarrow \texttt{TY}_1 \cap \texttt{TY}_2 \neq \varnothing$

The integration of two MACS databases results in a new MACS database. Different integration operations can be necessary depending on the particular occurred scenario. In order to classify all the situations that is necessary to handle, the general integration operations are first introduced in the following definition, then each of them will be described in details considering each possible scenario.

Definition 4.1 (MACS database integration). Given two MACS databases $\texttt{macs}_1 = \langle \texttt{DB}_1, \texttt{C}_{\texttt{DB}_1}, \texttt{TY}_1, \texttt{OBJ}_1, \texttt{REL}_1, \alpha_m, \mathcal{CR}_{\texttt{DB}_1} \rangle$ and $\texttt{macs}_2 = \langle \texttt{DB}_2, \texttt{C}_{\texttt{DB}_2}, \texttt{TY}_2, \texttt{OBJ}_2, \texttt{REL}_2, \alpha_m, \mathcal{CR}_{\texttt{DB}_2} \rangle$ their integration produces a new database $\texttt{macs}_3 = \langle \texttt{DB}_3, \texttt{C}_{\texttt{DB}_3}, \texttt{TY}_3, \texttt{OBJ}_3, \texttt{REL}_3, \alpha_m, \mathcal{CR}_{\texttt{DB}_3} \rangle$, whose components are obtained by applying different operations on the corresponding components of $\texttt{macs}_1$ and $\texttt{macs}_2$. Such operations depend on the interaction that exists between the two source databases, as reported in Table 4.1, in particular:

- $\texttt{DB}_3 = \text{MetricPosInt}(\texttt{DB}_1, \texttt{DB}_2, \texttt{C}_{\texttt{DB}_1}, \texttt{C}_{\texttt{DB}_2})$
- $\texttt{C}_{\texttt{DB}_3} = \text{MetricVarInt}(\texttt{C}_{\texttt{DB}_1}, \texttt{C}_{\texttt{DB}_2})$
- $\texttt{TY}_3 = \text{TypeInt}(\texttt{TY}_1, \texttt{TY}_2)$
- $\texttt{OBJ}_3 = \text{ObjectInt}(\texttt{OBJ}_1, \texttt{OBJ}_2)$
- $\texttt{REL}_3 = \text{LogicRelInt}(\texttt{REL}_1, \texttt{OBJ}_1, \texttt{REL}_2, \texttt{OBJ}_2)$

Integration scenario	$\langle \mathtt{TY}_\cap, \mathtt{OBJ}_\cap, \mathtt{DB}_\cap, \mathtt{REL}_\cap \rangle$	**Required operations**
A.0 - Nothing in common	$\langle \varnothing, \varnothing, \varnothing, \varnothing \rangle$	no adjustments of objects geometries
A.1 - Some classes in common, but no objects and points	$\langle \neg\varnothing, \varnothing, \varnothing, \varnothing \rangle$	no adjustments of objects geometries
A.2 - Some points in common, but no classes, objects and relations	$\langle \varnothing, \varnothing, \neg\varnothing, \varnothing \rangle$	adjustments of interfering objects geometries
A.3 - Some classes and points in common, but no objects	$\langle \neg\varnothing, \varnothing, \neg\varnothing, \varnothing \rangle$	adjustments of interfering objects geometry
A.4 - Some classes, objects and relations in common, but no points	$\langle \neg\varnothing, \neg\varnothing, \varnothing, \neg\varnothing \rangle$	objects update by geometry replacement and relation integration
A.5 - Some classes, objects, points and relations in common	$\langle \neg\varnothing, \neg\varnothing, \neg\varnothing, \neg\varnothing \rangle$	update by geometry modification and relation integration
B.1 - Some classes and points in common, but no objects ($\mathtt{OBJ}_2 = \varnothing$)	$\langle \neg\varnothing, \varnothing, \neg\varnothing, \varnothing \rangle$	adjustments of some positions
B.2 - Some classes and points in common, but no objects ($\mathtt{OBJ}_2 \neq \varnothing$)	$\langle \neg\varnothing, \varnothing, \neg\varnothing, \varnothing \rangle$	new objects insertion
B.3 - Some classes, objects and relations in common, but no points ($\mathtt{DB}_2 \neq \varnothing$)	$\langle \neg\varnothing, \mathtt{OID}_2.\mathtt{ID}, \varnothing, \neg\varnothing \rangle$	objects update by geometry replacement
B.4 - Some classes, objects and relations in common, but no points ($\mathtt{DB}_2 = \varnothing$)	$\langle \neg\varnothing, \mathtt{OID}_2.\mathtt{ID}, \varnothing, \neg\varnothing \rangle$	objects update by relations integration
B.5 - Some classes, objects, points and relations in common	$\langle \neg\varnothing, \mathtt{OID}_2.\mathtt{ID}, \neg\varnothing, \neg\varnothing \rangle$	update by geometry modification and relations integration

Table 4.1. Possible scenarios in the integration of two MACS databases. In the second column the tuple $\langle \mathtt{TY}_\cap, \mathtt{OBJ}_\cap, \mathtt{DB}_\cap, \mathtt{REL}_\cap \rangle$ represents the intersections $\langle \mathtt{TY}_1 \cap \mathtt{TY}_2, \mathtt{OBJ}_1.\mathtt{ID} \cap \mathtt{OBJ}_2.\mathtt{ID}, \mathtt{DB}_1.\mathtt{ID} \cap \mathtt{DB}_2.\mathtt{ID}, ext(\mathtt{REL}_1, \mathtt{OBJ}_1) \cap ext(\mathtt{REL}_2, \mathtt{OBJ}_2) \rangle$. Function *ext* has been defined in Sec. 3.2, it returns the set all topological relations that are valid into a MACS database, while $\varnothing$ denotes an empty intersection and $\neg\varnothing$ a not empty intersecton

- $\mathcal{CR}_{\mathtt{DB}_3} = \bigcup_{\mathtt{o} \in \mathtt{OBJ}_3} \mathcal{CR}(\mathtt{o}, \alpha_m)$ □

In order to integrate two spatial databases, the preliminary necessary operation is the identification of common classes, objects and positions. The more the databases are decoupled and come from independent sources, the more this operation is tough. As discussed in Sec. 2.3, many works are presented in literature dealing with this important issue, denoted as *schema integration* and *feature (point) matching*. This thesis supposes that the class, object and position matching operations have already been solved, since it focuses on the impact of spatial accuracy during an integration process based on object geometries. As mentioned in the introduction, we assume that the two databases are comparable in terms of schema and object instances. In particular, they should have the same level of details, and a one-to-one correspondence among objects has been previously obtained, eventually decomposing some objects in one source database. Therefore, common objects in the two source databases are supposed to share the same `ID` and the same holds also for common positions.

The simplest integration tasks are those regarding classes and objects. More specifically, the integration of classes produces simply their union:

$$\textsc{TypeInt}(\mathtt{TY}_1, \mathtt{TY}_2) = \mathtt{TY}_1 \cup \mathtt{TY}_2 \tag{4.1}$$

while the integration of the objects is obtained as follows:

$$\begin{aligned}&\textsc{ObjectInt}(\texttt{OBJ}_1, \texttt{OBJ}_2) = \\ &\quad \{o \mid (o \in \texttt{OBJ}_1 \wedge o.\texttt{ID} \notin \texttt{OBJ}_2.\texttt{ID}) \ \vee (o \in \texttt{OBJ}_2 \wedge o.\texttt{ID} \notin \texttt{OBJ}_1.\texttt{ID})\} \cup \\ &\quad \{\textsc{ObjPosInt}(o_1, o_2) \mid o_1 \in \texttt{OBJ}_1 \wedge o_2 \in \texttt{OBJ}_2 \wedge o_1.\texttt{ID} = o_2.\texttt{ID}\} \end{aligned} \tag{4.2}$$

where $\textsc{ObjPosInt}(o_1, o_2)$ is the procedure that identifies which positions have to be integrated and stored in the final database $\texttt{macs}_3$ as representatives for the object with the same identifier. In particular, this operation is necessary when corresponding objects are represented with a different number of positions; for instance, because they have been surveyed with a different scale, or because the object shape has changed over time. This choice can be done by considering the object surveying date, namely by keeping the positions of the most recent object, even its non matching positions, and discarding instead the non matching positions of the other older object. Otherwise a direct decision of the user is necessary.

The next sections are organized as follows, first the integration of the selected positions (metric observations) is considered in Sec. 4.2; in particular, a statistical method for computing the functions $\textsc{MetricPosInt}(\texttt{DB}_1, \texttt{DB}_2, \texttt{C}_{\texttt{DB}_1}, \texttt{C}_{\texttt{DB}_2})$ and $\textsc{MetricVarInt}(\texttt{C}_{\texttt{DB}_1}, \texttt{C}_{\texttt{DB}_2})$ is presented. Sec. 4.3 concentrates on the problem of integrating topological relations, it illustrates a method forcomputing the function $\textsc{LogicRelInt}(\texttt{R}_1, \texttt{R}_2)$. Finally, Sec. 4.4 treats the problem of maintaining the consistency between metric and logical observations on the integrated database.

4.2 Integrating Metric Observations

This section presents in details a method for integrating metric observations contained into two Macs databases. This method is denoted as $\textsc{MetricPosInt}_{kalman}(\texttt{DB}_1, \texttt{DB}_2, \texttt{C}_{\texttt{DB}_1}, \texttt{C}_{\texttt{DB}_2})$, where $\texttt{DB}_1$ and $\texttt{DB}_2$ are the set of position indices contained in the two databases, while $\texttt{C}_{\texttt{DB}_1}$ and $\texttt{C}_{\texttt{DB}_2}$ are the corresponding dispersion index matrices. This method is based on an application of the Kalman filter [69] to the vectors of coordinates, containing the representative of the positions that have to be integrated, and the matrices of their variance-covariance estimates.

Given two or more databases to be integrated, least squares-based methods can be used to find the solution that best fit all information contained in the source databases, considering also their accuracies. In particular, a least square-based estimation requires to solve the over-determined system of equations:

$$A \cdot \hat{x} - \ell = v$$

where:

- A is the design matrix that transforms observations into coordinates. This thesis considers only direct measurements; hence, it coincides with the identity matrix and can be safely ignored.
- x is the vector of parameters to be estimated.
- ℓ is the vector of observations.
- v is the vector of residuals.

Since the measurement observations can have different accuracies, a weight matrix W is introduced and multiplied in both equation sides. W is proportional to

observation accuracies; therefore, it is set as the inverse of the covariance matrix: $W = C^{-1}$.

$$W \cdot x - W \cdot \ell = W \cdot v$$
$$x = W^{-1} \cdot (\ell + v)$$

However, the integration of different data sources frequently cannot be performed at once, but it is a continuous and stable process that has to be executed any time new data become available or the existing one are updated.

Given an estimate x_0, let us suppose to have another set of observations ℓ_1. The question is how to obtain another estimate x_1, without considering again ℓ_0, but using only the current estimate and the new observations. Indeed, by applying again the last square estimation, we obtain an equation that still depends on ℓ_0:

$$x_1 = N_1^{-1} \cdot (W_0 \cdot \ell_0 + W_1 \cdot \ell_1) \tag{4.3}$$

where:

- $N_0 = W_0$,
- $N_1 = W_0 + W_1$,
- and in general $N_{i+1} = N_1 + W_{i+1}$.

As explained in [124], the Kalman filter can be used to update a least squares estimation as new observations are available, without requiring to store the previous integrated data. More specifically, the Kalman filter is a recursive estimator: the current state estimate is computed considering only the previous state estimate and the new available measurements. Eq. 4.3 can be rewritten in order to remove the dependency from ℓ_0, obtaining a static (not-time dependent) estimation of the Kalman filter:

$$x_1 = x_0 + N_1^{-1} \cdot W_1 \cdot (\ell_1 - x_0)$$

The matrix $K_1 = N_1^{-1} \cdot W_1 = (C_0^{-1} + C_1^{-1}) \cdot C_1^{-1}$ is called Kalman or gain matrix.

The Kalman filter has been originally designed to work with dynamic systems in which the new estimate depends on both the new available measurements and the time change. For instance, it can be applied for determining the position of a moving object at timestep $t+1$, starting from its position at timestep t, using some new observations and a model of its trajectory. In particular, given the current state estimate $\hat{x}_{t|t}$, the next state estimate $\hat{x}_{t+1|t+1}$ is determined into two steps:

1. a *predict* phase that projects forward in time the current estimate, producing *a priori* state estimate $\hat{x}_{t+1|t}$ with its corresponding a priori covariance estimate $C_{t+1|t}$, and
2. an *update* phase that corrects the a priori estimate on the basis of the new observations, producing *a posteriori* state estimate $\hat{x}_{t+1|t+1}$ with its corresponding a posteriori covariance estimate $C_{t+1|t+1}$.

In a static context, such as the integration process considered here, the state does not change due to the time passage, but only due to the availability of new observations. Therefore, the predict phase is not necessary: the a priori estimate

$\hat{x}_{t+1|t}$ coincides with the current estimate $\hat{x}_{t|t}$, and similarly the a priori covariance estimate $C_{t+1|t}$ coincides with the current one $C_{t|t}$.

Notice that, in order to effectively integrate two databases, they should share a common area; otherwise, there is no possibility to define a real correlation between them and no adjustments propagation is possible. Similarly, when a new object has to be integrated inside a pre-existing database, some information about its nearest objects has to be provided for correctly positioning it and adjusting dependent objects. Nevertheless, the proposed method is able to deal with all the cases in Table 4.1, in particular even with cases $A.0$, $A.1$ and $B.4$, where no positions are shared by the two source databases.

Algorithm 4.1 (Metric integration with no common objects or positions). Considering integration scenarios $A.0$, $A.1$ and $B.4$ of Table 4.1, in which no positions are shared between the two source Macs databases, the following integration functions can be applied:

$$\begin{aligned} &\textsc{MetricPosInt}_{union}(\mathtt{DB}_1, \mathtt{DB}_2, \mathtt{C}_{\mathtt{DB}_1}, \mathtt{C}_{\mathtt{DB}_2}) = [\mathtt{DB}_1\ \mathtt{DB}_2] \\ &\textsc{MetricVarInt}_{union}(\mathtt{C}_{\mathtt{DB}_1}, \mathtt{C}_{\mathtt{DB}_2}) = \begin{bmatrix} \mathtt{C}_{\mathtt{DB}_1} & \mathtt{C}_{\mathtt{zero}} \\ \mathtt{C}_{\mathtt{zero}}^{\mathtt{T}} & \mathtt{C}_{\mathtt{DB}_2} \end{bmatrix} \end{aligned} \tag{4.4}$$

where matrix $\mathtt{C}_{\mathtt{zero}}$ contains only zeros and $[a\ \ b]$ is the vector concatenation. □

Rationale. As regards to the vector of position indices, no positions are shared by the two source databases, hence the result vector is obtained by simply concatenating the original ones. Similarly, for the variance-covariance values, we observe that in the considered integration scenario, no information is available about the relative distance among the objects of the two source databases, consequently the covariance among their positions is set to zero. More generally, the covariance σ_{pq} between a pair of positions P and Q, where $P \in \mathtt{DB}_1 \wedge P \notin \mathtt{DB}_2$, and $Q \in \mathtt{DB}_2 \wedge Q \notin \mathtt{DB}_1$, is set to zero, as no information is available about their correlation.

In all the other cases of Tab. 4.1, an initial procedure has to be applied in order to prepare, starting from the two source databases, the position index vectors and the corresponding variance-covariance matrices that will be used by the Kalman filter. Notice that each vector (matrix) should contain position indices (variance-covariance values) regarding the whole set of objects that the resulting Macs database will contain. The obtained position index vectors are denoted as $\mathtt{V}_{\mathtt{DB}_1}$, $\mathtt{V}_{\mathtt{DB}_2}$, while the obtained covariance matrices as $\mathtt{C}'_{\mathtt{DB}_1}$ and $\mathtt{C}'_{\mathtt{DB}_2}$. They are built in different ways, according to the considered scenario presented in Table 4.1, as described by in the following algorithm.

Algorithm 4.2 (Initialization of vectors and matrices for the application of the Kalman filter). Given two sets of position indices $\mathtt{DB}_1$, $\mathtt{DB}_2$ and their corresponding dispersion indices $\mathtt{C}_{\mathtt{DB}_1}$, $\mathtt{C}_{\mathtt{DB}_2}$, the vectors $\mathtt{V}_{\mathtt{DB}_1}$, $\mathtt{V}_{\mathtt{DB}_2}$ and the variance-covariance matrices $\mathtt{C}'_{\mathtt{DB}_1}$, $\mathtt{C}'_{\mathtt{DB}_2}$ are initialized as follows:

(a) cases A.2, A.3, A.5 and B.1, B.5: the two source databases contain some common positions that have to be integrated.

(1) For each shared object, drop from each $\mathtt{DB}_i$ ($i \in \{1,2\}$) the positions that are not contained in any object geometry of $\mathtt{OBJ}_3$, as determined by the function ObjPosInt defined in Eq. 4.2.
(2) Initialize $\mathtt{V}_{\mathtt{DB}_1}$, $\mathtt{V}_{\mathtt{DB}_2}$, $\mathtt{C}'_{\mathtt{DB}_1}$ and $\mathtt{C}'_{\mathtt{DB}_2}$, as follows

$$\mathtt{V}_{\mathtt{DB}_1} = \begin{bmatrix} \mathtt{DB}_1 \setminus_{\mathtt{ID}} \mathtt{DB}_2 & \mathtt{DB}_1 \cap_{\mathtt{ID}} \mathtt{DB}_2 & \mathtt{DB}_2 \setminus_{\mathtt{ID}} \mathtt{DB}_1 \end{bmatrix}$$
$$\mathtt{V}_{\mathtt{DB}_2} = \begin{bmatrix} \mathtt{DB}_1 \setminus_{\mathtt{ID}} \mathtt{DB}_2 & \mathtt{DB}_2 \cap_{\mathtt{ID}} \mathtt{DB}_1 & \mathtt{DB}_2 \setminus_{\mathtt{ID}} \mathtt{DB}_1 \end{bmatrix}$$

where:
- $[a\ b\ c]$ represents the vector concatenation.
- $\mathtt{DB}_i \setminus_{\mathtt{ID}} \mathtt{DB}_j = \{p \mid p \in \mathtt{DB}_i \wedge p.\mathtt{ID} \notin \mathtt{DB}_j.\mathtt{ID}\}$
  It returns the set of position indices that are contained in $\mathtt{DB}_i$, but not in $\mathtt{DB}_j$.
- $\mathtt{DB}_i \cap_{\mathtt{ID}} \mathtt{DB}_j = \{p \mid p \in \mathtt{DB}_i \wedge p.\mathtt{ID} \in \mathtt{DB}_j.\mathtt{ID}\}$
  It returns for each position contained both in $\mathtt{DB}_i$ and $\mathtt{DB}_j$, the value contained in $\mathtt{DB}_i$. Notice that $\cap_{\mathtt{ID}}$ is not commutative, because it always returns the value of the position index contained in the first database.

$$\mathtt{C}'_{\mathtt{DB}_1} = \begin{bmatrix} \Pi_{1-2,1-2}(\mathtt{C}_{\mathtt{DB}_1}) & \Pi_{1-2,1\cap 2}(\mathtt{C}_{\mathtt{DB}_1}) & \mathtt{C}_{\mathtt{zero}} \\ \Pi_{1\cap 2,1-2}(\mathtt{C}_{\mathtt{DB}_1}) & \Pi_{1\cap 2,1\cap 2}(\mathtt{C}_{\mathtt{DB}_1}) & \mathtt{C}_{\mathtt{zero}} \\ \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\infty} \end{bmatrix}$$
$$\mathtt{C}'_{\mathtt{DB}_2} = \begin{bmatrix} \mathtt{C}_{\infty} & \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\mathtt{zero}} \\ \mathtt{C}_{\mathtt{zero}} & \Pi_{1\cap 2,1\cap 2}(\mathtt{C}_{\mathtt{DB}_2}) & \Pi_{1\cap 2,2-1}(\mathtt{C}_{\mathtt{DB}_2}) \\ \mathtt{C}_{\mathtt{zero}} & \Pi_{2-1,1\cap 2}(\mathtt{C}_{\mathtt{DB}_2}) & \Pi_{2-1,2-1}(\mathtt{C}_{\mathtt{DB}_2}) \end{bmatrix}$$

where:
- $\mathtt{C}_{\infty}$ is the matrix containing very high variance values on the main diagonal and zero elsewhere.
- $\mathtt{C}_{\mathtt{zero}}$ is the matrix containing only zeros.
- $\Pi_{a,b}(\mathtt{C})$ returns the sub-matrix of $\mathtt{C}$ containing only the elements $c_{i,j} \in \mathtt{C}$ where $i \in a$ and $j \in b$. In particular, the values a and b can be "$1-2$", which means the row (column) of positions $p \in \mathtt{DB}_1 \setminus_{\mathtt{ID}} \mathtt{DB}_2$, or "$1 \cap 2$", which means the row (column) of positions $p \in \mathtt{DB}_1 \cap_{\mathtt{ID}} \mathtt{DB}_2$.

(b) case A.4 and B.2, B.3: $\mathtt{DB}_2$ contains some new positions that do not exist in $\mathtt{DB}_1$ or that have to replace the corresponding positions in $\mathtt{DB}_1$. Let us suppose that $\mathtt{DB}_2$ contains also some information about the accuracy for the relative distance between its positions and some positions in $\mathtt{DB}_1$.

$$\mathtt{V}_{\mathtt{DB}_1} = \begin{bmatrix} \mathtt{DB}_1 \setminus_{\mathtt{ID}} \mathtt{DB}_2 & \mathtt{DB}_1 \cap_{\mathtt{ID}} \mathtt{DB}_2 & \mathtt{DB}_2 \setminus_{\mathtt{ID}} \mathtt{DB}_1 \end{bmatrix}$$
$$\mathtt{V}_{\mathtt{DB}_2} = \begin{bmatrix} \mathtt{DB}_1 \setminus_{\mathtt{ID}} \mathtt{DB}_2 & \mathtt{DB}_2 \cap_{\mathtt{ID}} \mathtt{DB}_1 & \mathtt{DB}_2 \setminus_{\mathtt{ID}} \mathtt{DB}_1 \end{bmatrix}$$

where $\mathtt{DB}_i \setminus_{\mathtt{ID}} \mathtt{DB}_j$ and $\mathtt{DB}_i \cap_{\mathtt{ID}} \mathtt{DB}_j$ are defined as before.

$$\mathtt{C}'_{\mathtt{DB}_1} = \begin{bmatrix} \Pi_{1-2,1-2}(\mathtt{C}_{\mathtt{DB}_1}) & \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\mathtt{zero}} \\ \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\infty} & \mathtt{C}_{\mathtt{zero}} \\ \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\mathtt{zero}} & \mathtt{C}_{\infty} \end{bmatrix}$$

$$\mathtt{C}'_{\mathtt{DB}_2} = \begin{bmatrix} \mathtt{C}_{\infty} & \Delta(\mathtt{C}_{\mathtt{zero}}) & \Delta(\mathtt{C}_{\mathtt{zero}}) \\ \Delta(\mathtt{C}_{\mathtt{zero}}) & \Pi_{1\cap 2,1\cap 2}(\mathtt{C}_{\mathtt{DB}_2}) & \Pi_{1\cap 2,2-1}(\mathtt{C}_{\mathtt{DB}_2}) \\ \Delta(\mathtt{C}_{\mathtt{zero}}) & \Pi_{2-1,1\cap 2}(\mathtt{C}_{\mathtt{DB}_2}) & \Pi_{2-1,2-1}(\mathtt{C}_{\mathtt{DB}_2}) \end{bmatrix}$$

where:

- $\mathtt{C}_{\infty}$, $\mathtt{C}_{\mathtt{zero}}$ and $\Pi_{a,b}(\mathtt{C})$ are defined as before.
- $\Delta(\mathtt{C}_{\mathtt{zero}})$ is a matrix containing the covariance between positions i and j, when known from relative distance measures, or zero otherwise. □

Rationale. The initialization algorithm is justified by the following considerations.

- Case (a) – As regards to the position index vectors, for non-shared positions only one pair of coordinates is available. However, the two databases have to be represented together, hence the original vectors have to be normalized to a common dimension. Therefore, for non-shared positions, another pair of coordinates is simulated in the other database, equal to the original one, but with very low accuracy. This information about the accuracy of the simulated measures is represented by the matrices $\mathtt{C}_{\mathtt{zero}}$ and $\mathtt{C}_{\infty}$: no correlation exists between a simulated measure and the other ones (zero covariances), while high variances (low accuracy) is established for them. Conversely, for shared objects two pairs of coordinates are available with different accuracies, and the matrices can be populated accordingly.
- Case (b) – The initialization of the position index vectors is performed as above, while the normalization of the variance-covariance matrices is quite different. First of all, $\mathtt{DB}_2$ contains some new positions that are not present in $\mathtt{DB}_1$ or that have to replace the ones contained in $\mathtt{DB}_1$. Therefore, each common position in $\mathtt{DB}_1$ has to become very inaccurate with respect to the one contained in $\mathtt{DB}_2$, hence its variance in $\mathtt{C}_{\mathtt{DB}_1}$ is replaced with a very high value (matrix $\mathtt{C}_{\infty}$) and its covariances with respect to other positions become null (matrix $\mathtt{C}_{\mathtt{zero}}$). Moreover, we assume that some information about the accuracy of relative distances between position in $\mathtt{DB}_2$ and $\mathtt{DB}_1$ may be known. This information is eventually inserted into the matrix $\mathtt{C}'_{\mathtt{DB}_2}$ and this is indicated by the use of the Δ operator.

Now, the application of the Kalman filter can be considered.

Algorithm 4.3 (Metric Integration with Kalman filter). Given the vectors $\mathtt{V}_{\mathtt{DB}_1}$, $\mathtt{V}_{\mathtt{DB}_2}$ and the matrices $\mathtt{C}'_{\mathtt{DB}_1}$, $\mathtt{C}'_{\mathtt{DB}_2}$, computed through Alg. 4.2, the Kalman filter is applied as follows:

$$\mathtt{V}_{\mathtt{DB}_3} = \mathtt{V}_{\mathtt{DB}_1} + \mathtt{K} \cdot (\mathtt{V}_{\mathtt{DB}_2} - A \cdot \mathtt{V}_{\mathtt{DB}_1})$$

$\mathtt{K}$ is named *Kalman* or *gain matrix* and it represents the adjustment applied to the measurements in $\mathtt{V}_{\mathtt{DB}_1}$ due to the presence of the measurements in $\mathtt{V}_{\mathtt{DB}_2}$. It is obtained as follows:

$$\mathtt{K} = \mathtt{C}'_{\mathtt{DB}_1} \cdot (\mathtt{C}'_{\mathtt{DB}_1} + \mathtt{C}'_{\mathtt{DB}_2})^{-1} \tag{4.5}$$

A is the *design matrix* which defines the relation between the observations and the parameters. Thesis considers only direct measurements; therefore, it can be omitted. As a consequence, the Kalman integration formula becomes:

$$\mathtt{V}_{\mathtt{DB}_3} = \mathtt{V}_{\mathtt{DB}_1} + \mathtt{K} \cdot (\mathtt{V}_{\mathtt{DB}_2} - \mathtt{V}_{\mathtt{DB}_1}) \tag{4.6}$$

□

From $\mathtt{V}_{\mathtt{DB}_3}$ it is easy to obtain $\mathtt{DB}_3$ which represents the result of the function $\textsc{MetricPosInt}_{kalman}(\mathtt{DB}_1, \mathtt{DB}_2, \mathtt{C}_{\mathtt{DB}_1}, \mathtt{C}_{\mathtt{DB}_2})$.

The filter allows not only to update the coordinates of the position indices, but also to estimate the accuracy of the resulting database, namely to update the covariance matrix as follows:

$$\mathtt{C}_{\mathtt{DB}_3} = (I - \mathtt{K}) \cdot \mathtt{C}'_{\mathtt{DB}_1} \cdot (I - \mathtt{K})^T + \mathtt{K} \cdot \mathtt{C}'_{\mathtt{DB}_2} \cdot \mathtt{K}^T \tag{4.7}$$

where I is the identity matrix and $\mathtt{C}_{\mathtt{DB}_3}$ represents the result of the function $\textsc{MetricVarInt}(\mathtt{C}_{\mathtt{DB}_1}, \mathtt{C}_{\mathtt{DB}_2})$.

The elements of the variance-covariance matrices involved into the integration process and of the Kalman matrix have the following interesting properties.

Property 4.1. Let us denote with $\mathtt{c}^1_{a,b}$, $\mathtt{c}^2_{a,b}$, $\mathtt{c}^3_{a,b}$ and $\mathtt{k}_{a,b}$ the coefficients of the matrices $\mathtt{C}_{\mathtt{DB}_1}$, $\mathtt{C}_{\mathtt{DB}_2}$, $\mathtt{C}_{\mathtt{DB}_3}$ and $\mathtt{K}$ in row a and column b, respectively. Given two positions $P = (x_p, y_p)$ and $Q = (x_q, y_q)$, the following properties hold:

- If the elements related to the variance of the x and y components of a position P are equals in each source matrix, then they also coincide in the Kalman and the integrated variance-covariance matrix.
  $\mathtt{c}^1_{x_p,x_p} = \mathtt{c}^1_{y_p,y_p} \wedge \mathtt{c}^2_{x_p,x_p} = \mathtt{c}^2_{y_p,y_p} \Rightarrow \mathtt{c}^3_{x_p,x_p} = \mathtt{c}^3_{y_p,y_p} \wedge \mathtt{k}_{x_p,x_p} = \mathtt{k}_{y_p,y_p}$

- If the elements related to the covariance between the x and y components of a position P are zero in each source matrix, then they are also zero in the Kalman and the integrated variance-covariance matrix.
  $\mathtt{c}^1_{x_P,y_P} = \mathtt{c}^1_{y_P,x_P} = 0 = \mathtt{c}^2_{x_P,y_P} = \mathtt{c}^2_{y_P,x_P} \Rightarrow$
  $\mathtt{c}^3_{x_P,y_P} = \mathtt{c}^3_{y_P,x_P} = 0 \wedge \mathtt{k}_{x_P,y_P} = \mathtt{k}_{y_P,x_P} = 0$

- If the elements related to the covariance between the x (or y) components of two positions P and Q are equals in each source matrix, then they also coincide in the Kalman and the integrated variance-covariance matrix.
  $\mathtt{c}^1_{x_p,x_q} = \mathtt{c}^1_{y_p,y_q} \wedge \mathtt{c}^2_{x_p,x_q} = \mathtt{c}^2_{y_p,y_q} \Rightarrow \mathtt{c}^3_{x_p,x_q} = \mathtt{c}^3_{y_p,y_q} \wedge \mathtt{k}_{x_p,x_q} = \mathtt{k}_{y_p,y_q}$

- If the elements related to the covariance between the x and y components of two positions P and Q are zero in each source matrix, then they are also zero in the Kalman and the integrated variance-covariance matrix.
  $\mathtt{c}^1_{x_p,y_q} = \mathtt{c}^1_{y_q,x_p} = \mathtt{c}^1_{y_p,x_q} = \mathtt{c}^1_{x_q,y_p} = 0 = \mathtt{c}^2_{x_p,y_p} = \mathtt{c}^2_{y_p,x_p} = \mathtt{c}^2_{y_p,x_q} = \mathtt{c}^2_{x_q,y_p} \Rightarrow$
  $\mathtt{c}^3_{x_p,y_q} = \mathtt{c}^3_{y_q,x_p} = \mathtt{c}^3_{y_p,x_q} = \mathtt{c}^3_{x_q,y_p} = 0 = \mathtt{k}_{x_p,y_q} = \mathtt{k}_{y_q,x_p} = \mathtt{k}_{y_p,x_q} = \mathtt{k}_{x_q,y_p}$

□

These properties confirm that the initial configuration of the variance-covariance matrix (see Eq. 3.3) is preserved by the proposed integration method. Another important property of the integration method is that the coordinate estimator defined in Eq. 4.6 is correct or unbiased, as illustrated below.

Property 4.2. The estimator $\mathtt{V}_{\mathtt{DB}_3}$ defined in Eq. 4.6 is a correct (unbiased) estimator of the parameter $\overline{\mathtt{V}}_{\mathtt{DB}_3}$, namely:

$$E[\mathtt{V}_{\mathtt{DB}_3}] = \overline{\mathtt{V}}_{\mathtt{DB}}$$

i.e. the expected value of the random variable $\mathtt{V}_{\mathtt{DB}_3}$ is equal to the parameter $\overline{\mathtt{V}}_{\mathtt{DB}}$, representing the reference or true integrated database content.

Proof. The average of the estimator $\mathtt{V}_{\mathtt{DB}_3}$ can be rewritten as:

$$E[\mathtt{V}_{\mathtt{DB}_3}] = E[\mathtt{V}_{\mathtt{DB}_1} + \mathtt{K}(\mathtt{V}_{\mathtt{DB}_2} - \mathtt{V}_{\mathtt{DB}_1})]$$

by applying to this definition the properties of the average operator, the following transformation can be performed:

$$\begin{aligned} E[\mathtt{V}_{\mathtt{DB}_3}] &= E[\mathtt{V}_{\mathtt{DB}_3} + \mathtt{K}(\mathtt{V}_{\mathtt{DB}_2} - \mathtt{V}_{\mathtt{DB}_1})] \\ &= E[\mathtt{V}_{\mathtt{DB}_3}] + E[\mathtt{K}(\mathtt{V}_{\mathtt{DB}_2} - \mathtt{V}_{\mathtt{DB}_1})] \\ &= E[\mathtt{V}_{\mathtt{DB}_1}] + \mathtt{K} \cdot E[(\mathtt{V}_{\mathtt{DB}_2} - \mathtt{V}_{\mathtt{DB}_1})] \\ &= E[\mathtt{V}_{\mathtt{DB}_1}] + \mathtt{K} \cdot (E[\mathtt{V}_{\mathtt{DB}_2}] - E[\mathtt{V}_{\mathtt{DB}_1}]) \end{aligned}$$

by considering that $E[\mathtt{V}_{\mathtt{DB}}] = E[\mathtt{V}_{\mathtt{DB}_3}] = E[\mathtt{V}_{\mathtt{DB}_1}] = E[\mathtt{V}_{\mathtt{DB}_2}]$. It follows that:

$$\begin{aligned} E[\mathtt{V}_{\mathtt{DB}_3}] &= E[\mathtt{V}_{\mathtt{DB}_1}] + \mathtt{K} \cdot \mathtt{C}_{\mathtt{zero}} \\ &= E[\mathtt{V}_{\mathtt{DB}_1}] \end{aligned}$$

From which it can be derived by generalization:

$$E[\mathtt{V}_{\mathtt{DB}_3}] = E[\mathtt{V}_{\mathtt{DB}}]$$

□

Notice that the effectiveness of a least square-based method can be compromised by the presence of blunders. A *blunder* is an erroneous observation that is clearly in contrast with the other available observations. In the integration context, blunders influence the point matching phase of the two source databases. Several blunder detection techniques have been proposed [147]; however, this thesis assumes that the quality of the considered information is ensured by the data provider, which is responsible for performing a correct point matching of the source databases, hence we can safely abstract from this problem.

It is clear that in real situations a least squares-based methods cannot be applied as is to an entire database, in particular for the costs of inverting the involved matrices. Sec. 4.6 will present a distributed version of this integration procedure that can help to overcome this kind of problems.

4.3 Integrating Logical Observations

This section discusses the problem of integrating logical observations contained into two distinct MACS databases $\texttt{macs}_1$ and $\texttt{macs}_2$. In particular, referring to Def. 4.1, we define a method for computing the function LOGICRELINT($\texttt{REL}_1, \texttt{OBJ}_1, \texttt{REL}_2, \texttt{OBJ}_2$).

As regards to the integration of logical observations, a significant case occurs when the two databases share at least one object. Anyway, the proposed method is able to handle any possible case; indeed, different operations are necessary according to the rate of objects sharing. In particular, if no objects are shared the known relations are all preserved, while the new relations between objects of $\texttt{OBJ}_1$ and objects of $\texttt{OBJ}_2$ have to be declared unknown. Actually, considering the confidence region of these objects, more precise relations can be derived by computing the relations among their confidence region, as described by the following algorithm.

Algorithm 4.4 (Computation of objects relations from confidence regions relations). Given two sets of objects O_1 and O_2, where $O_1.\texttt{ID} \cap O_2.\texttt{ID} = \varnothing$, the following function can be defined for representing the knowledge about the topological relations existing among the objects of $O_1 \cup O_2$. Such function is obtained by considering the relations between their confidence region:

$$\begin{aligned} &\textsc{TopFromSupp}(O_1, O_2) = \\ &\quad \{\langle \texttt{o}_1, \texttt{o}_2, R_x \rangle \mid (\texttt{o}_1, \texttt{o}_2) \in O_1 \times O_2 \ \wedge \ \texttt{R}_x = \textsc{SuppRel}(\texttt{o}_1, \texttt{o}_2, \alpha)\} \end{aligned} \tag{4.8}$$

where $\textsc{SuppRel}(\texttt{o}_1, \texttt{o}_2, \alpha)$ is defined as in Lis. 4.1.

Rationale. Considering Lis. 4.1 and starting from the first condition we can observe that, if the confidence regions of the two objects are disjoint, then for the confidence region definition (Def. 3.12) the objects are disjoint. Otherwise, if they have intersecting confidence regions, $\texttt{o}_1$ is a surface and the $\texttt{o}_2$ confidence region is inside $\texttt{o}_1$ without touching its boundary, then no points of $\texttt{o}_2$ can have a position that is outside $\texttt{o}_1$, hence $\texttt{o}_2$ *in* $\texttt{o}_1$. The third conditional block shows a situation that is the inverse of the previous one. Finally, the last conditional block says that, if the confidence regions of two surfaces intersect without considering the confidence region of their interior, the surfaces certainly have intersecting interiors, hance the existing relation between them can be only one among *in*, *contains*, *covers*, *covered_by*, *equal* or *overlap*.

Given the function $\textsc{TopFromSupp}(O_1, O_2)$, the integration of the topological relations contained into two MACS databases is performed.

Algorithm 4.5 (Topological relation integration). Given two distinct MACS databases $\texttt{macs}_1$ and $\texttt{macs}_2$, the integration of the set of topological relations (or logical observations) that are known in each of them is obtained trough the function LOGICRELINT($\texttt{REL}_1$, $\texttt{OBJ}_1$, $\texttt{REL}_2, \texttt{OBJ}_2$) as described below. Notice that in order to obtain this result, the complete sets of relations known by $\texttt{macs}_1$ and $\texttt{macs}_2$ have to be computed. They are denoted as $\texttt{R}_1 = ext(\texttt{REL}_1, \texttt{OBJ}_1)$ and $\texttt{R}_2 = ext(\texttt{REL}_2, \texttt{OBJ}_2)$. Starting from them, $\texttt{R}_3$ is obtained as follows (referring to Tab. 4.1 for the cases definition and to Tab. 3.2 for relation symbols):

Listing 4.1 Algorithm for computing the topological relation existing between two objects starting from the relation between their supports.

input: $\mathtt{o}_1$, $\mathtt{o}_2$, α
output: $\mathtt{R}_x$

$\mathtt{R}_x \leftarrow$ SuppRel$(\mathtt{o}_1, \mathtt{o}_2, \alpha)$

if $\mathcal{CR}(\mathtt{o}_1, \alpha)$ `disjoint` $\mathcal{CR}(\mathtt{o}_2, \alpha)$ **then**
  $\mathtt{R}_x = \{disjoint\}$
elseif $\mathtt{o}_2$.geo.type $= S \wedge$
  $\mathcal{CR}(\mathtt{o}_1, \alpha)$ `in` $\mathtt{o}_2 \wedge$
  $\mathcal{CR}(\mathtt{o}_1, \alpha)$ `disjoint` $\mathcal{CR}(\partial\mathtt{o}_2, \alpha)$ **then**
  $\mathtt{R}_x = \{in\}$
elseif $\mathtt{o}_1$.geo.type $= S \wedge$
  $\mathtt{o}_1$ `contains` $\mathcal{CR}(\mathtt{o}_2, \alpha) \wedge$
  $\mathcal{CR}(\partial\mathtt{o}_1, \alpha)$ `disjoint` $\mathcal{CR}(\mathtt{o}_2, \alpha)$ **then**
  $\mathtt{R}_x = \{contains\}$
elseif $\mathtt{o}_1$.geo.type $= S \wedge$
  $\mathtt{o}_2$.geo.type $= S \wedge$
  $\mathtt{o}_1$ `contains` $\mathcal{CR}(\mathtt{o}_2, \alpha) \wedge$
  $(\mathcal{CR}(\mathtt{o}_1, \alpha) \setminus \mathcal{CR}(\partial\mathtt{o}_1, \alpha))$ `intersects` $(\mathcal{CR}(\mathtt{o}_2, \alpha) \setminus \mathcal{CR}(\partial\mathtt{o}_2, \alpha))$ **then**
  $\mathtt{R}_x = \{in, contains, cover\_by, covers, overlap, equal, cross\}$
end if
return $\mathtt{R}_x$

- in cases $A.0$, $A.1$, $A.2$, $A.3$ and $B.1$, $B.2$ no objects are shared by the two source databases $\mathtt{macs}_1$ and $\mathtt{macs}_2$, hence:

$$\mathtt{R}_3 = \mathtt{R}_1 \cup \mathtt{R}_2 \cup \text{TopFromSupp}(\mathtt{OBJ}_1, \mathtt{OBJ}_2)$$

  where TopFromSupp$(\mathtt{OBJ}_1, \mathtt{OBJ}_2)$ has been introduced in Alg. 4.4.
- in cases $A.4$, $A.5$ and $B.3$, $B.4$, $B.5$ there are some common objects between the databases to be integrated, hence the function works differently:

$$\begin{aligned}\mathtt{R}_3 = &(\mathtt{R}_1 \setminus_{\mathtt{ID}} \mathtt{R}_2) \cup (\mathtt{R}_2 \setminus_{\mathtt{ID}} \mathtt{R}_1) \cup \\ &\text{TopFromSupp}(\mathtt{OBJ}_1 \setminus_{\mathtt{ID}} \mathtt{OBJ}_2, \mathtt{OBJ}_2 \setminus_{\mathtt{ID}} \mathtt{OBJ}_1) \cup \\ &\text{MergeTopRel}(\mathtt{R}_1, \mathtt{OBJ}_1 \cap_{\mathtt{ID}} \mathtt{OBJ}_2, \mathtt{R}_2, \mathtt{OBJ}_2 \cap_{\mathtt{ID}} \mathtt{OBJ}_1)\end{aligned}$$

  where
  - $(\mathtt{R}_i \setminus_{\mathtt{ID}} \mathtt{R}_j) = \{\langle \mathtt{o}_1, \mathtt{o}_1, \mathtt{R}_x \rangle \mid \langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R}_x \rangle \in \mathtt{R}_i \wedge \langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R}_y \rangle \notin \mathtt{R}_j\}$
    It returns the topological relations that are defined in $\mathtt{R}_i$, such that no topological relations are defined between the same pair of objects in $\mathtt{R}_j$.
  - $(\mathtt{OBJ}_i \setminus_{\mathtt{ID}} \mathtt{OBJ}_j) = \{o \mid o \in \mathtt{OBJ}_i \wedge o.\mathtt{ID} \notin \mathtt{OBJ}_j.\mathtt{ID}\}$
  - The function TopFromSupp has been defined in Alg. 4.4.
  - The function MergeTopRel$(\mathtt{R}_1, O_1, \mathtt{R}_2, O_2)$ is defined as follows:

$$\text{MergeTopRel}(\mathtt{R}_1, O_1, \mathtt{R}_2, O_2) = \tag{4.9}$$

$$\langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R} \rangle \mid \mathtt{o}_1 \in O_1 \wedge \mathtt{o}_2 \in O_2 \wedge$$

$$\langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R}_x \rangle \in \mathtt{R}_1 \wedge \langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R}_y \rangle \in \mathtt{R}_2 \wedge \mathtt{R} = \mathtt{R}_x \cap \mathtt{R}_y)\} \tag{4.10}$$

The result of LogicRelInt($\mathtt{REL}_1, \mathtt{OBJ}_1, \mathtt{REL}_2, \mathtt{OBJ}_2$) = $\mathtt{REL}_3$ is obtained by considering the entries of $\mathtt{R}_3$ that represents disjunction of relations or empty relations. □

The MergeTopRel function can produce empty relations (as result of the intersection $\mathtt{R}_x \cap \mathtt{R}_y$); these empty relations represent inconsistencies between the databases to be integrated and have to be solved by the intervention of a domain expert. Notice that in the source datasets the coherence constraint is originally satisfied and these inconsistencies can be generated only in the resulting dataset, as a consequence of the fusion operations performed separately on metric and logical observations.

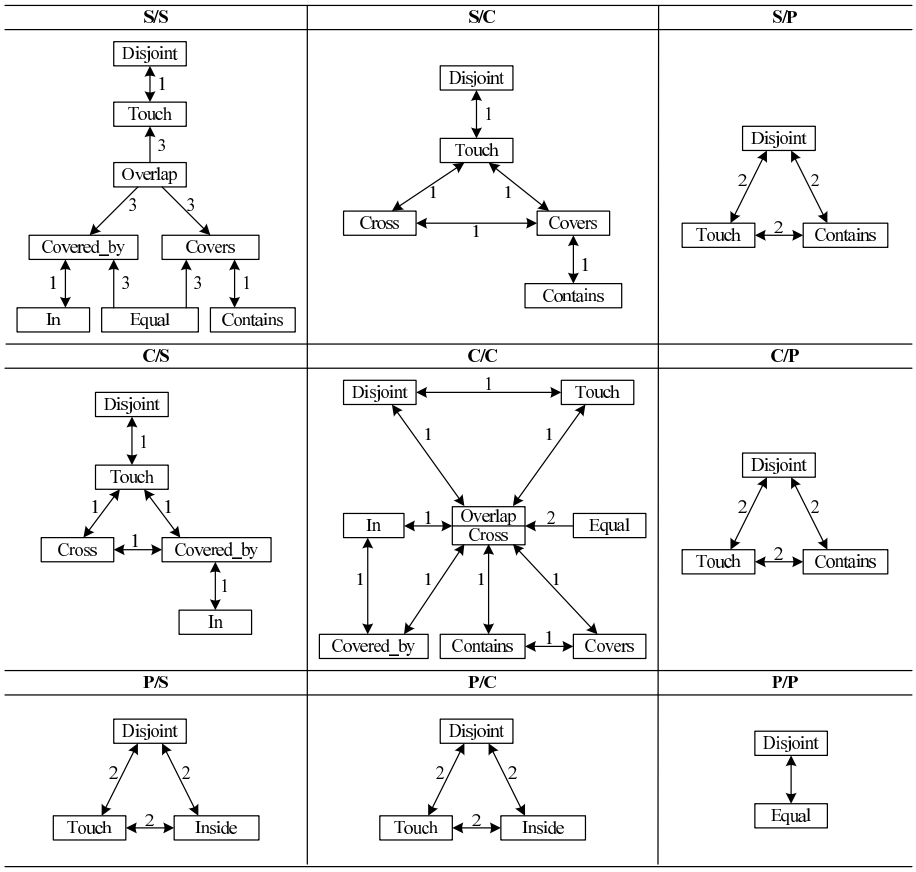


Fig. 4.1. Proximity between topological relations classified on the basis of the type of the involved objects. Let us notice that for not cluttering the representation, the relations *overlap* and *cross* between two curves have been collapsed into a unique box because they have the same distance from the other relations. The distance between this relations is 1 if the dimension of the intersection between their interior is considered.

A manual intervention of a domain expert is necessary whenever logical observations contained in the source databases are discordant. However, if the cost of

a manual intervention is too high or the user is not able to determine the right relation for the final database, some automatic procedures can be implemented in order to convert the inconsistency into a loss of certainty. In this regard, the proximity relationship among topological relations is considered. This relation has been first introduced in [40] for the definition of conceptual neighborhoods starting from the 9-intersection matrices. This definition has been extended in [12] in order to be applied to relations defined by means of sets of 9-intersection matrices, as those defined in Table 3.2. In particular, the distance between two relations is computed considering the minimum distance between the corresponding 9-intersection matrices. This thesis adopts the same approach for defining, given a topological relation r_1 between specific object types (e.g. between surfaces), the set of relations that are near to it. A topological relation r_1 is *near* to another relation r_2, if r_2 is characterized by a matrix with the minimum distance (variation), with respect to other relations, from the matrix characterizing r_1. The following definition formally specifies the proximity between topological relations. Fig. 4.1 illustrates the proximity between topological relations computed considering the involved object types. An arc is depicted between two topological relations if they are near and the label on each arc denotes the distance between them. Let us notice that when a topological relation has several matrices associated to it, each of these can have different distances with respect to the matrix of another relation, but for simplicity only the minimum distance is reported in the diagram.

Definition 4.2 (Topological relation proximity). Given two topological relations r_1 and r_2 defined between objects of type t_1 and t_2, and represented by the set of 9-intersection matrices M_1 and M_2, respectively, r_1 is said to be *near* r_2 if

$$distance(r_1, r_2) = min\{distance(r_1, r) \mid r \neq r_1 \wedge r \in R_{topo}\}$$

where the *distance* function computes the distance between topological relations as the minimum number of discordant elements between matrices $m_1 \in M_1$ and $m_2 \in M_2$. □

If this kind of approach can be acceptable for the user, it can be assumed that when the topological relations in the two source databases are not compatible but are near, then the resulting relation becomes the disjunction of the original ones. Formally, this result can be obtained by replacing the intersection $\mathtt{R}_x \cap \mathtt{R}_y$ in Eq. 4.10 with

$$near(r_1, \mathtt{o}_1.\mathtt{geo.type}, \mathtt{o}_2.\mathtt{geo.type}) \cap near(r_2, \mathtt{o}_1.\mathtt{geo.type}, \mathtt{o}_2.\mathtt{geo.type})$$

where $near(r, t_1, t_2)$ computes the set of relations that are near to r when objects of types t_1 and t_2 are considered.

4.4 Integrating Metric and Logical Observations Together

The complete integration of two MACS databases requires to combine metric and logical observations together. In particular, Sec. 3.2 has introduced the coherence constraint between soft topological relations, which are those derived from object

representatives, and hard topological relations, which are those explicitly stored. Moreover, the same section has established that for reducing the quantity of stored information, when a topological relation is known, it can be directly derived from the geometry of the object representatives without additional information.

In general, after the integration operations presented in the previous sections a check phase is necessary in order to verify that the coherence constraint is satisfied in the resulting MACS database $\texttt{macs}_3$. This means that for each pair of objects in $\texttt{OBJ}_3$ the soft topological relation between them has to be computed, denoted as r_{soft}, and compared with the relation eventually stored in $\texttt{REL}_3$, denoted as $\texttt{R}$. If $r_{soft} \in \texttt{R}$, then the coherence constraint is satisfied, otherwise it is necessary to modify the positions defining the involved objects geometry, in order to obtain a new situation where r_{soft} changes and becomes one of the relations of R. Indeed, we always suppose that logical observations have higher priority with respect to metric observations, because they are not subject to uncertainty.

The remainder of this section analyzes how metric observations compliant with a topological relation r_1 have to be transformed in order to become compliant with another desired topological relation r_2. In doing so, let us notice that some transitions from one topological relation to another, like the transition *disjoint* $\rightarrow$ *touch*, require that two distinct positions of the objects become the same position. This case is denoted with the term *position snapping* ($\rightarrow\leftarrow$). For other transitions the inverse operation is required, i.e. a shared position has to be transformed into two distinct ones. This operation is denoted as *position decoupling* ($\leftarrow\rightarrow$). Finally, in some cases the switch of a position location with respect to a curve or surface is necessary. This operation is denoted as *position switching* ($\rightleftarrows$).

The following algorithm summarizes the main steps of a position snapping operation together with the necessary preconditions to their application.

Algorithm 4.6 (Position Snapping). Given two distinct MACS features a and b, the *position snapping operation* ($\rightarrow\leftarrow$) produces a new geometry for them such that at least one position is shared. It can be performed between two features if and only if there is a position in one of them whose confidence region intersects the confidence region of the other feature:

$$\begin{aligned}(\exists P \;.\; P \in a.\texttt{geo.pos} \wedge \mathcal{CR}_p(\alpha) \cap \mathcal{CR}(b,\alpha) = \varnothing) \vee \\ (\exists Q \;.\; Q \in b.\texttt{geo.pos} \wedge \mathcal{CR}_q(\alpha) \cap \mathcal{CR}(a,\alpha) = \varnothing)\end{aligned}$$

Given such property, the set S containing the pairs of positions to snap is built as follows:

1. For each position $P_1 \in a.\texttt{geo.pos}$ such that $\mathcal{CR}_{p_1}(\alpha) \cap \mathcal{CR}(b,\alpha) \neq \varnothing$, find the corresponding position (i.e. position at minimum distance) $P_2 \in b$ such that it is an existing position of b or it is generated by projection. Add the pair (P_1, P_2) to the set S.
2. For each position $P_2 \in b.\texttt{geo.pos}$ such that $\mathcal{CR}_{p_2}(\alpha) \cap \mathcal{CR}(a,\alpha) \neq \varnothing$, find the corresponding position (i.e. position at minimum distance) $P_1 \in a$ such that it is an existing position of a or it is generated by projection. Add the pair (P_1, P_2) to the set S.
3. For each pair $(P_1, P_2) \in S$:

a) Substitute P_2 with P_1 in b. The two objects now share a common position.
b) Consider P_2 as a new observation for P_1 to be integrated.

Some alternatives of the general position snapping algorithm presented in Alg. 4.6 can be necessary to treat particular situations. They are explained in the following algorithm, while Fig. 4.2 provides some example of their application.

Algorithm 4.7 (Position Snapping Variants). Given two MACS features a and b, the following alternatives to the general position snapping algorithm in Alg. 4.6 have been defined:

- *One position snapping* ($a \rightarrow\!\leftarrow_1 b$): this operation has the same precondition and behaviour of the general position snapping, but it requires to snap only one position of b. Therefore, in point 3 not all positions are snapped but only one pair is chosen. This choice can be random or taken by a human agent.
- *Two position snapping* ($a \rightarrow\!\leftarrow_2 b$): this operation is similar to the previous one, but requires to snap exactly two subsequent positions of b. Again, if more than two pairs are contained in S, a random or human choice is taken among them in point 3. The precondition is modified as follows in order to require the existence of at least two positions to snap:

$$\begin{aligned}&((\exists P_1 \,.\, P_1 \in a.\texttt{geo.pos} \wedge \mathcal{CR}_{p_1}(\alpha) \cap \mathcal{CR}(b,\alpha) \neq \varnothing) \vee \\ &\quad (\exists Q_1 \,.\, Q_1 \in b.\texttt{geo.pos} \wedge \mathcal{CR}_{q_1}(\alpha) \cap \mathcal{CR}(a,\alpha) \neq \varnothing)) \wedge \\ &((\exists P_2 \,.\, P_2 \in a.\texttt{geo.pos} \wedge \mathcal{CR}_{p_2}(\alpha) \cap \mathcal{CR}(b,\alpha) \neq \varnothing) \vee \\ &\quad (\exists Q_2 \,.\, Q_2 \in b.\texttt{geo.pos} \wedge \mathcal{CR}_{q_2}(\alpha) \cap \mathcal{CR}(a,\alpha) \neq \varnothing))\end{aligned}$$

- *Right position snapping* ($a \rightarrow\!\Leftarrow b$): this variants requires that all positions of b are snapped. The precondition now requires that the confidence region of all positions in b intersects the confidence region of a, and the confidence region of all positions in a that have a matching with a position in b shall intersect the confidence region of b:

$$\begin{aligned}&\forall Q \in b.\texttt{geo.pos} \,.\, \mathcal{CR}(a,\alpha) \cap \mathcal{CR}_q(\alpha) \neq \varnothing \wedge \\ &\quad \forall P \in match(a,b) \,.\, \mathcal{CR}(a,\alpha) \cap \mathcal{CR}_p(\alpha) \neq \varnothing\end{aligned}$$

  where $match(a,b)$ returns all the positions of a that have a matching with a position of b or that are between two matching positions. Given such precondition, the algorithm requires that the first two phases are substituted by the following one: $\forall Q_i \,.\, b.\texttt{geo.pos}$ identify the corresponding position $P_i \in a$, and add (P_i, Q_i) to S.
- *All positions snapping* ($a \Rightarrow\!\Leftarrow b$): this variant requires that all positions of both a and b are snapped. The precondition in this case requires that the confidence region of all positions of a intersects the confidence region of b, and that the confidence region of all positions in a intersects the confidence region of a:

$$\begin{aligned}&\forall P \in a.\texttt{geo.pos} \,.\, \mathcal{CR}_p(\alpha) \cap \mathcal{CR}(b,\alpha) = \varnothing \wedge \\ &\quad \forall Q \in b.\texttt{geo.pos} \,.\, \mathcal{CR}_q(\alpha) \cap \mathcal{CR}(a,\alpha) = \varnothing\end{aligned}$$

  Given such precondition operations 1 and 2 of the general algorithm require to find a corresponding position for all positions in a and b. □

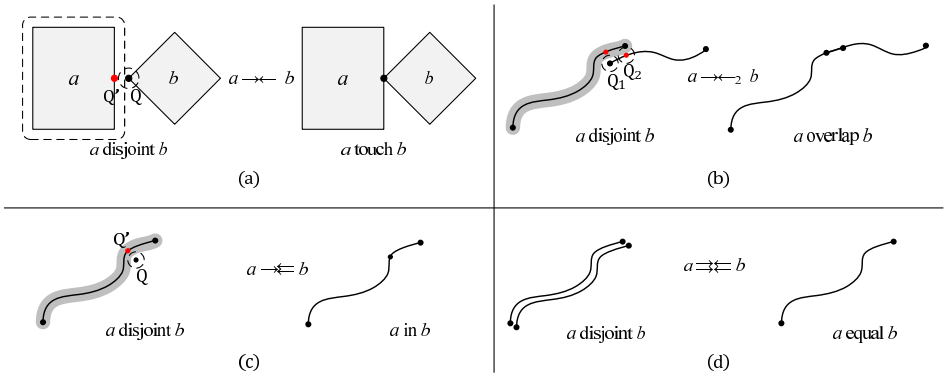


Fig. 4.2. (a) Example of generic *positions snapping* that transforms a *disjoint* relation between surfaces into a *touch* one. (b) Example of *two positions snapping* that transforms a *disjoint* relation between curves into an *overlap* one. (c) Example of *right positions snapping* that transform a *disjoint* relation between a curve and a point into an *in* one. (d) Example of *all positions snapping* that transform a *disjoint* relation between two curves into an *equal* one.

The main steps performed by a position decoupling operation and the necessary preconditions to their application are summarized in the following algorithm. Some examples of its application are provided in Fig. 4.3.

Algorithm 4.8 (Position decoupling). Given two MACS features a and b, the *position decoupling operation* ($\leftarrow\rightarrow$) produces a new geometry for them such that a shared position has been substituted by two distinct and independent positions. It can be performed between two features if and only if there is a position P shared by both of them:

$$\exists P \;.\; P \in a.\texttt{geo.pos} \wedge P \in b.\texttt{geo.pos}$$

The following variants can be recognized:

- *In position decoupling* ($a \overset{in}{\leftarrow\rightarrow} b$): the position P has to be substituted with two positions that are contained in the corresponding objects.
  1. Substitute P with two new positions Q_1 and Q_2 where the distance between them is the minimum representable distance ϵ such that Q_1 *in* b and Q_2 *in* a, where *in* stands for the corresponding topological relation.
  2. Maximize the accuracy of relative distance between Q_1 and Q_2.
- *In left position decoupling* ($a \overset{in_L}{\leftarrow\rightarrow} b$): the position P has to be substituted with two positions Q_1 and Q_2 between which a minimum distance ϵ exists and such that Q_1 *in* b and Q_2 *disjoint* a.
- *Out position decoupling* ($a \overset{out}{\leftarrow\rightarrow} b$): the position P has to be substituted with two positions Q_1 and Q_2 between which a minimum distance ϵ exists and such that Q_1 *disjoint* b and Q_2 *disjoint* a.
- *Cross position decoupling* ($a \overset{cr}{\leftarrow\rightarrow} b$): the position P has to be substituted with two positions Q_1 and Q_2 between which a minimum distance ϵ exists and such that a *cross* b after the decoupling.

- *All in position decoupling* ($a \overset{in}{\leftarrow\!\!\!\!\rightarrow} b$): this position is similar to $a \overset{in_L}{\leftarrow\!\!\!\!\rightarrow} b$, but in this case all sharing positions have to be decoupled, and after the operation the topological relation *a in b* has to be satisfied.
- *All out position decoupling* ($a \overset{out}{\leftarrow\!\!\!\!\rightarrow} b$): this position is similar to $a \overset{out}{\leftarrow\!\!\!\!\rightarrow} b$, but in this case all sharing positions have to be decoupled, and after the operation the topological relation *a disjoint b* has to be satisfied. □

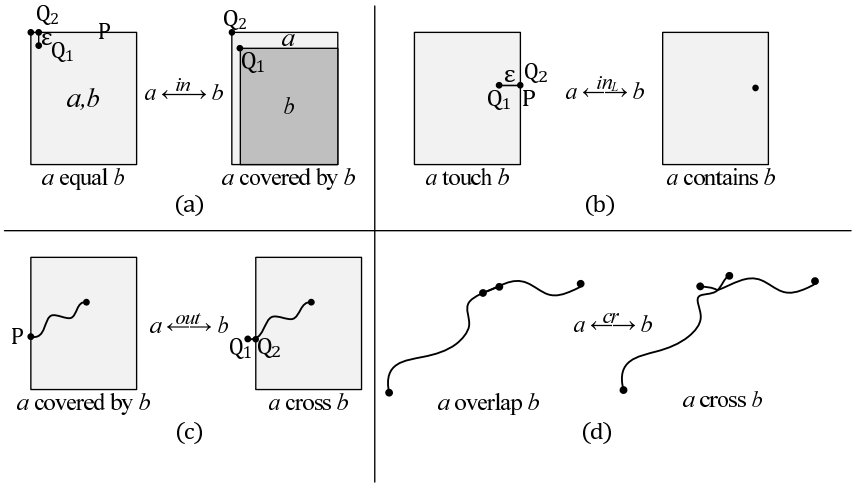


Fig. 4.3. (a) Example of *in positions decoupling* that transforms an *equal* relation between two surfaces into a *covered by* one. (b) Example of *in left positions decoupling* that transforms a *touch* relation between a surface and a point into a *contains* one. (c) Example of *out positions decoupling* that transform a *covered by* relation between a surface and a curve into a *cross* one. (d) Example of *cross positions decoupling* that transforms an *overlap* between two curves into a *cross*.

Finally, the details of the position switching operation are defined in the following algorithm. Some examples of its application are provided in Fig. 4.4.

Algorithm 4.9 (Position switching). Given two MACS features a and b, the *position switching operation* ($\rightleftarrows$) allows one to change the location of a position of b with respect to a curve or surface represented by a. This operation is obtained by combining a snapping operation followed by a decoupling one. In other words the position P to switch has be firstly substituted by a shared position, secondly this shared position is decoupled into the desired direction. The following variants can be recognized:

- *In position switching* ($a \overset{in}{\rightleftarrows} b$): It is the combination of $a \rightarrow\!\leftarrow b$ followed by $a \overset{in}{\leftarrow\!\!\!\!\rightarrow} b$. The operation precondition is the precondition of $a \rightarrow\!\leftarrow b$.
- *Out position switching* ($a \overset{out}{\rightleftarrows} b$): It is the combination of $a \rightarrow\!\leftarrow b$ followed by $a \overset{out}{\leftarrow\!\!\!\!\rightarrow} b$. The operation precondition is the precondition of $a \rightarrow\!\leftarrow b$.

- *Cross position switching* ($a \overset{cr}{\rightleftarrows} b$): It is the combination of $a \rightarrow\!\leftarrow b$ followed by $a \overset{cr}{\longleftrightarrow} b$. The operation precondition is the precondition of $a \rightarrow\!\leftarrow b$.
- *All in position switching* ($a \overset{in}{\rightleftarrows}_{all} b$): It is the combination of $a \rightarrow\!\Leftarrow b$ followed by $a \overset{in}{\leftarrow\!\Rightarrow} b$. The operation precondition is the precondition of $a \rightarrow\!\Leftarrow b$.
- *All out position switching* ($a \overset{out}{\rightleftarrows}_{all} b$): It is the combination of $a \rightarrow\!\Leftarrow b$ followed by $a \overset{out}{\leftarrow\!\Rightarrow} b$. The operation precondition is the precondition of $a \rightarrow\!\Leftarrow b$. □

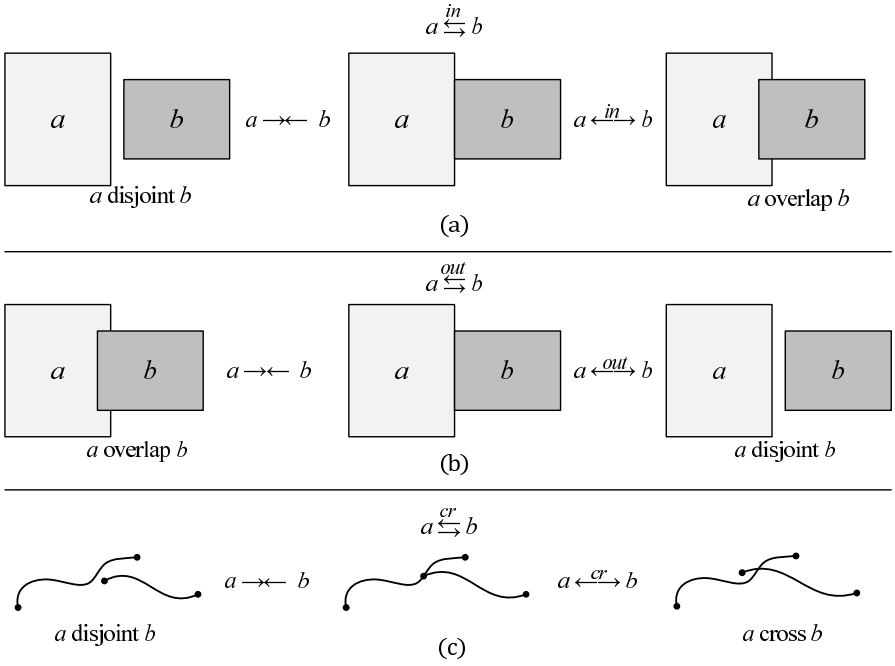


Fig. 4.4. (a) Example of *in positions switching* that transforms a *disjoint* relation between two surfaces into an *overlap* one. (b) Example of *out positions switching* that transform an *overlap* relation between two surfaces into a *disjoint* one. (c) Example of *cross positions switching* that transform a *disjoint* relation between two curves into a *cross* one.

The previous algorithms defines how the geometry of a MACS feature can be modified with respect to the geometry of another feature. The following algorithm describe how metric and logical information can be aligned together, eventually using such operations.

Algorithm 4.10 (Alignment of positions with respect to logical observations). Given an integrated MACS databases $\mathtt{macs}_3$ and the initial databases $\mathtt{macs}_1$ and $\mathtt{macs}_2$, the alignment of positions with respect to logical observations is an iterative process that is executed until the following condition holds:

$$\{(\mathtt{o}_1, \mathtt{o}_2) \mid (\mathtt{o}_1, \mathtt{o}_2) \in \mathtt{OBJ}_3 \times \mathtt{OBJ}_3 \wedge \mathtt{o}_1\ r_{soft}\ \mathtt{o}_2 \wedge \langle \mathtt{o}_1, \mathtt{o}_2, \mathtt{R} \rangle \in \mathtt{REL}_3 \wedge r_{soft} \notin \mathtt{R}\} = \varnothing$$

The iterated core algorithm is composed of the following steps:

1. For each violation of consistency between a pair of objects $(\mathtt{o}_1, \mathtt{o}_2)$, the necessary relation transition $r_A \to r_B$ is identified.
2. For each relation transition its applicability is evaluated; in particular, some transitions are not admitted a priori, other ones require operations that, in specific cases, could not be applied (see Tables 4.2-4.10).
3. For each relation transition that is not applicable, since its preconditions are not satisfied, the user intervention is required.
4. For each relation transition $r_A \to r_B$ that is applicable and such that $\mathtt{o}_1\ r_{soft}\ \mathtt{o}_2$ in $\mathtt{macs}_i$ $(i \in \{1,2\})$ and $r_{soft} = r_B$, namely r_B is valid in one of the two source databases, the accuracy of relative distances between the involved objects is maximized in $\mathtt{macs}_i$. In other words, the covariance of all pairs of positions $(P_i, Q_i) \in \mathtt{o}_1.\mathtt{geo.pos} \times \mathtt{o}_2.\mathtt{geo.pos}$ having intersecting confidence regions are set to the value $(\sigma_P^2 + \sigma_Q^2)/2$ (see Eq.3.4) in the corresponding matrix $\mathtt{C}_{\mathtt{DB}_i}$. By maximizing the accuracy of the relative distance between the two objects, they become a rigid body and move accordingly without changing the relative positions of their points. After this transformation of the covariance matrix $\mathtt{C}_{\mathtt{DB}_i}$, the computation of $\mathtt{DB}_3 = \textsc{MetricPosInt}_{kalman}(\mathtt{DB}_1, \mathtt{DB}_2, \mathtt{C}_{\mathtt{DB}_1}, \mathtt{C}_{\mathtt{DB}_2})$ is performed again.
5. For each relation transition $r_A \to r_B$ that is applicable but does not satisfy the previous condition, it is necessary to modify some pairs of positions $(P_j, Q_j) \in \mathtt{O}_1.\mathtt{geo.pos} \times \mathtt{O}_2.\mathtt{geo.pos}$ having intersecting supports, through the application of the operations requested by the transition, as shown in Tables 4.2-4.10. This leads to the definition of a new $\mathtt{DB}'_3$ that needs to be integrated with $\mathtt{DB}_3$ in order to obtain the final database:

$$\mathtt{DB}_{\mathtt{final}} = \textsc{MetricPosInt}_{kalman}(\mathtt{DB}_3, \mathtt{DB}'_3, \mathtt{C}_{\mathtt{DB}_3}, \mathtt{C}_{\mathtt{DB}'_3})$$

□

Notice that in Tables 4.2-4.10 some allowed transitions involve pairs of relations that are not near. These cases are allowed since the transition can be obtained with a local geometry modification, i.e. by applying a minimal change on objects positions.

The main idea underlying phases 4 and 5 is that the positions of objects involved into a particular topological relation have to become a rigid body that can move in space but in a uniform manner: they have to maintain their relative reciprocal positions in order to keep the effect of the previous transformations. This is the aim of the covariance correction proposed in phase 4 and in the operations eventually applied in phase 5. In other words, the idea is that the knowledge of a particular topological relation augments the confidence about the relative distance between the involved objects, hence its accuracy increases.

The approach presented here differs from to the one proposed in [58, 59] for several reasons. First of all, this approach considers the integration of both metric and topological information, while [58,59] the authors suppose to have only one set of topological relation that has to be valid on the integrated geometry. Using a set of equations representing the topological relations that are valid in the two source

$d_{*,*}$	rel							
	$t_{*,*}$	$i_{*,*}$	$c_{*,*}$	$e_{*,*}$	$r_{*,*}$	$o_{*,*}$	$b_{*,*}$	$v_{*,*}$
$d_{S,S}$	(1) $a \rightarrow\leftarrow b$	NA	NA	NA	ND	(4) $a \overset{in}{\rightleftarrows} b$	NA	NA
$d_{S,C}$	(1) $a \rightarrow\leftarrow b$	ND	NA	ND	(2) $a \overset{in}{\rightleftarrows} b$	ND	ND	NA
$d_{S,P}$	(2) $a \rightarrow\leftarrow b$	ND	(2) $a \overset{in}{\rightleftarrows}_{all} b$	ND	ND	ND	ND	ND
$d_{C,S}$	(1) $a \rightarrow\leftarrow b$	NA	ND	ND	(2) $a \overset{in}{\rightleftarrows} b$	ND	NA	ND
$d_{C,C}$	(1) $\partial a \rightarrow\leftarrow_1 b$ or $a \rightarrow\leftarrow_1 \partial b$ or $\partial a \rightarrow\leftarrow \partial b$	NA	NA	NA	(1) $a \overset{cr}{\rightleftarrows} b$	(1) $a \rightarrow\leftarrow_2 b$	NA	NA
$d_{C,P}$	(2) $\partial a \rightarrow\leftarrow b$	ND	(2) $a^{\circ} \rightarrow\Leftarrow b$	ND	ND	ND	ND	ND
$d_{P,S}$	(2) $a \rightarrow\leftarrow b$	(2) $a \overset{in}{\rightleftarrows}_{all} b$	ND	ND	ND	ND	ND	ND
$d_{P,C}$	(2) $\partial a \rightarrow\leftarrow b$	(2) $a^{\circ} \rightarrow\Leftarrow b$	ND	ND	ND	ND	ND	ND
$d_{P,P}$	ND	ND	ND	(3) $a \Rightarrow\Leftarrow b$	ND	ND	ND	ND

Table 4.2. Transitions between topological relations: case *disjoint* → *rel*. The relation acronyms are as follows: *d* disjoint, *t* touch, *i* in, *b* coveredBy, *c* contains, *v* covers, *e* equal, *r* cross, and *o* overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

datasets is not a valid approach in our case, because if such sets contain discordant information the method cannot find a solution that satisfy all the equations. Moreover, our method consider not only single relations, but also sets (disjunctions) of topological relations between objects, hence the number of necessary equations, that have to be added into the system in the approach of [58] [59], can increase considerably making the integration impracticable. Finally, thanks to the role covered by the accuracy of the relative distances, most of the topological relations that are valid before the integration, remain satisfied also in the integrated database: in practice very few relations are violated after the metric integration phase.

In order to prove that the proposed operations (shown in Tables 4.2-4.10) are sufficient conditions for obtaining the needed relation transitions, we show below

$t_{*,*}$	rel							
	$d_{*,*}$	$i_{*,*}$	$c_{*,*}$	$e_{*,*}$	$r_{*,*}$	$o_{*,*}$	$b_{*,*}$	$v_{*,*}$
$t_{S,S}$	(1) $a \overset{out}{\Longleftrightarrow} b$	NA	NA	NA	ND	(3) $a \overset{in_L}{\longleftrightarrow} b$	NA	NA
$t_{S,C}$	(1) $a \overset{out}{\Longleftrightarrow} b$	ND	NA	ND	(req. 1) $a \overset{cr}{\longleftrightarrow} b$	ND	ND	(req. 1) $a \overset{in_L}{\longleftrightarrow} b$
$t_{S,P}$	(2) $a \overset{out}{\Longleftrightarrow} b$	ND	(2) $a \overset{in_L}{\longleftrightarrow} b$	ND	ND	ND	ND	
$t_{C,S}$	(1) $a \overset{out}{\Longleftrightarrow} b$	NA	ND	ND	(req. 1) $a \overset{cr}{\longleftrightarrow} b$	ND	(req. 1) $b \overset{in_L}{\longleftrightarrow} a$	ND
$t_{C,C}$	(1) $a \overset{out}{\Longleftrightarrow} b$	NA	NA	NA	(1) $a \overset{cr}{\longleftrightarrow} b$	(1) $a \rightarrow\!\leftarrow_2 b$	NA	NA
$t_{C,P}$	(2) $a \overset{out}{\Longleftrightarrow} b$	ND	(2) $a \overset{in_L}{\longleftrightarrow} b$	ND	ND	ND	ND	ND

Table 4.3. Transition between topological relations: case *touch* → *rel*. The relation acronyms are as follows: *d* disjoint, *t* touch, *i* in, *b* coveredBy, *c* contains, *v* covers, *e* equal, *r* cross, and *o* overlap. Each cell reports in round brackets the distance between the two considered topological relations ("*req. d*" means that the transition is allowed only when the distance between the matrix of the current relation and the requested one is *d*) and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

$e_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$i_{*,*}$	$c_{*,*}$	$r_{*,*}$	$o_{*,*}$	$b_{*,*}$	$v_{*,*}$
$e_{S,S}$	NA	NA	NA	NA	ND	NA	(3) $a \overset{in}{\longleftrightarrow} b$	(3) $a \overset{out}{\longleftrightarrow} b$
$e_{C,C}$	NA	NA	NA	NA	NA	(2) $a \overset{out}{\longleftrightarrow} b$	(3) $\partial a \overset{out}{\longleftrightarrow} b$	(3) $\partial b \overset{out}{\longleftrightarrow} a$
$e_{P,P}$	(3) $a \overset{out}{\longleftrightarrow} b$	ND	ND	ND	ND	ND	ND	ND

Table 4.4. Transition between topological relations: case *equal* → *rel*. The relation acronyms are as follows: *d* disjoint, *t* touch, *i* in, *b* coveredBy, *c* contains, *v* covers, *e* equal, *r* cross, and *o* overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

the proof of this property for the transitions starting from a disjoint relation. In a similar way the same property can be proved for the other transitions.

Theorem 4.3 (Operations for disjoint transitions). *Let us consider Tab. 4.2 showing the allowed transitions starting from the disjoint relation. Each column,*

$r_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$i_{*,*}$	$c_{*,*}$	$e_{*,*}$	$o_{*,*}$	$b_{*,*}$	$v_{*,*}$
$r_{S,C}$	NA	(1) $a \rightarrow\!\!\!\!\leftarrow (a \cap b_P)$	ND	NA	ND	ND	ND	(1) $a \rightarrow\!\!\!\!\leftarrow (b_P \setminus a)$
$r_{C,C}$	NA	(1) $a \overset{out}{\leftarrow\!\!\!\!\rightarrow} b$ $(\partial a \rightarrow\!\!\leftarrow b$ or $a \rightarrow\!\!\leftarrow \partial b)$	NA	NA	NA	(1) $a \rightarrow\!\!\leftarrow_2 b$	NA	NA

Table 4.5. Transition between topological relations: case $cross \rightarrow rel$. The relation acronyms are as follows: d disjoint, t touch, i in, b coveredBy, c contains, v covers, e equal, r cross, and o overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert. b_P (a_P) is the set of representative points corresponding to the positions used for representing the geometry of b (a).

$i_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$c_{*,*}$	$e_{*,*}$	$r_{*,*}$	$o_{*,*}$	$b_{*,*}$	$v_{*,*}$
$i_{S,S}$	NA	NA	NA	NA	ND	(4) $a \overset{out}{\rightleftarrows} b$	(1) $a \rightarrow\!\!\leftarrow b$	NA
$i_{C,S}$	NA	NA	ND	ND	(2) $a \overset{out}{\leftarrow\!\!\rightarrow} b$	ND	(1) $a \rightarrow\!\!\leftarrow b$	ND
$i_{P,S}$	(2) $a \overset{out}{\rightleftarrows} b$	(2) $b \rightarrow\!\!\!\!\leftarrow a$	ND	ND	ND	ND	ND	ND
$i_{C,C}$	NA	NA	NA	NA	(1) $a \overset{out}{\leftarrow\!\!\!\!\rightarrow} b$ $a^\circ \rightarrow\!\!\leftarrow_1 b^\circ$	(1) $a \overset{out}{\leftarrow\!\!\!\!\rightarrow} b$ $a \rightarrow\!\!\leftarrow_2 b$	(1) $\partial a \rightarrow\!\!\leftarrow \partial b$	NA
$i_{P,C}$	(2) $a \overset{out}{\leftarrow\!\!\rightarrow} b$	(2) $a \rightarrow\!\!\leftarrow \partial b$	ND	ND	ND	ND	ND	ND

Table 4.6. Transition between topological relations: case $in \rightarrow rel$. The relation acronyms are as follows: d disjoint, t touch, i in, b coveredBy, c contains, v covers, e equal, r cross, and o overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

representing a given target relation $rel_{*,*}$, reports for each type pair t_1, t_2 the operations that are a sufficient condition for obtaining the target relation, starting from two disjoint objects of types t_1, t_2.

Proof. We present the proof for the first column of the table regarding the transition $(a\ disjoint\ b) \rightarrow (a\ touch\ b)$, the proof for the other columns follows a similar

$o_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$i_{*,*}$	$c_{*,*}$	$e_{*,*}$	$r_{*,*}$	$b_{*,*}$	$v_{*,*}$
$o_{S,S}$	NA	$a \rightarrow\!\!\!\leftarrow (a \cap b_P)$ or $b \rightarrow\!\!\!\leftarrow (a_P \cap b)$	NA	NA	NA	NA	$b \rightarrow\!\!\!\leftarrow (a_P \setminus b)$ or $a \rightarrow\!\!\!\leftarrow (a_P \setminus b)$	$a \rightarrow\!\!\!\leftarrow (b_P \setminus a)$ or $b \rightarrow\!\!\!\leftarrow (b_P \setminus a)$
$o_{C,C}$	NA	$a \overset{out}{\leftarrow\!\!\!\rightarrow} b$ ($\partial a \rightarrow\!\!\!\leftarrow b$ *or* $a \rightarrow\!\!\!\leftarrow \partial b$)	NA	NA	NA	$a \overset{cr}{\leftarrow\!\!\!\rightarrow} b$	NA	NA

Table 4.7. Transition between topological relations: case *overlap* → *rel*. The relation acronyms are as follows: *d* disjoint, *t* touch, *i* in, *b* coveredBy, *c* contains, *v* covers, *e* equal, *r* cross, and *o* overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert. b_P (a_P) is the set of representative points corresponding to the positions used for representing the geometry of b (a).

$c_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$i_{*,*}$	$e_{*,*}$	$r_{*,*}$	$o_{*,*}$	$b_{*,*}$	$v_{*,*}$
$c_{S,S}$	NA	NA	NA	NA	ND	(1) $a \overset{out}{\rightleftarrows} b$	NA	(1) $a \rightarrow\!\!\!\leftarrow \partial b$
$c_{S,C}$	NA	NA	NA	ND	(2) $a \overset{out}{\leftarrow\!\!\!\rightarrow} b$	(1)	ND	(1) $a \rightarrow\!\!\!\leftarrow b$
$c_{S,P}$	(2) $a \overset{out}{\rightleftarrows} b$	(2) $a \rightarrow\!\!\!\leftarrow b$	ND	ND	ND	ND	ND	ND
$c_{C,C}$	NA	NA	NA	NA	(1) $a \overset{out}{\leftarrow\!\!\!\rightarrow} b$ $a^\circ \rightarrow\!\!\!\leftarrow_1 b^\circ$	(1) $a \overset{out}{\leftarrow\!\!\!\rightarrow} b$ $a \rightarrow\!\!\!\leftarrow_2 b$	NA	(1) $\partial a \rightarrow\!\!\!\leftarrow \partial b$
$c_{C,P}$	$a \overset{out}{\leftarrow\!\!\!\rightarrow} b$	$\partial a \rightarrow\!\!\!\leftarrow b$	ND	ND	ND	ND	ND	ND

Table 4.8. Transition between topological relations: case *contains* → *rel*. The relation acronyms are as follows: *d* disjoint, *t* touch, *i* in, *b* coveredBy, *c* contains, *v* covers, *e* equal, *r* cross, and *o* overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

reasoning. Here is used the notation introduced in Tab. 3.2 for representing the patterns of the 9-intersection matrices.

- Type pairs (S,S), (S,C), (C,C) and (C,S).
  According to Tab. 3.2 the pattern for the disjoint relation in these cases is FFT – $\mathbf{FF}T$ – TTT.
  - Case (S,S)
    If we apply the required operation $a \rightarrow\!\!\!\leftarrow b$ when objects types are surfaces,

$b_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$i_{*,*}$	$c_{*,*}$	$e_{*,*}$	$r_{*,*}$	$o_{*,*}$	$v_{*,*}$
$b_{S,S}$	NA	NA	(1) $a \overset{in_L}{\Leftarrow\Rightarrow} b$	NA	(3) $a \Rightarrow\Leftarrow b$	ND	(1) $a \overset{out}{\leftarrow\rightarrow} b$	NA
$b_{C,S}$	NA	(1) $b \rightarrow\Leftarrow a$	(1) $a \overset{in_L}{\Leftarrow\Rightarrow} b$	ND	ND	 $a \overset{out}{\leftarrow\rightarrow} b$	ND	ND
$b_{C,C}$	NA	NA	(1) $\partial a \overset{in_L}{\Leftarrow\Rightarrow} b$	NA	NA	(1) $a \overset{out}{\Leftarrow\Rightarrow} b$ $a^\circ \rightarrow\leftarrow_1 b^\circ$	(1) $a \overset{out}{\Leftarrow\Rightarrow} b$ $a \rightarrow\leftarrow_2 b$	NA

Table 4.9. Transition between topological relations: case $coveredBy \rightarrow rel$. The relation acronyms are as follows: d disjoint, t touch, i in, b coveredBy, c contains, v covers, e equal, r cross, and o overlap. Each cell reports in round brackets the distance between the two considered topological relations and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

$v_{*,*}$	rel							
	$d_{*,*}$	$t_{*,*}$	$i_{*,*}$	$c_{*,*}$	$e_{*,*}$	$r_{*,*}$	$o_{*,*}$	$b_{*,*}$
$v_{S,S}$	NA	NA	NA	(1) $b \overset{in_L}{\Leftarrow\Rightarrow} a$	(3) $a \Rightarrow\Leftarrow b$	ND	(3) $a \overset{out}{\leftarrow\rightarrow} b$	NA
$v_{S,C}$	NA	(1) $a \rightarrow\Leftarrow b$	ND	(1) $b \overset{in_L}{\Leftarrow\Rightarrow} a$	ND	(1) $a \overset{out}{\leftarrow\rightarrow} b$	ND	ND
$v_{C,C}$	NA	NA	NA	(1) $\partial b \overset{in_L}{\Leftarrow\Rightarrow} a$	NA	(1) $a \overset{out}{\Leftarrow\Rightarrow} b$ $a^\circ \rightarrow\leftarrow_1 b^\circ$	(1) $a \overset{out}{\Leftarrow\Rightarrow} b$ $a \rightarrow\leftarrow_2 b$	NA

Table 4.10. Transition between topological relations: case $covers \rightarrow rel$. The relation acronyms are as follows: d disjoint, t touch, i in, b coveredBy, c contains, v covers, e equal, r cross, and o overlap. Each cell reports in round brackets the distance between the two considered topological relations ("*req. d*" means that the transition is allowed only when the distance between the matrix of the current scene and the requested relation rel is d) and below the operations that have to be applied in order to obtain the requested relation. The symbol *ND* indicates that the target relation rel is not defined for the considered geometric types, while *NA* indicates that *rel* cannot be obtained without the intervention of a domain expert.

then the geometries of a and b are locally modified in such a way that after the modification a and b share at least one position. As a consequence, the intersection $\partial a \cap \partial b$ is now not empty and the pattern becomes: FFT – $F\mathbf{T}T$ – TTT, which is the pattern of the touch relation for types (S, S).

– Case (S, C)
The application of the required operation $a \rightarrow\leftarrow b$ in presence of a surface and a curve, leads to the situation in which either $\partial a \cap \partial b$ becomes not empty

or $\partial a \cap b^{\circ}$ does Therefore, the resulting pattern becomes $FFT-F\mathbf{T}T-TTT$ or $FFT-\mathbf{T}FT-TTT$, which again are patterns of touch.
  - Case (C,S)
    The reasoning is the dual of the previous one.
  - Case (C,C)
    When the involved objects are two curves, the required operation is $\partial a \nrightarrow\nleftarrow \partial b$ or $\partial a \nrightarrow\nleftarrow_1 b$ or $a \nrightarrow\nleftarrow_1 \partial b$. As a consequence, either their boundaries share a position ($\partial a \cap \partial b \neq \varnothing$), or the boundary of one curve shares a position with the interior of the other ($\partial a \cap b^{\circ} \neq \varnothing \vee a^{\circ} \cap \partial b \neq \varnothing$). Therefore, the pattern becomes $FFT-F\mathbf{T}T-TTT$ or $FFT-\mathbf{T}FT-TTT$ ($F\mathbf{T}T-FFT-TTT$), which again are patterns of touch.
- Type pairs (S,P) and (C,P)
  According to Table 3.2 the disjoint pattern in these cases is $FFT-\mathbf{F}FT-TFT$.
  - Case (S,P)
    The required operation is $a \nrightarrow\nleftarrow b$. The application of this operation leads to a situation where the point coincides with one curve endpoint: $\partial a \cap b \neq \varnothing \wedge a^{-} \cap b = \varnothing$. Therefore, the pattern becomes $FFT-\mathbf{T}FT-\mathbf{F}FT$, which is the pattern of touch.
  - Case (C,P)
    The required operation is $\partial a \nrightarrow\nleftarrow b$. After the application of such operation, one endpoint of the curve coincides with the point, thus $\partial a \cap b \neq \varnothing \wedge a^{-} \cap b = \varnothing$. Therefore, the pattern becomes $FFT-\mathbf{T}FT-\mathbf{F}FT$, which is the pattern of touch.
- Type pairs (P,S) and (P,C): the reasoning in this case is similar to the previous one. □

4.5 Properties of the Integration Process

This section presents some properties of the proposed integration process. In particular, we start by discussing the central role covered by the accuracy of each measure during the integration, showing that the final position of a location depends not only on the integrated measures, but also on their accuracy and their correlation with near positions. Subsequently, we state that the accuracy of measures and the certainty of logical observations are always increased after the integration process or at least coincide with the accuracy and certainty of the most accurate source database, respectively. In order to demonstrate these properties, we first analyze the trend of the Kalman matrix coefficients in relation to the different accuracies of the two source databases. Given two Macs databases $\texttt{macs}_1$ and $\texttt{macs}_2$ that have to be integrated, the coefficients of the Kalman matrix associate to each absolute or relative measure a value proportional to its accuracy and normalized with respect to the overall accuracy of the two source databases. In particular, the coefficients of the Kalman matrix assume a value as follows:

- The coefficients $\texttt{k}_{x_p,x_p} = \texttt{k}_{y_p,y_p}$ related to the variance of a position P have a value between 0 and 1.

$$\mathtt{k}_{x_p,x_p} = \begin{cases} a \in [0, 0.5) & \text{if } \mathtt{DB}_m^2 <_{acc} \mathtt{DB}_m^1 \\ a = 0.5 & \text{if } \mathtt{DB}_m^2 =_{acc} \mathtt{DB}_m^1 \\ a \in (0.5, 1] & \text{if } \mathtt{DB}_m^2 >_{acc} \mathtt{DB}_m^1 \end{cases}$$

- The coefficients $\mathtt{k}_{x_p,x_q} = \mathtt{k}_{y_p,y_q}$ for $Q \neq P$ related to the covariance between two different positions P and Q have a value between -1 and 1.

$$\mathtt{k}_{x_p,x_q} = \begin{cases} b \in [-1, 0) & \text{if } \mathtt{DB}_m^2 <_{acc} \mathtt{DB}_m^1 \\ b = 0 & \text{if } \mathtt{DB}_m^2 =_{acc} \mathtt{DB}_m^1 \\ b \in (0, 1] & \text{if } \mathtt{DB}_m^2 >_{acc} \mathtt{DB}_m^1 \end{cases}$$

- The coefficients $\mathtt{k}_{x_p,y_p} = \mathtt{k}_{y_p,x_p}$ corresponding to the covariance between the x and y coordinates of a same position P, and the coefficients $\mathtt{k}_{x_p,y_q} = \mathtt{k}_{y_p,x_q}$ for $Q \neq P$, corresponding to the covariance between the x and y coordinate of two distinct positions P and Q, are zero.

From these characteristics of the Kalman matrix, the first property of the integration process can be derived.

Property 4.3. Given two MACS databases $\mathtt{macs}_1$ and $\mathtt{macs}_2$ that have to be integrated, the shift of a position P from its location in $\mathtt{macs}_1$ increases if its accuracy in $\mathtt{macs}_2$ is greater than its accuracy in $\mathtt{macs}_1$.

Proof. Given the vectors $\mathtt{V}_{\mathtt{DB}_1}$ and $\mathtt{V}_{\mathtt{DB}_2}$ built as in Alg. 4.2, the vector of position indices $\mathtt{V}_{\mathtt{DB}_3}$ for the integrated database $\mathtt{macs}_3$ is obtained from the Eq. 4.6 as:

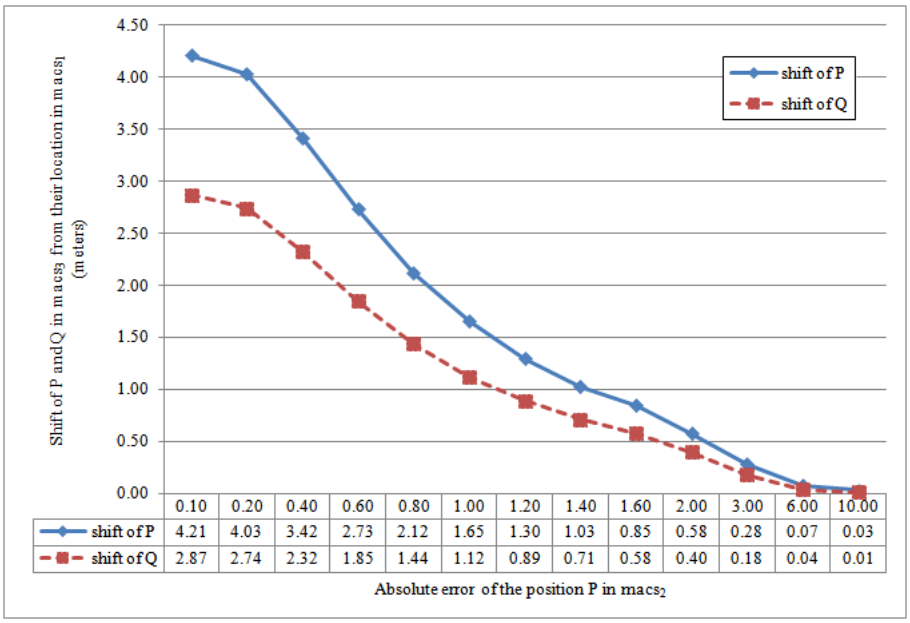
$$\mathtt{V}_{\mathtt{DB}_3} = \mathtt{V}_{\mathtt{DB}_1} + \mathtt{K} \cdot (\mathtt{V}_{\mathtt{DB}_2} - \mathtt{V}_{\mathtt{DB}_1})$$

Let us suppose for simplicity that inside the two source databases there are only two positions $P = (x_p, y_p)$ and $Q = (x_q, y_q)$. The shift of the integrated x coordinate of the position P, denoted as x_P^3, from its original value in $\mathtt{DB}_m^1$ becomes:

$$x_p^3 - x_p^1 = k_{x_p,x_p} \cdot (x_p^2 - x_p^1) + \mathtt{k}_{x_p,x_q} \cdot (x_q^2 - x_q^1)$$

where $k_{i,j}$ is the coefficient of the Kalman matrix in row i and column j, respectively. Independently from the measurements contained in $\mathtt{macs}_2$ (x_p^2 and x_q^2), the shift of x_p^3 from the value x_p^1 directly depends upon the coefficient $\mathtt{k}_{x_p,x_p}$ and $\mathtt{k}_{x_p,x_q}$ of the Kalman matrix. The trend of the Kalman matrix coefficients states that the more the accuracy of the position P in $\mathtt{macs}_2$ increases with respect to the accuracy of the same position in $\mathtt{macs}_1$, the more the value of the coefficients $\mathtt{k}_{x_p,x_p}$ and $\mathtt{k}_{x_p,x_q}$ tends to one, determining a greater shift of x_p^3, that can eventually become equal to x_p^2. Notice that the shift of P is due not only to a direct update of its measure in $\mathtt{macs}_2$, but also to the propagation of the update of other positions, in a quantity that directly depends upon the accuracy of the relative distance between them. □

Example 4.4. Let us suppose that $\mathtt{macs}_1$ contains two positions $P = (100, 100)$ and $Q = (123, 123)$ that have both an absolute accuracy *abs_err* of $0.8m$ (with *abs_fr* = 95%), while their relative distance has an accuracy of $0.6m$ (with *rel_fr* = 95%). Conversely, $\mathtt{macs}_2$ contains another measure for $P = (103, 103)$ that has to be



	0.10	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	2.00	3.00	6.00	10.00
shift of P	4.21	4.03	3.42	2.73	2.12	1.65	1.30	1.03	0.85	0.58	0.28	0.07	0.03
shift of Q	2.87	2.74	2.32	1.85	1.44	1.12	0.89	0.71	0.58	0.40	0.18	0.04	0.01

Fig. 4.5. Shift of the positions P and Q with respect to their original measures in $\texttt{macs}_1$, considering different absolute error *abs_err* for P in $\texttt{macs}_2$.

integrated with the one contained in $\texttt{macs}_1$. The test considers the different shift of P and Q in $\texttt{macs}_3$ from their positions in $\texttt{macs}_1$, varying the error *abs_err* of P in $\texttt{macs}_2$ during the integration. The results of this test are reported in the graph of Fig. 4.5. The graph clearly illustrates that greater is the accuracy of x_p in $\texttt{macs}_2$ (i.e. smaller is its circular error), greater is the shift of both points after the integration process. Moreover, even if the trend for the two points is similar, the shift of P is greater than the shift of Q, because it is directly involved in the integration process, while the shift of Q is only due to the propagation of the P integration.

Property 4.4. Given the MACS database $\texttt{macs}_3$ obtained by integrating two source MACS databases $\texttt{macs}_1$ and $\texttt{macs}_2$, the accuracy of each integrated measure in $\texttt{macs}_3$ is not smaller than the accuracy of the corresponding measure in the two source databases. In particular, if the accuracy of a measure in one database is very high, then the corresponding measure in the other database does not influence the integration process and the resulting accuracy corresponds to the greatest one.

Proof. The metric accuracy of a position P is defined in Sec. 3.3 and it inversely depends on the positions variance. The variance for the integrated position P in $\texttt{macs}_3$ is computed using the Eq. 4.7:

$$\mathtt{C}_{\mathtt{DB}_3} = (I - \mathtt{K}) \cdot \mathtt{C}_{\mathtt{DB}_1} \cdot (I - \mathtt{K})^T + \mathtt{K} \cdot \mathtt{C}_{\mathtt{DB}_2} \cdot \mathtt{K}^T$$

Let us suppose that $\texttt{macs}_2$ contains a very accurate measure for P, then as stated above the coefficient k_{x_P,x_P} (or equivalently k_{y_P,y_P}) of the Kalman matrix has a

value near to one. From this, it follows that the resulting variance value in $\mathtt{C}_{\mathtt{DB}_3}$ is very close to (at most coincides with) the element contained in $\mathtt{C}_{\mathtt{DB}_2}$, namely to the most accurate one. Conversely, if $\mathtt{macs}_1$ contains a very accurate measure for P, then the coefficient k_{x_P,x_P} of the Kalman matrix has a value near to zero and the variance value in $\mathtt{C}_{\mathtt{DB}_3}$ for P is very close to the one in $\mathtt{C}_{\mathtt{DB}_1}$. In the other cases, if the two source databases contain both relatively accurate measures for P, the diagonal position k_{x_P,x_P} of the Kalman matrix contains a value positive but smaller than one. This value multiplied with the elements of the original matrices produces a value that is smaller than the original ones; moreover, their sum is smaller than each original value as the coefficient of K are normalised with respect to the overall accuracy of the two databases (the sum of the two original variances). As a consequence, as the variance of each measure decreases at each iteration, the quality of the integrated position always increases. □

Example 4.5. Let us consider again the two MACS databases in Ex. 4.4 and perform the integration between them taking into account the new value of absolute error computed after the integration process. The variation between the error of the integrated measures and the error of the same measures in $\mathtt{macs}_1$ is reported in the graph of Fig. 4.6, considering different values of absolute error for the position P in $\mathtt{macs}_2$. Notice that as P becomes less accurate, such variation decreases. When the error of P in $\mathtt{macs}_2$ becomes equal to the error of P in $\mathtt{macs}_1$ ($0.80m$), the variation is still greater than zero: the integration of two measures with the same accuracy produces a new measure that is more accurate than the two source ones. Finally, if the measure of P is very inaccurate, it has not effect during the integration process also as regards to the error of the integrated measure; indeed, as the error in $\mathtt{macs}_2$ increases, the error variation settles to a value near to zero.

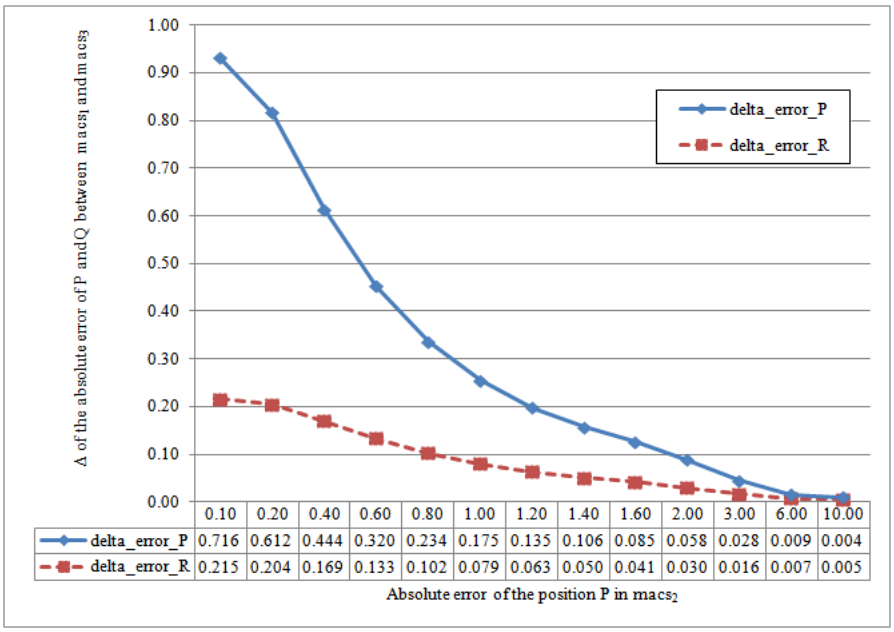
From this property and the definition of average global accuracy estimator $acc_m(\mathtt{macs})$, given in Sec. 3.3, it directly derives that $acc_m(\mathtt{macs}_3)$ is always greater than or equal to $acc_m(\mathtt{macs}_1)$ and $acc_m(\mathtt{macs}_2)$.

Property 4.5. Given a MACS database $\mathtt{macs}_3$ obtained by integrating two source MACS databases $\mathtt{macs}_1$ and $\mathtt{macs}_2$, the certainty of each logical observation does not decrease during the integration process, it can only remain unchanged or increase.

Proof. The certainty of each logical observation is defined in Eq. 3.11. Discarding the optimization mentioned at the end of Sec. 4.3, given two disjunction of topological relations R_1 and R_2, their integration always produces a set of relations R_3 whose cardinality is smaller than the cardinality of both the original ones, or equal to the smallest ones ($|R_3| \leq min(|R_1|, |R_2|)$. Therefore, putting this new cardinality into the certainty formula, a certainty index is obtained that is equal to the greater one or is greater than both the original ones. □

From this property and the definition of average global certainty estimator $acc_t(\mathtt{macs})$, given in Sec. 3.3, it directly derives that $acc_t(\mathtt{macs}_3)$ is always greater than or equal to $acc_t(\mathtt{macs}_1)$ and $acc_t(\mathtt{macs}_2)$.

The presented properties allows one to conclude that the proposed integration process does not decrease (and usually increases) the overall knowledge of a certain



	0.10	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	2.00	3.00	6.00	10.00
delta_error_P	0.716	0.612	0.444	0.320	0.234	0.175	0.135	0.106	0.085	0.058	0.028	0.009	0.004
delta_error_R	0.215	0.204	0.169	0.133	0.102	0.079	0.063	0.050	0.041	0.030	0.016	0.007	0.005

Fig. 4.6. Variation of the absolute error for the integrated positions P and Q in $\texttt{macs}_3$ with respect to the value in $\texttt{macs}_1$, considering different absolute error for P in $\texttt{macs}_3$.

geographical area represented in a MACS database with respect to both metric and logical observations.

4.6 Distributed Integration of two MACS databases

This section extends the integration framework previously presented, in order to make its application feasible in a distributed context even in presence of an huge amount of data. In particular, the integration context considered in the following is represented by an SDI environment such that:

- The reference SDI region is partitioned into several areas, which are managed by local agencies (called *SDI members*).
- A central agency (called *SDI manager*) is active and responsible for creating and maintaining a global repository that contains the current global spatial database.
- Data are structured in several datasets having different metric accuracies. Each SDI member provides its own data to the SDI manager and maintains a local copy for performing its operations.
- Updates, with different levels of accuracy, are handled by each SDI member and they are propagated automatically to the SDI manager and from here, if necessary, to the adjacent SDI members.

The aim of this section is to discuss how metric and logical observations can be locally integrated by each SDI member and then globally combined together by the SDI manger in order to obtain the final result.

4.6.1 Distributed Integration of Metric Information

In the SDI context introduced above, it is not reasonable that the SDI manager performs the integration by applying the proposed procedure on the whole global SDI database. However, Ex. 1.4 illustrates the problems that can arise when in a distributed context an update is performed only locally. In particular, the main issue regards the propagation of the integration effects on near databases owned by other SDI members.

This section discusses how the Kalman integration introduced in Sec. 4.2 can be performed locally by each SDI member and then combined globally by a centralized SDI manager, exploiting the information locally computed.

Given two set of position indices $\mathtt{V}_{\mathtt{DB}_1}$ and $\mathtt{V}_{\mathtt{DB}_2}$ to be integrated, they can be subdivided into m chunks on which Eq. 4.5-4.7 can be computed in parallel to generate m local estimates $\mathtt{V}_{\mathtt{DB}_3}[i]$, with $i \in [1, m]$. In other words, each SDI member can independently apply the metric integration introduced in Sec. 4.2, producing its local estimate $\mathtt{V}_{\mathtt{DB}_3}[i]$. All these estimates can be subsequently combined together by the SDI manager to provide the global database estimate $\mathtt{V}_{\mathtt{DB}_3}$.

Starting from the estimates $\mathtt{V}_{\mathtt{DB}_3}[i]$, $\mathtt{C}_{\mathtt{DB}_3}[i]$ and $\mathtt{K}_3[i]$, computed by each SDI member on its local data using Eq. 4.5-4.7, the global estimate $\mathtt{V}_{\mathtt{DB}_3}$ can be determined by the SDI manager through the following formula, adapted from [55]:

$$\begin{aligned} \mathtt{V}_{\mathtt{DB}_3} = \mathtt{C}_{\mathtt{DB}_3} \cdot (\mathtt{C}_{\mathtt{DB}_1}^{-1} \cdot \mathtt{V}_{\mathtt{DB}_1} + \\ + \sum_{i=1}^{m} \left(\mathtt{C}_{\mathtt{DB}_3}^{-1}[i] \cdot \mathtt{V}_{\mathtt{DB}_3}[i] - \mathtt{C}_{\mathtt{DB}_1}^{-1}[i] \cdot \mathtt{V}_{\mathtt{DB}_1}[i] \right)) \end{aligned} \tag{4.11}$$

where $\mathtt{V}_{\mathtt{DB}_1}$ and $\mathtt{C}_{\mathtt{DB}_1}$ are the position index vector and the variance-covariance matrix of the previously computed database integration, while the global covariance matrix $\mathtt{C}_{\mathtt{DB}_3}$ is obtained from the local covariance estimates $\mathtt{C}_{\mathtt{DB}_3}[i]$ and the previous estimate $\mathtt{C}_{\mathtt{DB}_1}$ as follows:

$$\mathtt{C}_{\mathtt{DB}_3}^{-1} = \mathtt{C}_{\mathtt{DB}_1}^{-1} \cdot \sum_{i=1}^{m} \mathtt{C}_{\mathtt{DB}_3}^{-1}[i] \cdot \mathtt{C}_{\mathtt{DB}_1}^{-1}[i] \tag{4.12}$$

It has been show in [55] that the estimate $\mathtt{V}_{\mathtt{DB}_3}$ produced by Eq. 4.11 corresponds to the one produced by directly applying Eq. 4.6 to the global SDI database and the overall vector of new observations.

Therefore, as regards to metric observations, in the considered integration scenario each SDI member communicates its local estimate to the SDI manager, who is responsible to globally combine these estimates using Eq. 4.11-4.12. After the global estimate has been computed, the SDI manager is also responsible to notify the updates to all the SDI managers that have been affected by the integration. In the following section the problem of integrating logical observations in a distributed way is analyzed.

4.6.2 Distributed Integration of Logical Information

This section discusses how logical information can be integrated in the stated distributed context. More specifically, the proposed integration process is composed of three steps: the integration of metric information, the integration of logical information and the integration of metric and logical information together. The previous section analyzes how the first phase can be performed in a distributed way, namely how estimates computed by locally applying the Kalman integration can be combined together for correctly propagating the integration effects.

Relatively to logical information, it can be observed that the integration procedure introduces in Sec. 4.3 can be locally performed by each SDI member. Such procedure essentially merges the relations that are known in the two source databases and determines the other possible ones by using the confidence region of the objects among which no relations have been explicitly defined. The resulting relations can regard only objects that are contained in the local database, or pairs of objects such that one is contained in the local database and the other is contained into another one.

For the relations that involve only objects contained in the local database, each SDI member can also locally perform the third integration phase that removes the generated inconsistencies. In other words, in the database transferred by each SDI member to the SDI manager is free from logical inconsistencies.

Conversely, for relations regarding two adjacent datasets, the SDI member can only determine the resulting relations and transfer them to the SDI manager. The SDI manager will integrate the received relations with the ones originally contained in its global database. If this integration determines empty relations, the SDI members that own the involved spatial objects will be notified and their intervention is required for determining the final relation.

Finally, the SDI manager will perform the last integration phase on the dataset boundary, in order to verify the presence of inconsistencies between the determined topological relations and the integrated geometries. At the end of this stage the SDI manager identifies the portion of the database that has been changed, in terms of metric and logical information, and from it the SDI members that have to be notified.

4.7 Efficient Covariance Matrix Representation

As discussed in Sec. 3.1, the dimension of the covariance matrix can hinder the application of the technique, in particular in a distributed context. This section proposes an efficient representation of the covariance matrix that substantially reduces the quantity of information that has to be stored and transferred prior and after an integration.

Let us consider the scenario reported in Fig. 4.7: the violet polygons are new buildings contained in a portion of LDB that have to be integrated into a portion of RDB, the yellow and the overlapped green ones are the buildings and the street areas contained in both database portions, respectively. The initial global database in which the two new buildings have to be integrated is composed of 1243 points.

This means that the size of the covariance matrix to be transferred to the SDI member for the integration is 2486×2486, namely 6,180,196 values potentially encoded in double precision using 64 bits, for a total of at least 47 MB. Similarly, the two new buildings are represented using 71 points, hence the covariance matrix resulting from the integration contains 6,906,384 elements (54 MB) that have to be transferred back to the SDI manager.

Fig. 4.7. Situation before performing the integration: the orange (lighter) polygons are contained only in the considered portion of the global RDB database, while the violet (darker) polygons are the new buildings of the considered LDB portion to be integrated; finally, the yellow polygons and the overlying green ones are the buildings that the integration process considers as part of both the global and the local SDI database portions.

The size of a covariance matrix can be reduced by representing only the unique values inside it, together with the positions where these values are located. Notice that half of the values are zeros, and the variance of the x and y coordinates of each position coincides, while the covariance between the x coordinates of each pair of positions has the same value of the covariance between the y coordinates of the same pair of positions. Anyway, the number of distinct elements inside a covariance matrix can be further reduced by considering a threshold τ below which different covariance values are considered as the same. The choice of the threshold τ is critical in order to both reduce the amount of information to be stored and transferred, and control the effects of this approximation on subsequent operations. The idea is to represent the covariance values using a fixed point representation, where the adopted length is determined by the threshold τ. Greater is τ, smaller is the loss induced by this representation and greater remains the number of distinct values

inside the covariance matrix. Given this encoding, the matrix can be compressed by representing it as a list of zones characterized by the same value.

Table 4.11. Results of the covariance matrix compression experiments.

bits	$unique(C_0^a)$	$\Delta(C_0)$	*max distance*	*avg distance*	$unique(C_1)$	$unique(C_1^a)$	$\Delta(C_1)$
15	3	99.99%	8.791360	0.886661	1,275,763	6,954	99,72%
16	5	99.99%	2.678981	0.479957	2,056,498	28,582	98,83%
17	8	99.99%	2.520073	0.780865	2,412,541	44,700	98,18%
18	13	99.99%	1.160076	0.239304	2,555,241	67,158	97.26%
19	23	99.99%	0.102096	0.031367	2,487,269	118,104	95.18%
20	43	99.99%	0.059160	0.014984	2,471,040	197,914	91,93%
21	83	99.98%	0.029044	0.007157	2,449,771	300,705	87,73%
22	164	99.96%	0.011367	0.003272	2,441,795	403,541	83.54%
23	325	99.92%	0.006115	0.001643	2,446,224	483,474	80.28%
24	647	99.84%	0.003190	0.000851	2,455,008	535,180	78.17%
25	1,291	99.68%	0.001914	0.000461	2,443,862	564,681	76.97%
26	2,579	99.37%	0.000867	0.000233	2,448,969	580,246	76,33%
27	5,155	98.73%	0.000394	0.000108	2,472,271	588,600	75,99%
28	10,306	97.46%	0.000195	0.000054	2,454,520	592,710	75.82%
29	20,608	94.93%	0.000101	0.000027	2,467,701	595,124	75.72%
30	41,183	89.87%	0.000004	0.000013	2,441,109	596,440	75,67%
31	81,463	79.96%	0.000028	0.000006	2,464,989	597,639	75.62%
32	149,232	63.29%	0.000014	0.000004	2,506,776	597,985	72.93%

Table 4.11 summarizes the results of this encoding technique applied to the situation in Fig. 4.7. In particular, the initial global covariance matrix C_0 contains 406,546 unique values (on a total of 6,180,196 elements), this matrix has been approximated using a different number of bits. For each number of bits reported in the column *bits*, column $unique(C_0^a)$ contains the number of unique values in the approximated covariance matrix encoded using this number of bits, column $\Delta(C_0)$ is the obtained compression ratio, columns *max distance* and *avg distance* reports the maximum and average distance in meters between the points in the exact resulting database and the database obtained with the approximated covariance matrix, respectively; column $unique(C_1)$ reports the number of unique values in the exact integrated covariance matrix, while $unique(C_1^a)$ contains the number of unique values in the approximated integrated covariance matrix, finally $\Delta(C_1)$ is the compression ratio for the integrated covariance matrix. The compression ratio is defined as the difference between the number of unique values contained in the exact matrix and the number of unique values in the approximated matrix, divided by the number of unique values in the exact matrix.

The average distances in meters between the points in the integrated database computed using the exact covariance matrix and the corresponding points integrated with the approximated one are reported also in Fig. 4.8. Let us notice that using an encoding with less than 16 bits, the maximum and average distances are not negligible, while between 20 to 32 bits they are less than few centimeters.

The compression ratio for the original matrix C_0 and for the obtained one C_1 are represented in Fig. 4.9, it is evident that the compression is higher, even using

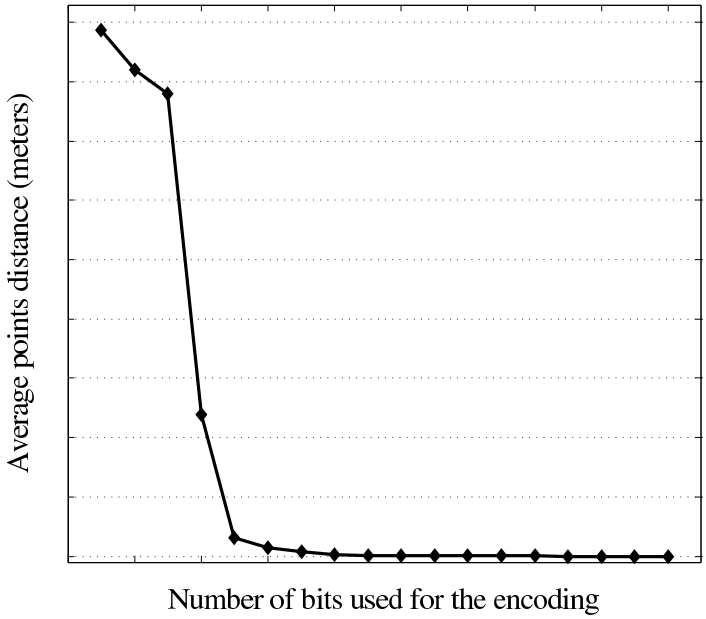


Fig. 4.8. Relation between the number of bits used for encoding the covariance matrix and the average distance between corresponding points integrated with the exact and approximated covariance matrix, respectively.

a greater number of bits, for the original matrix C_0, because it is assumed to be generated from only two accuracy metadata using the procedure in Eq. 3.3, while the integrated one is obtained combining the accuracy values of both integrated databases.

The effects of the local source database accuracy on the compression factor of the obtained integrated covariance matrix C_1 have been also analyzed. Fig. 4.10 summarizes the different compression ratios in two cases: the first one, represented by the blue solid line, assumes for the local source database LDB an estimated absolute error of 0.6 meters, and an estimated relative error of 0.4 meters within a distance d of 200 meters or of 0.8 meters otherwise. Conversely, the green dashed line represents the case when the local source database LDB is characterized by an estimated absolute error of 1.4 meters, and an estimated relative error of 0.8 meters within 200 meters or of 1.6 meters otherwise. It can be observed that with the most accurate hypothesis the compression factor is smaller: indeed, in this case the objects directly involved in the integration process become more accurate than in the other case, consequently their variance and covariance values in C_1 are subject to a greater change with respect to their original values and they are less likely affected by the τ parameter. A similar situation is also determined by changing the distance d used in the definition of the estimated relative error. Fig. 4.11 shows how the compression ratio of C_1 changes in two cases where the estimated relative error is defined by the same parameters, except for the distance d that is assumed of 200 meters in one case (solid blue line) and of 1200 meters in the other one (dashed green line). As you can notice, the compression ratio

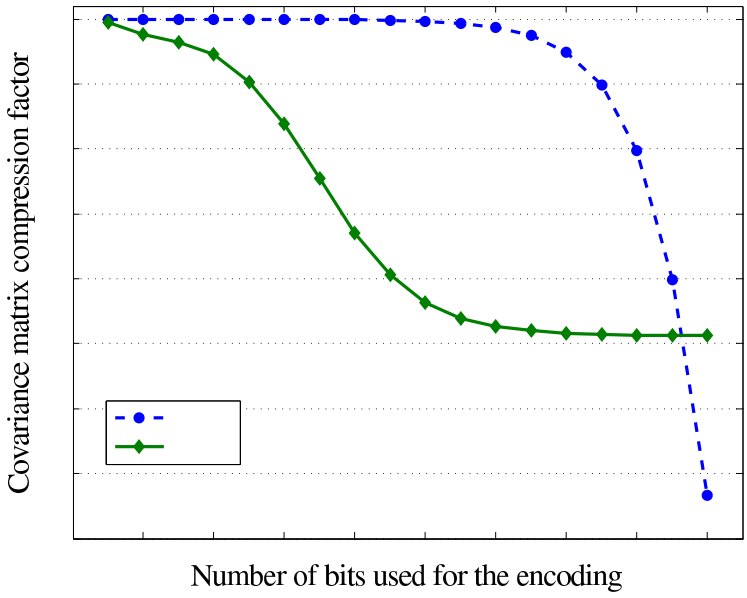


Fig. 4.9. Relation between the number of bits used for encoding the covariance matrix and the compression ratio of the original covariance matrix C_0 (dashed blue line) and of the integrated covariance matrix C_1 (solid green line).

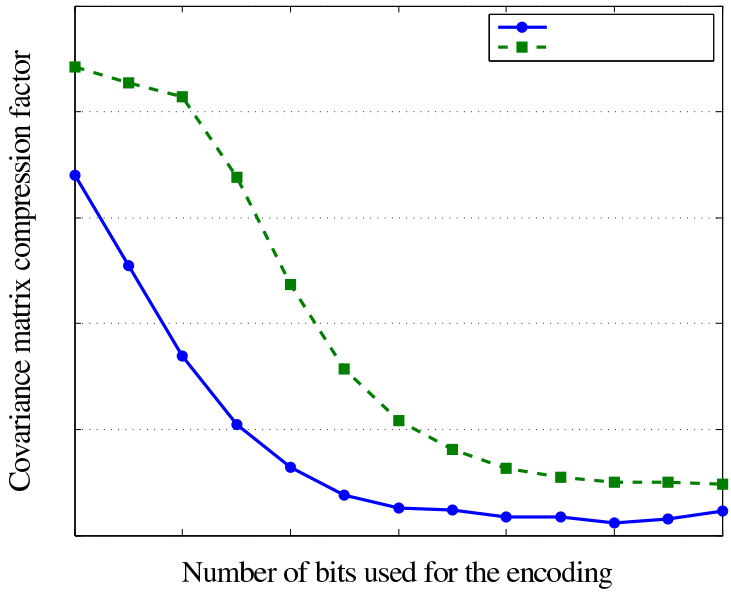


Fig. 4.10. Different compression ratios of the resulting covariance matrix C_1 considering different accuracy parameters of the local source database LDB.

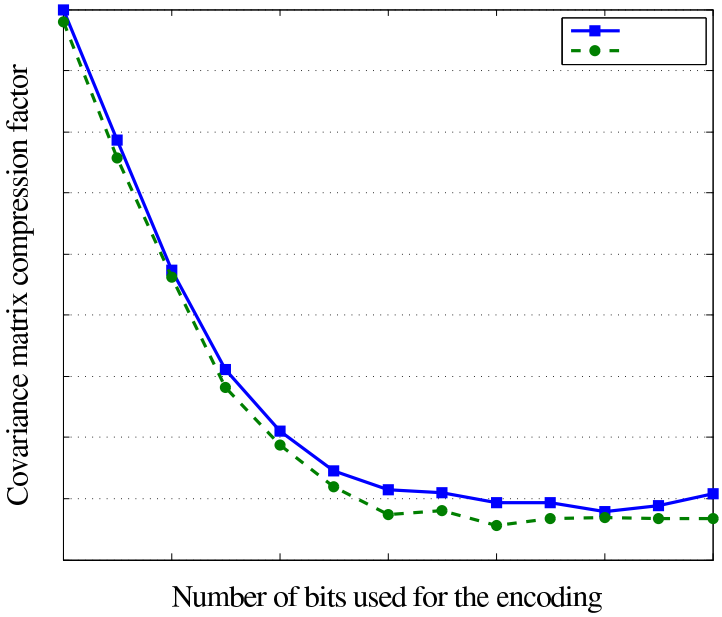


Fig. 4.11. Different compression ratios of the resulting covariance matrix C_1 considering different distances d used for the definition of the relative distance error in LDB.

is greater with the smallest value of d: the propagation of the integration effects increases as the parameter d increases, since d delimits the influence region of each object. As a consequence, with a smaller distance d, the number of variance and covariance values that are updated is smaller.

The proposed compression technique is based on the identification of homogeneous zones inside the covariance matrix. Thanks to the way in which the covariance matrix is computed and the meaning of its elements, it follows that these zones correspond also to geographically identifiable regions. Therefore, an alternative compact representation of the covariance matrix can be defined as a set of tuples $\{(\sigma_i^2, \sigma_i, R_i)\}_{i=1}^{n}$ where R_i is the region obtained as the union of the objects characterized by the same variance σ_i^2 and covariance σ_i values. For instance, in Fig. 4.12 is illustrated the distribution of the variance values after the integration, where the darker blue objects are those characterized by higher variance values (lower accuracy), while the lighter blue ones are those with lower variance values (higher accuracy). The method ensures that the objects directly involved in the integration process have increased their absolute accuracy (lower variance values), and that this absolute accuracy decreases as the distance to the updated objects increases. This compact representation of the variance-covariance matrix as zones with homogeneous accuracies can also be used for storage purposes, not only during transferring.

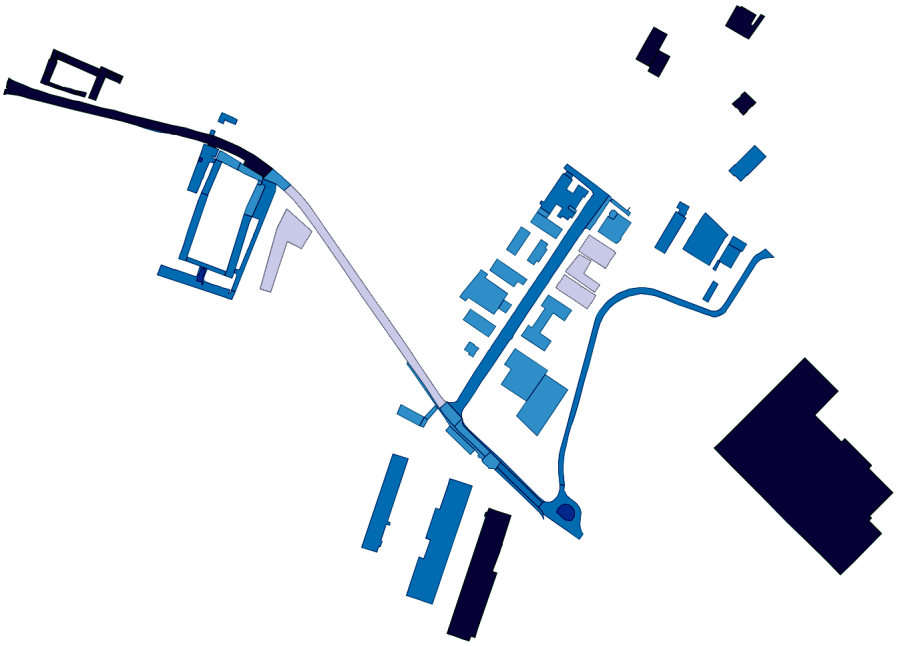


Fig. 4.12. Variance distribution after the integration process performed with a covariance matrix for the global RDB encoded using 20 bits. The darker blue objects are those with the higher variance values, while the lighter blue ones are those with the smaller variance values.

4.8 Integration Applied to a Real-World Case

This section validates the proposed integration technique against the real-world case presented in Sec. 1.2. The aim is to analyze how the various problems exposed in the introduction can solved by the proposed integration process. In particular, the main problems that can arise with an integration that does not consider the accuracies of the source databases can be summarized by the following points:

1. No information is available about the quality of an integrated database: when two databases characterized by different accuracies are integrated, what is the accuracy of the result? Surely, it cannot be the accuracy of any of the source databases. Moreover, if an object with a certain accuracy is inserted into a database characterized by a lower accuracy, what do we know about the accuracy of its relative distance with respect to the surrounding ones?
2. The integrated database can be very accurate in terms of shapes but it can become very inaccurate in terms of positions and relative distances.
3. Some of the topological relations that are known to be valid in the source database can be violated by the integrated result.

In the following sections, we recall the problematic situations highlighted in Sec. 1.2, regarding the construction of an Italian regional SDI in Lombardy. More specifically, we consider the presence of a global regional database containing less

accurate data, called RDB, and a local database containing more accurate and up-to-date information, called LDB, which is owned by a pilot municipality. During

	RDB	LDB
`abs_error`	3m	0.8m
`abs_fr`	95%	95%
`rel_err`	$\begin{cases} 1.4\text{m} + d/100 & \text{if } d < 400\text{m} \\ 1.8\text{m} & \text{otherwise} \end{cases}$	$\begin{cases} 0.6\text{m} + d/100 & \text{if } d < 600\text{m} \\ 1.2\text{m} & \text{otherwise} \end{cases}$
`rel_fr`	95%	95%

Table 4.12. Initial accuracy parameters for the two source databases.

the integration process the accuracy parameters reported in Tab. 4.12 are assumed for the two source datasets. The tests have been performed on a Intel 4 Core 3,30 GHz i5-2500K CPU, with 16 GB of RAM on a Windows 7 64-bit operating system.

The aim of this chapter is to illustrate how the found issues can be solved by the proposed integration technique. Notice that in order to correctly perform the integration and allow the propagation of its effects, the updated database LDB is assumed to always share some positions with RDB, namely in case a new object has to be inserted into the global database, some accuracy metadata about it and some existing surrounding objects are known. Some details about the process implementation will be given in Chap. 7.

4.8.1 Role of Accuracy During Metric Integration

This section illustrates the role played by accuracy metadata during the integration of metric observations. In particular, we recall the situation presented in Ex. 1.1

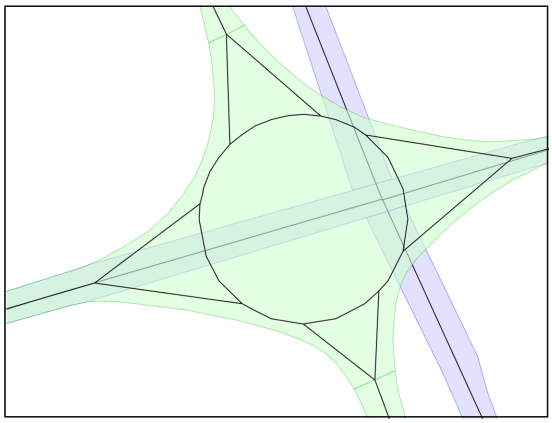


Fig. 4.13. Situation before performing the integration: the violet polygons represents the content of RDB, while the green ones are the content of LDB.

and depicted in Fig. 4.13: the blue polygons are the content of RDB, while the green ones are contained in LDB. We assume that the horizontal street is owned by the local municipality which is responsible for collecting its updates, while the vertical street is under the region competency. During the years the cross between the municipality and the regional street has been replaced by a roundabout. This change has been stored into the local database LDB, and has to be integrated with the content of the regional database. In order to perform this integration, we assume that some information about the relative distance between the roundabout and the regional street are contained in LDB. This information is represented by the shared vertical polygons connected to the roundabout which are very accurate in terms of relative distances, but very inaccurate in terms of absolute positions with respect to their representation in RDB.

As mentioned in Ex. 1.1, if the new roundabout is integrated in RDB by simply placing the new geometries into the old ones, the regional street becomes disconnected by the new roundabout. The result of the integration performed with the proposed approach is illustrated in Fig. 4.14.a. Since LDB is assumed more accurate than RDB, the position of the resulting geometry should not differ much from the position of LDB. Conversely, if the integration is performed by considering RDB more accurate than LDB, the result is the one reported in Fig. 4.14.b: in this case the position of the resulting geometry is near the one of RDB.

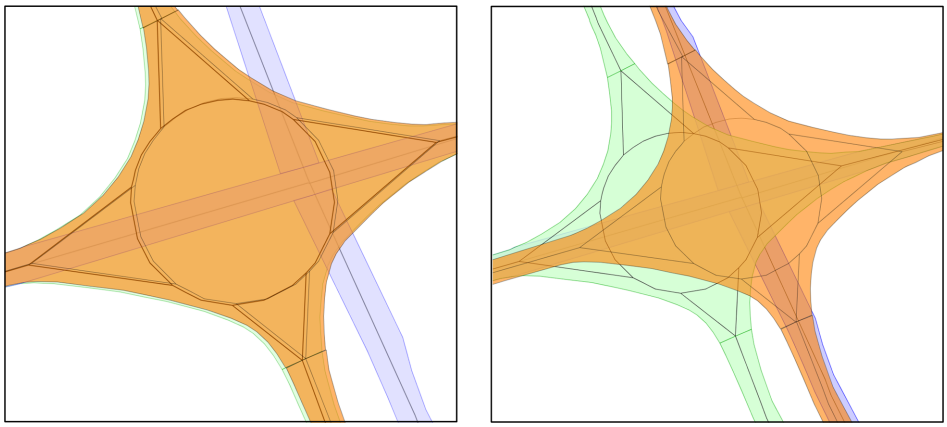


Fig. 4.14. The result of the integration process: (a) considering LDB more accurate than RDB, and (b) considering the inverse situation.

Tab. 4.13 reports some metadata about the source and the integrate databases; in particular, the metric accuracy estimator acc_m of each database is highlighted. As you can notice, the accuracy of the resulting database is greater than both the source databases.

	RDB	LDB	RDB_{result}
Number of features	5	25	25
Number of positons	50	240	240
Covariance matrix dimension	(100×100)	(480×480)	(480×480)
Metric accuracy acc_m	0.87	3.25	3.34

Table 4.13. Some information about the source and the integrated databases contained in Fig. 4.13 and Fig. 4.14, respectively.

4.8.2 Preservation of Relative Distances

This section considers the issue presented in Ex. 1.2 about the preservation of relative distances. In the considered example, a new building has been collected by LDB and has to be added into a portion of RDB, we assume that in LDB some relative distance information is available between the new building and a known reference point representing a bus stop, and that the same is true in RDB between the same reference point and an existing building. Notice that the reference point is present in both databases, but with slightly different positions. A naïve integration that simply places the new building into RDB will produce a database that is accurate in terms of shapes, but inaccurate in terms of relative distances. Indeed, as mentioned in the introduction the distance between the new building and the reference point can vary from about 7-8 meters.

Fig. 4.15 illustrates the content of the two source databases. In particular, the content of RDB includes the green polygons, the yellow street areas and the black reference point inside the horizontal street area; while the content of LDB includes the light red big polygon and the reference point outside the horizontal street area.

	RDB	LDB	RDB_{result}
Number of features	9	5	10
Number of positons	170	681	721
Covariance matrix dimension	(340×340)	(1362×1362)	(1442×1442)

Table 4.14. Some information about the source and the integrated databases represented in Fig. 4.15 and Fig. 4.16, respectively.

The result of the integration performed using the proposed integration method is illustrated in Fig. 4.16. As you can notice, the relative distance between all buildings and the horizontal street areas are substantially preserved after the integration. Indeed, the position of the new building has been slightly modified in order to preserve the relative distance, this is determined by the fact that the accuracy of absolute positions is less than the accuracy of relative distances. Tab. 4.14 reports some information about the involved databases.

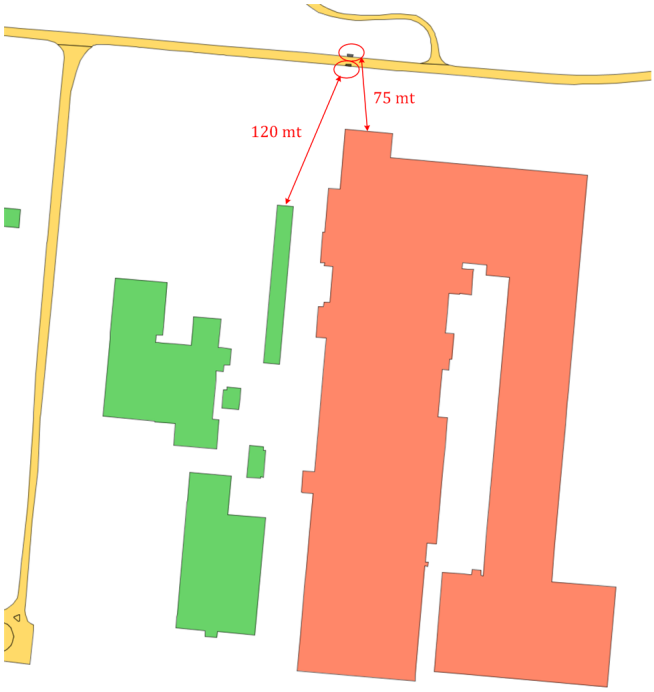


Fig. 4.15. Initial situation before applying the integration procedure: RDB contains the green buildings, the yellow street areas, and the black reference point inside the horizontal street area, while LDB contains the light red big building and the black reference point outside the horizontal street area.

4.8.3 Preservation of Topological Relations

This section treats the problem of the preserving the topological relations during the integration process. First of all, we consider the case in Ex. 1.3 where a new building labeled new_A has to be inserted into RDB. The content of the original datasets is illustrated in Fig. 4.17: the orange and yellow polygons represent the content of RDB, while the violet and the green ones are the content of LDB. In particular, the violet polygon is the new building that has to inserted into RDB, while the surrounding green buildings are the content shared by the two databases. If the new building is overlaid on RDB, an overlap is obtained between it and the right orange polygon, This is due to the different accuracies of the two databases, indeed in LDB the position of this neighbour building is quite different.

The result obtained using the proposed integration technique is reported in Fig. 4.18: the new building has been perfectly inserted between the two surrounding ones. In particular, thanks to the role covered by the accuracy of the relative distance between the new building and the two near ones, the disjoint relation has been preserved. In particular, this effect is due to the presence of the two shared buildings in LDB. Tab. 4.15 reports some information about the two source databases and the integrated one.

Fig. 4.16. The result of the integration performed with the proposed technique: the relative distance between all the polygons with respect to the horizontal street area are substantially preserved.

	RDB	LDB	RDB_{result}
Number of features	84	5	86
Number of positons	1209	71	1222
Covariance matrix dimension	(2418×2418)	(142×142)	(2444×2444)

Table 4.15. Some information about the source and the integrated databases represented in Fig. 4.17 and Fig. 4.18, respectively.

A similar situation is depicted in Fig. 4.19: some new polygons, depicted in green and labeled as new_A-new_F, have to be inserted into RDB, whose content is

	RDB	LDB	RDB_{result}
Number of features	216	9	222
Number of positons	2825	261	2934
Covariance matrix dimension	(5650×5650)	(522×522)	(5868×5868)

Table 4.16. Some information about the source and the integrated databases represented in Fig. 4.19 and Fig. 4.20, respectively.

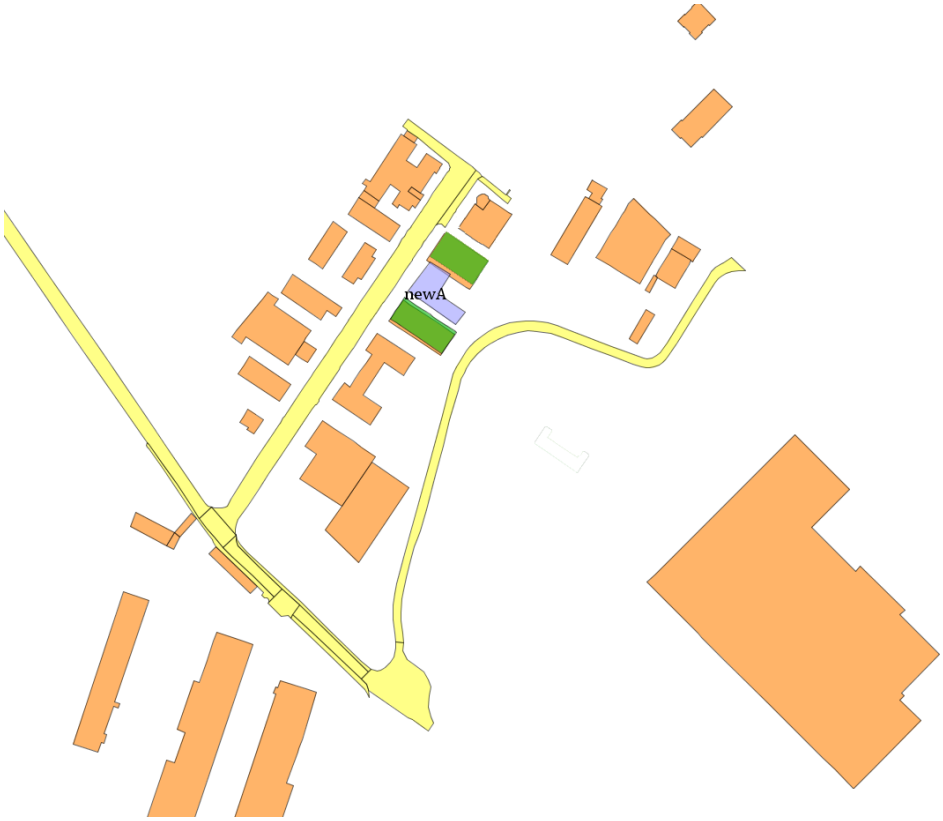


Fig. 4.17. Content of the two source databases: the orange polygons represent the buildings contained in RDB, while the yellow polygons are its street areas, the violet polygon is the new building of LDB that has to be inserted into RDB, and finally the green polygons are the buildings that LDB shares with the global database.

represented by the yellow and orange polygons. In particular, for correctly performing the integration process, the measurements of some street areas of RDB have performed also in LDB, they are identified by the violet polygons.

If the new buildings are simply placed into RDB, some of them will overlap an existing neighbour street area. This is due to the different accuracies of the two source databases. Conversely, by applying the proposed integration process, we obtain the result depicted in Fig. 4.20.

4.9 Summary and Concluding Remarks

This chapter has discussed the problem of integrating spatial data coming from different sources and characterized by different positional accuracies. The spatial data integration problem has become an important research field in recent years, due to the development of SDIs in many different countries and the need of orga-

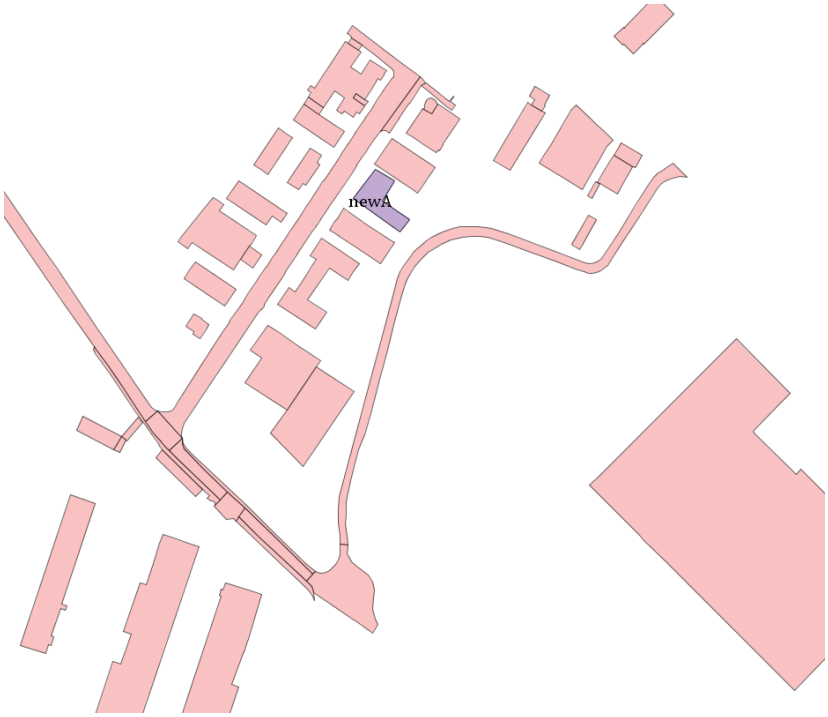


Fig. 4.18. The result of the integration performed with the proposed technique: the new building labeled with new_A is now perfectly inserted between the two surrounding buildings: thanks to the role covered by the accuracy of relative distances, the disjoint relation has been preserved.

nizations to share not only data, but also tools, processes and competencies in an effective way. In literature many research efforts deal with the problem known as data integration or information fusion. In particular, such activity requires different phases that range from the identification of similar concepts to the integration of entire schemas. In the geographical field, an important activity regards the feature or point matching and the alignment of different geometries. The main difficulty in performing this activity is induced by the inherent inaccuracy of spatial data. A certain amount of error in the representation of spatial data always exists, because the measurements needed to survey the shape, extension and position of objects with the maximum accuracy are often too expensive, or because the maximum accuracy is not necessary to satisfy the application requirements. Available integration techniques usually align the dataset with lower accuracy to the more accurate one, assuming that the last one is correct. In this way, corresponding features in the two datasets are aligned but in a sub-optimal manner. Moreover, no updated quality information are provided for the adjusted dataset.

In this chapter we have considered the context of an highly coupled SDI where members have adhered to a common global schema. Therefore, we assume that a previous merging operations have already been performed on the local schemas, and we concentrate on the second aspect of the integration problem regarding the

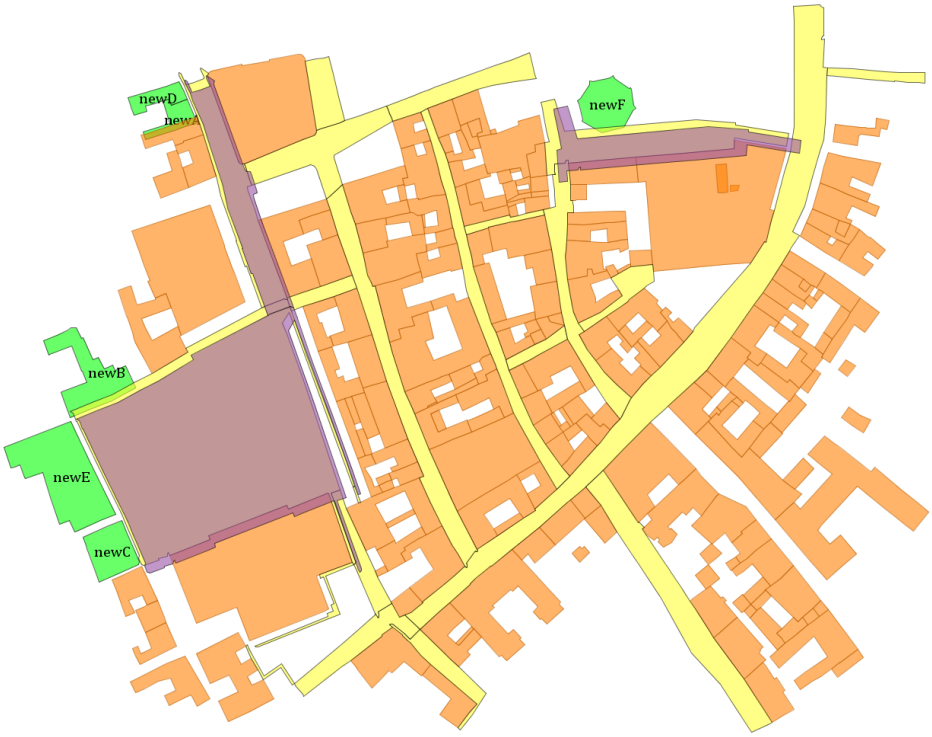


Fig. 4.19. Initial situation before the integration: the yellow and orange polygons represent the content of RDB, the green polygons are the new buildings of LDB that have to be inserted into RDB, while the violet ones are the shared street areas contained in LDB.

integration of geometries characterized by different quality levels. Moreover, we consider the most frequent case in which only derived coordinates are available, together with just few metadata about the accuracy of absolute positions and relative distances.

Metric and logical observations are stored together into a MACS database and they are jointly considered during the integration process. More specifically, such process can be decomposed into three phases: the first one combines together metric information contained in the two source databases, using an application of the Kalman filter, the second phase regards the integration of logical observations, and finally the last phase puts together the obtained metric and logical information, eventually solving inconsistencies. This integration procedure takes care of the accuracy of both source databases and produces also updated quality information for the resultant one.

In order to make feasible the application of the proposed framework in a distributed context, even in presence of a huge amount of data, a distributed version of the integration process has been introduced. Such version allows SDI members to locally perform the integration of the new data at their disposal, and then to communicate the updated information to a central SDI manager which is respon-

Fig. 4.20. The result of the integration performed with the proposed technique: the new buildings labeled with new_A-new_F are now perfectly integrated with the surrounding street areas: thanks to the role covered by the accuracy of relative distances, the disjoint relations have been preserved.

sible to propagate the integration effects on the neighbour datasets and notify the other involved SDI members. One of the main issues of the proposed technique resides on the dimension of the variance-covariance matrix that has to be stored and managed during the computation. In this respect, a technique is proposed for compressing such matrix and the effects of this compression on the following integrations are discussed.

Finally, some properties of the proposed integration procedure and its application on some real-world cases have been described. In particular, this application concentrates on the issues highlighted in the introduction, regarding the integration of spatial data without considering their accuracy, and it shows how they can be solved by the proposed framework.

In the following chapters the applicability of workflow technologies is evaluated for supporting distributed geographical processes with the characteristics described in the introduction. In particular, the integration process presented in this chapter will be used as motivating example through Part II. Indeed, it can become a distributed application that requires the coordination and collaboration of several agents with different competencies, and in which not all the operations can be automated. This last part also discusses some implementation details of the proposed integration technique.

Part II

Geo-Processing in Practice

5

Background: Workflow Management Systems

The realization of the integration framework proposed in the previous chapter can be very challenging, not only for the complexity of the involved operations, but also for the need to coordinate different and distributed agents in performing such operations. For this reason, the second part of the thesis investigates the applicability of workflow technologies for the realization of the proposed framework. Workflow technologies are increasingly applied for building new information systems, especially when these systems cross the boundary of a single organization. The use of this technology eases the reuse and integration of existing applications, facilitates the deployment of new applications and increases their flexibility, lowering the efforts required to adapt the system to future contingent needs.

Given the characteristics of the proposed integration framework, it seams reasonable to exploit existing workflow technologies for implementing it. The aim of this chapter is twofold: (1) giving a brief overview of the methods and the technological infrastructure underpinning workflow technology, and (2) introducing some real WfMSs in order to expose their strengths and weaknesses. At this regards, since it is not possible to consider all existing WfMSs, in this chapter we concentrate on some well-known representatives, which capture the essential solutions. The structure of the chapter is as follows: Sec. 5.1 introduces the business process management methodology, while Sec. 5.2 discusses the main characteristics of a service oriented architecture, and Sec. 5.3 clarifies the different meanings underlying the terms workflow. Finally, Sec. 5.4 and Sec. 5.5 presents two main paradigms in the design of processes, known as control-flow and data-flow oriented, while Sec. 6.2 introduces some existing offerings in the geographical domain.

Starting from this classification, the following chapter evaluates the possibility of using workflow technologies for supporting distributed geo-processing. In particular, the considered WfMSs are compared from a general point of view, using the well-known workflow patterns [5], and then from a geographical point of view, considering the implementation of the proposed integration process as case study. Workflow patterns have been extensively used for evaluating existing workflow systems; hence, they can be easily used as a comparison mean among different offerings. Conversely, the proposed integration process is a good representative for geographical processes, since it presents many of the characteristics highlighted in the introduction.

5.1 Business Process Management

An organization can be understood in term of its relevant processes and this is also true for a virtual organization, like an SDI. A *Business Process* (BP) is a collection of interrelated activities that are coordinately performed inside an organization for achieving a predefined goal [141]. In order to achieve its goals and fulfill the stackeholders' needs, any organization have to *coordinate* the interaction of different agents. This coordination is obtained exchanging information through a more or less automated *information system*.

Business Process Management (BPM) is an approach to describe and manage an organization in terms of its processes. The main goal of BPM is understanding how an organization operates, which activities it is performing and the relationships among them. The explicit documentation of BPs allows people inside an organization to share a common vision on how the organization works, improving coordination. Ideally, an organization is aware of which BPs are currently running and how they are related. An organization systematically measures the performance of its internal processes with the aim of continuously improve them: new BPs are carefully designed and inefficient ones are discarded or re-engineered. Essentially, the BPM approach can be seen as a stable process for improving processes, opposed to occasional adjustments due to contingent needs. The phases of this process are reported in Fig. 5.1 and are usually denoted as the *BP lifecycle*. The *design and analysis* phase includes the analysis of existing processes and the

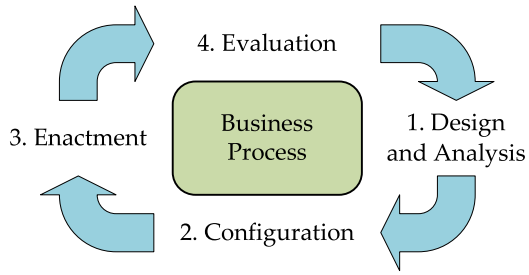


Fig. 5.1. BP lifecycle adapted from [141]

design of desired new ones. Processes are explicitly represented through one or more BP models which are usually expressed by means of a predefined graphical modeling language, such as YAWL and BPMN that are described in the following sections. In the *configuration* phase, BP models are enhanced with additional information that facilitates its automatic execution through a computerized system. After the configuration phase a BP is ready to be executed (*enactment*). The execution starts when a particular event occurs, for example a new request is received from a customer or a stakeholder. A BP is continuously monitored and data about its execution is recorded for evaluating effectiveness and performance parameters (*evaluation*).

BPs can be classified with respect to different aspects [141]. For instance, as regards to the *degree of automation*, we can distinguish between fully automated

processes and processes that require human intervention during their execution. A BP is usually carried out by human agents, hence their interaction with the software system is a primary concern. Moreover, with respect to the *degree of repetition*, BPs can be also classified in production and collaborative processes. Production processes are usually well-structured and highly repetitive, in general these systems prescribe the activities and their execution constraints in a complete fashion. They are executed hundred of times a day and they are less sensitive to the used implementation technology, because they are relatively stable and their usage for long time can pay-off the initial investment for an ad-hoc solution. On the opposite side, collaborative processes occur only few times before changing, such as large engineering efforts or scientific experiments: for them it is fundamental to carefully track their current state and execution constraints, in order to ensure controllability and/or reproducibility. For this kind of processes, that are characterized by long-running executions and a low number of instances, the applied technology becomes crucial.

The development of an SDI allows to create a virtual organization characterized by a set of processes that spawn multiple existing organizations. These processes present some distinctive characteristics with respect to those typically performed inside an organization. In particular, they can involve sophisticated elaborations on huge amount of spatial data. Due to the nature of an SDI and the high investment needed to develop it, it is reasonable to assume that simple processes are automated or will be automated with specific optimized software systems. The real challenge in which we are interested is how to build reliable and flexible applications to support collaborative processes. The integration process proposed in this thesis is a clear example of this kind of processes.

5.2 Service Oriented Architecture (SOA)

Organizations can exploit different type of software systems for supporting their activities and automating their information systems. The development of a new

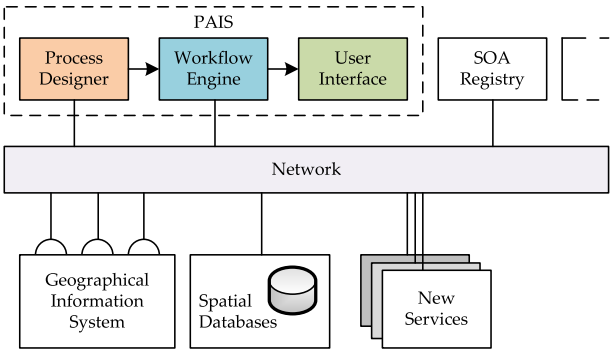


Fig. 5.2. High-level view of a SOA implementation.

information system rarely starts from scratch: old software systems cannot be

completely replaced with new ones, due to the high costs induced by both the new software realization and the training of people inside an organization that have to use it. This condition is particularly true in the geographical context, where the amount of available systems, tools and libraries, besides the knowledge needed to use them, make impractical the development of a completely new system.

Two primary forces drive the introduction of new information technologies: the need to integrate existing systems for raising automation and the need for more flexible systems for lowering change efforts: a service oriented architecture try to meet these needs. The *Service Oriented Architecture* (SOA) is a software architecture based on loosely coupled autonomous components, called services. A *service* usually implements a new reusable function or exposes a function of an existing software system. A service provides a well defined interface that does not depend on the implementation language; this interface is enriched with self-describing metadata and published in a SOA registry to be automatically discovered and invoked across the network. Services are basic blocks that can be combined together for implementing distributed applications.

Service is often synonymous of web-service, because SOA technologies are mostly driven by web-oriented standards; indeed, they are a well suited tool for overcoming the organization boundaries. A *web-service* represent a particular kind of software component which is accessible through the web and executed on a remote system.

The use of SOA has been encouraged also in the geographical field by the OGC [4] through the development of a series of standards for spatial data sharing and processing. In particular, the Web Processing Service (WPS) interface specification [99] promotes the constructions of standardized services for performing web-based processing of spatial data. The main characteristics of OGC WPS will be discussed in Sec. 5.6.1.

5.3 Workflow Management Systems

The term WfMS is commonly abused and actually can denote very different kinds of systems. For the aims of this thesis, four type of WfMSs can be distinguished: desktop, server, business and scientific.

Desktop WfMSs – They denote the automation of a set of repetitive steps or operations inside a particular application, as happens in office automation. This kind of workflows is not considered any further in this thesis, because it offers limited functionalities and cannot meet the requirement of a distributed information system.

Server Workflows – They are programs obtained combining existing services, for instance using the Business Process Modeling Language (BPEL) [97]. The construction of these programs is usually supported by the use of an *application server* which provides a set of functionalities that allow the easy composition of new software components. More specifically, an application server is a software which provides facilities such as security management, data services, transaction

support, load balancing, and management of large distributed systems. Examples of Java-based application servers are JBoss Applicaton Server (JBoss AS) [3] and Oracle GlassFish [1].

The low-level functionalities provided by an application server are usually exploited through high-level languages and tools built upon them, that are often classified as WfMSs. However, their purpose is to ease the application development and the user is not aware of the underlying defined processes. This is in contrast with the following two kinds of WfMSs, whose primary aim is to document the processes inside an organization.

Process-Aware Information Systems – *Business* WfMSs or *Process-Aware Information Systems* (PAISs) are software systems driven by explicit process specifications, with the aim to coordinate the involved agents in performing their activities. The main goal of a PAIS is to enact a process, manage human resources, and trace the processes execution, by interpreting a given process specification. Process specifications managed by PAISs usually focus on the identification of the involved activities and the order dependencies among them, hence these systems are classified as *control-flow oriented*.

The usual architecture of a PAIS is the one described in Fig. 5.3, where rectangular boxes represent software components and the oval ones denote data. This architecture includes a BP Designer, through which a process model can be defined, but also a BP Engine, which interprets the specification an generates a minimal user interface for coordinating and monitoring the process execution. This architecture reveals that not all software systems that take advantage of a graphical language can be considered a PAIS, including even some WfMSs.

In this thesis we are interested in PAISs that are able to implement interactive software systems from the provided process specifications. Many open-source and commercial offerings are available, a good example of such architecture is the open source YAWL System [127].

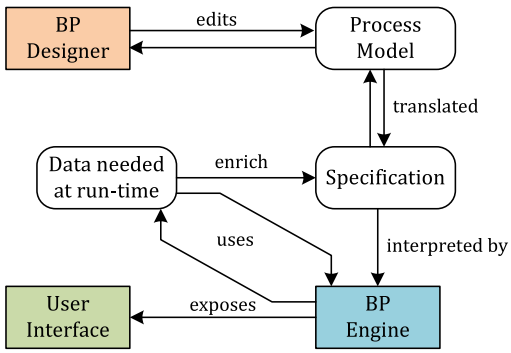


Fig. 5.3. PAIS architecture.

Scientific WfMSs – They are software systems developed for automating large-scale scientific experiments. The main goals of scientific workflows are reusing

domain specific functions and tools, and supporting their easy integration through a graphical environment, ideally allowing domain experts to define sophisticated analysis (experiments) without any programming knowledge. Moreover, the nature of the defined computations may require the scheduling on cluster of computers or remote resources (Grid computing); therefore, many scientific WfMSs provides a support for transparently executing their tasks on a Grid environment.

Unlike business workflows, scientific workflows are usually classified as *data-flow oriented*: data drive the computation and relationships among tasks are determined by data dependencies: a task can execute only and any time the necessary input is available. This characteristic allows them to support an implicit form of data parallelism: by subdividing the input data into independent chunks the same task is automatically performed in parallel many times, depending on the amount of currently available computational resources.

Examples of scientific workflows are: Kepler [81], Taverna [98] and Triana [83]. These systems are not easy to compare, since each of them provides specialized functionalities for a particular scientific domain.

For the implementation of the integration process, this thesis concentrates only the last two kinds of WfMSs. In particular, it takes as representative of PAISs the YAWL System [127], while as representative of Scientific WfMSs the Kepler System [81]. This choice has been justified by the fact that the YAWL System has born after an exhaustive analysis of the existing systems, in order to show how workflow patterns [5] can be supported in a coherent way. Conversely, Kepler captures many concepts underlying data-flow systems and at now it seems to be the most mature and complete open source scientific WfMSs [88]; hence, it can be considered a good representative of this kind of systems.

5.4 Control-Flow Oriented Modeling Languages

A Control-Flow Oriented (CFO) modeling language describes a process by explicitly representing the execution order of the underlying process components, leaving data dependencies almost implicit. The execution order is partially determined at design-time by a control-flow relation and partially depends on the state in which specific language constructs are evaluated. The process state is modeled with a set of variables and one or more threads of control.

A CFO diagram is a directed graph with at least an initial and a final vertex marked with specific start and end constructs. The remaining vertices are preexisting component occurrences mixed with predefined language constructs, such as *choice*, *merge*, *fork* and *join*, collectively called routing constructs. Two vertices of the diagram are connected by an edge representing a control-flow dependency.

The execution of a CFO diagram can be informally explained in terms of tokens that flow inside a graph, resembling what happens in Petri Nets theory [91]. A token is produced by the start element when the process starts, while the end element consumes all the tokens that reach it. When a token reaches a choice, it is copied in only one of its outgoing branches depending on the evaluation of a

specified condition. A tokens that reaches a fork is duplicated on each outgoing branches. A token that reaches a join, waits until another tokens arrives in the other incoming branches before continue. Finally, when a token reaches a merge element, it is directly copied on the outgoing branch. Examples of CFO diagrams will be discussed in the following when some representative languages are introduced.

The CFO paradigm has been historically adopted for designing process modeling languages (PMLs), becuase it is apparently simpler than DFO one, at least because data aspects can be ignored at early design stages. This over simplification is paid at later design stages and during implementation, when data-flow relations shall be finally integrated in order to obtain a working software system.

5.4.1 Web-Services Business Process Execution Language (BPEL)

The *Business Process Execution Language* [94], also known as Web Services BPEL (WS-BPEL), is a standard executable language for specifying workflows through the composition of web-services.

WS-BPEL is classified as an *orchestration* language, namely it allows to specify an executable process that involves some message exchanges with other processes, such that the message exchange sequences are controlled by the orchestration designer. This is in contrast with *choreography* languages, such as the Web Service Choreography (WS-Choreography) language [137], which specifies a protocol for peer-to-peer interactions with the purpose to guarantee interoperability, for instance by defining some examples of legal sequences of message exchanges. In other words, orchestration languages concentrate on the presence of a central control entity which determines the behaviour of the system, while choreography languages refer to a distributed system which operates according to rules without a centralized control.

The following is an example of WS-BPEL workflow, taken from [94]. The process `HelloWorld` receives from another process a name, and then prints a message containing this name. More specifically, the section `partnerLinks` specify a connection between the current process and another one of type `Greeter-Caller`; the section `variables` holds two variables, one will be used to store the incoming message, and the other to store the outgoing message. Finally, the section `sequence` specifies the sequence of operations that the process will perform: (1) it receives a message that activate the process, (2) it compose a string using the received message, (3) it replies by sending back a greeting string.

```
<process name="HelloWorld" ...>
  <partnerLinks>
    <partnerLink name="caller"
                 partnerLinkType="tns:Greeter-Caller" myRole="Greeter" />
  </partnerLinks>

  <variables>
    <variable name="request" messageType="tns:nameMessage" />
    <variable name="response" messageType="tns:greetingMessage" />
  </variables>

  <sequence name="MainSeq">
```

```
    <receive name="ReceiveName" operation="sayHello"
             partnerLink="caller" portType="tns:Greeter"
             variable="request" createInstance="yes" />
    <assign name="ComposeGreeting">
      <copy>
        <from
          expression=
             "concat('Hello, ',
                      bpel:getVariableData('request', 'name'), '!')" />
        <to variable="response" part="greeting" />
      </copy>
    </assign>
    <reply name="SendGreeting" operation="sayHello" partnerLink="caller"
           portType="tns:Greeter" variable="response" />
  </sequence>
</process>
```

The creation of a process instance in WS-BPEL is always implicit; activities that receive messages can be annotated with the attribute `createInstance="yes"`, in order to indicate that its occurrence causes the creation of a new workflow instance, as in the previous example.

The constructs provided by WS-BPEL are almost structured. In particular, they allow to specify sequence, parallel and conditional executions, besides to send and receive messages or handle exceptions.

In WS-BPEL outgoing and incoming messages are described using the Web Service Description Language (WSDL). A WSDL definition allows one to specify port types offered by a service, the connections between services, and so on. WS-BPEL workflows represent *stateful long-running interactions* in which each interaction has a beginning, a defined behaviour during its lifetime, and an end. For instance, a WS-BPEL process describing a purchase order starts with a request from the customer and ends with the shipping notices and invoices.

WS-BPEL is essentially an XML-based language for which there is not a standard graphical notation. Some proposes exist in literature for adopting the Business Process Modeling Notation (BPMN) as a graphical representation of WS-BPEL constructs. As an illustration of the feasibility of this approach, the BPMN specification includes an informal and partial mapping [143] from BPMN to BPEL 1.1. However, more specific investigations on this field have exposed fundamental differences between the two languages, which make the translation difficult and in some cases impossible [101,139]. Even more difficult is the problem of maintaining synchronized a BPML graphical model and the generated BPEL code, in the sense of propagating any modification from one specification to the other, and viceversa.

5.4.2 Business Process Modeling Notation (BPMN)

The *Business Process Modeling Notation* (BPMN) is a standard graphical language for specifying business process. It is based on a flowchart technique similar to the Activity Diagrams [96] of the Unified Modeling Language (UML).

BPMN models consist of diagrams built using a predefined set of graphical elements, depicted in Fig. 5.4, that can be classified into four main categories:

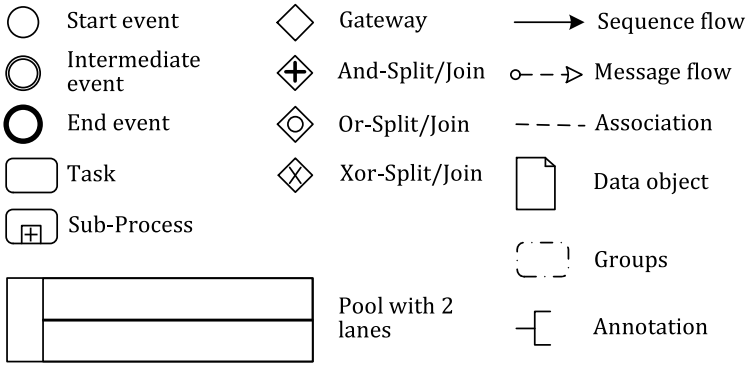


Fig. 5.4. Basic BPMN constructs.

- *Flow elements* – they are the main elements.
  - *Event*: an event is represented as a circle and denotes something that happens. In particular, *start events* act as process triggers, i.e. catching an incoming message starts a process instance, *end events* represent the result of a process instance execution, and *intermediate events* represent something that happens between a start and end event.
  - *Activity*: an activity is represented through a rounded-corner rectangle and describe the kind of work that has to be done. In particular, it can be a *task*, namely a single indivisible unit of work, a *sub-process*, denoted by a plus symbol inside the box, or a *transaction*, a particular kind of sub-process in which all the contained activities have to be executed as a whole.
  - *Gateway:* a gateway is represented as a diamond shape and denotes forking or merging paths, depending on the expressed condition.
- *Connecting elements* – they connect together flow elements.
  - *Sequence flow*: it is represented as a solid line ending with an arrow. It shows in which order the activities are performed. A sequence flow can also start from a gateway element, in this case a condition can be specified on each flow, for determining if it is enabled or not, while a slash symbol denotes the default flow.
  - *Message flow*: it is represented as a dashed line with an open circle at the start and an open arrowhead at the end. It defines a flow of information across organization boundaries. Let us notice that a message flow can never be used to connect activities or events in the same process.
  - *Association*: it is represented by using a dotted line and defines an association between an artifact or text, and a flow element.
- *Swim lines* – they are visual mechanisms to organize and categorize activities.
  - *Pool*: it represents major participants in a process, typically by separating different organizations. A pool can contain one or more lanes.
  - *Lane*: it is used to organize activities inside a pool according to function or role. It is depicted as a rectangle stretching the width or height of the pool.
- *Artifacts* – they allow the designer to specify some more information.
  - *Data object*: it shows which data is required or produced by an activity.

- *Group*: it is represented with a rounded-corner rectangle and dashed lines. It is used to visually group some activities but it has no effects on the model.
- *Annotation*: it allows one to insert some useful comments.

Example 5.1 (Order fulfillment). Fig. 5.5 depicts an example of BPMN process regarding the fulfillment of an order. After the order has been received, a condition is evaluated about its acceptance. If the order is rejected, it is immediately closed and the process completed. Otherwise, the process is filled and two activities are

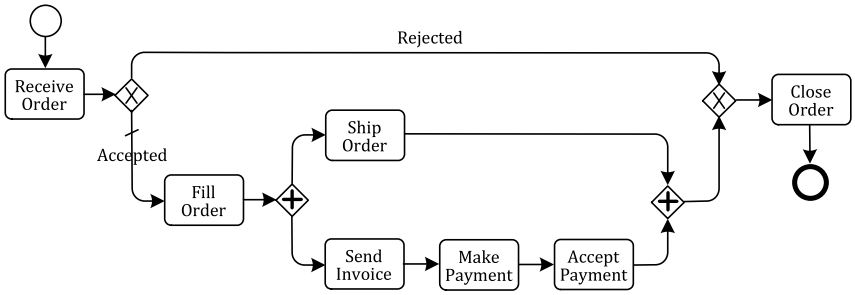


Fig. 5.5. Example of BPMN process.

performed in parallel: one related to the shipment and one related to the payment. When the order has been shipped and the payment accepted the order can be closed and the process completed.

5.4.3 The YAWL System

YAWL System [127] is an open source WfMS based on the *Yet Another Workflow Language* (YAWL) modeling language. It is born as an academic research project of the BPM Research Group from the formal analysis of the existing offerings.

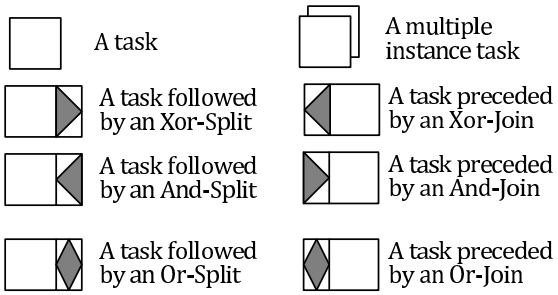


Fig. 5.6. Basic YAWL constructs.

This system has been chosen as representative for PAISs, because it provides a clearly stated semantics and captures many of the workflow control-flow patterns [109], data patterns [108] and resource patterns [111] in a coherent system.

Moreover, through the Workflow Patterns initiative [5], it has been compared against many other existing WfMSs. Sec. 6.1 presents a comparison in terms of workflow patterns among WS-BPEL, BPMN, YAWL and Kepler.

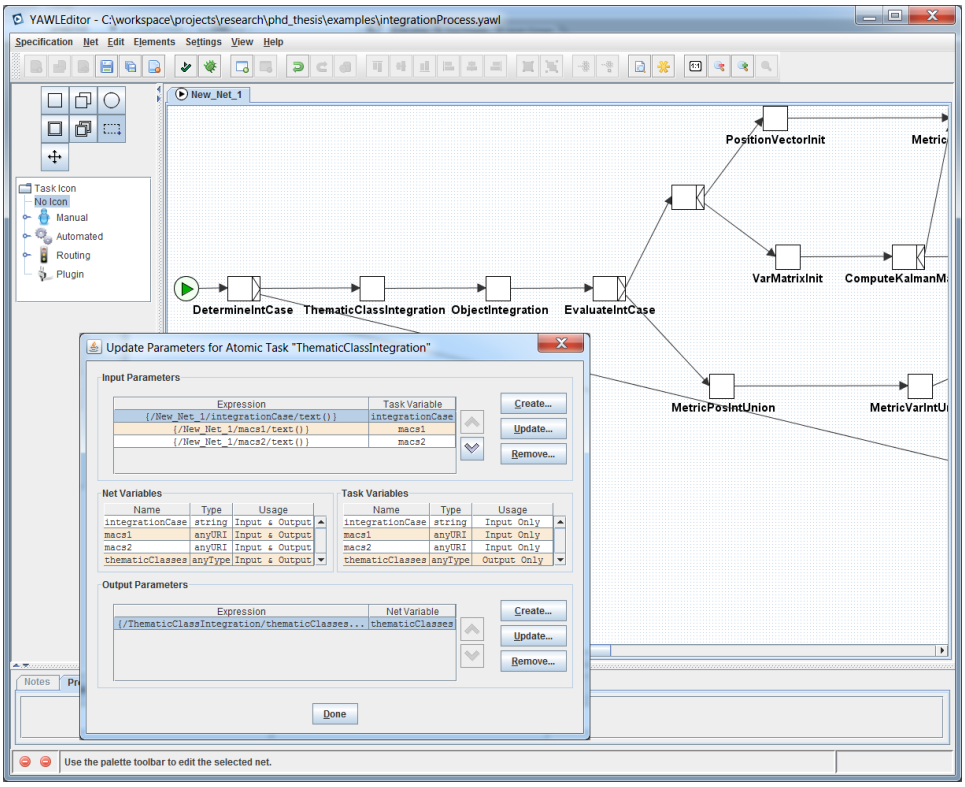


Fig. 5.7. Example of task definitions using the YAWL Editor.

The YAWL System is composed of a Java rich client application, called YAWL Editor, for designing processes, and an Engine which is a web application implemented in the Java Enterprise Edition platform. Fig. 5.6 summarizes the basic YAWL symbols. In particular:

- An *Xor-Split* denotes an exclusive choice, only one of its outgoing branches will be enabled on the basis of a particular condition on workflow variables.
- An *Xor-Join* waits the completion of only one of its incoming branches to continue.
- An *And-Split* denotes a parallel split, all the outgoing branches will be enabled in parallel.
- An *And-Join* represents a synchronization of all its incoming branches.
- An *Or-Split* denotes a multiple-choice: one or more of its outgoing branches will be enabled on the basis of particular conditions on workflow variables.

- An *Or-Join* represents a synchronization of all its incoming branches that have been previously activated and that can complete.
- A *multiple instance* task is a task for which multiple instances may be generated and executed in parallel.

YAWL provides a good framework for managing human resources, coordinating their activities and monitoring the process evolution. In particular, the YAWL Editor allows one to specify how each task will be offered or allocated to a certain user and how this task will be started by that user. Moreover, we can define what operations can be performed by the user to which a task has been allocated, for instance if he can suspend or delegate the task execution to another user. Finally, some tasks can be marked as automatic, thus they will be executed without allocating them to any user.

Besides the definition of control-flow relations among activities, the data-flow relations are specified by means of shared variables. For each task, the set of internal variables and their mapping with external net variables can be specified. In this way, an enabled task instance can read the value of one or more variables and store at completion the produced values on zero or more, not necessarily distinct, external variables. Fig. 5.7 shows an example of mapping between net and local variables: task `TematicClassIntegration` uses in input the variables $\texttt{macs}_1$, $\texttt{macs}_2$ and `integrationCase`, representing respectively the addresses of the two source MACS databases and the integration case, then it stores the result into the output variable called `thematicClasses`.

Once a process specification has been complete, it can be loaded into the Engine. A centralized web application allows the workflow administrator to manage

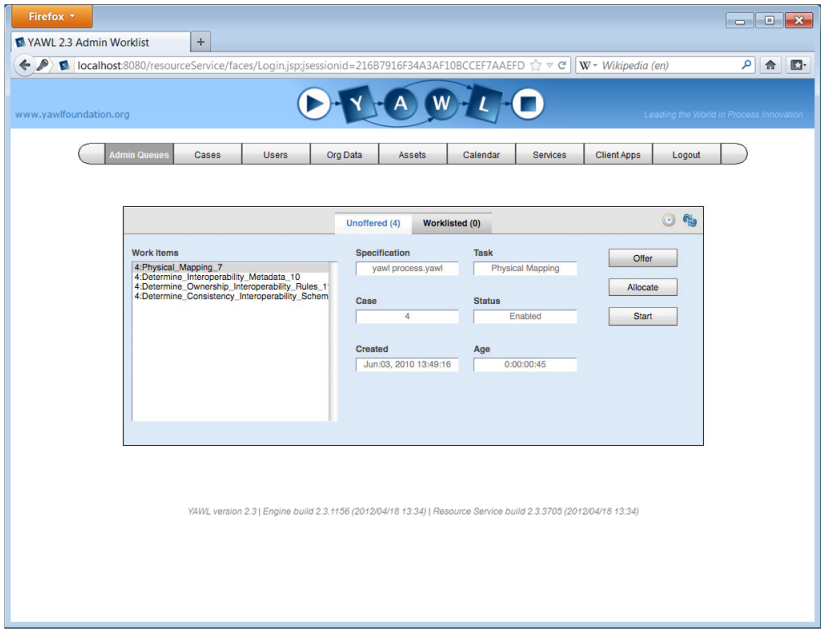


Fig. 5.8. Work list of the workflow administrator.

tasks, assign them to users, monitor their execution, as illustrated in Fig. 5.8. Moreover, he can manage users and roles by defining a hierarchy of roles and the relationships among them, as illustrated in Fig. 5.9, or he can assign particular privileges to a user. Through the same web interface, users can see their work list, containing the task currently allocated or offered to them and the task they are performing, besides to begin or complete a task.

From a specification YAWL system can automatically generate the graphical user interface for the distributed application, that allows the management of the process execution and can be further adapted using XML custom forms.

The strength and weaknesses of YAWL in designing and implementing geo-processes will be discussed in Sec. 6.2.

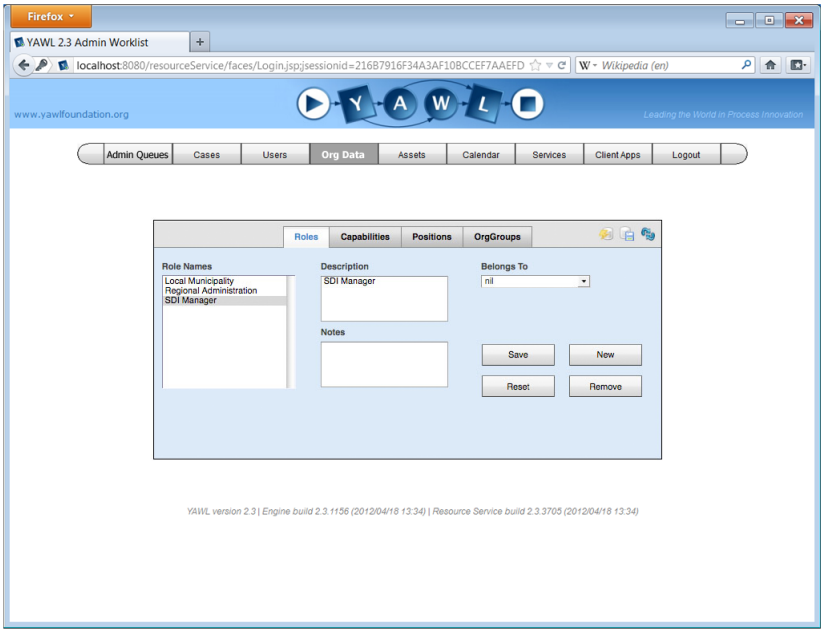


Fig. 5.9. Roles management in YAWL.

Example 5.2. Fig. 5.10 depicts the same process presented in Ex. 5.1, regarding the fulfillment of an order, this time realized in YAWL.

5.5 Data-Flow Oriented Modeling Languages

A Data-Flow Oriented (DFO) modeling language describes a process by explicitly representing data dependencies among its constituent components. A DFO diagram is the graphical representation of a process described in terms of concurrent components that communicate through channels. A channel explicitly captures the notion of data dependency between two components. The state of a process is mainly determined by the contents of its channels.

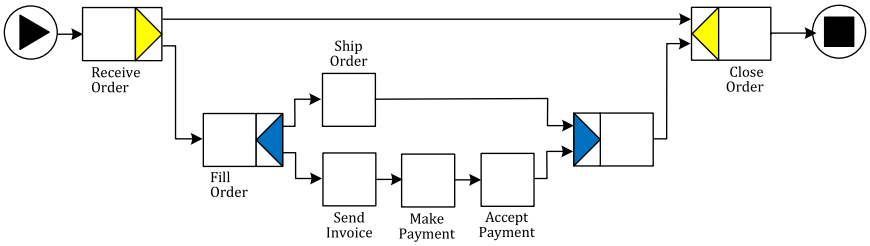


Fig. 5.10. Example of YAWL process.

A channel is a unidirectional queue of ideally unbounded capacity that connects an output port of a component with the input port of another one, or eventually a port of the same component creating a self-loop. Components communicate with each other by receiving and sending messages to their ports. A receive from an input port removes the first message in the incoming channel queue or waits if the queue is empty. A send to an output port places a new message in the outgoing channel without blocking the execution.

If two components `A` and `B` are connected through a channel, it means that a data dependency exists between them, namely that `B` *may* need some data produced by `A` to continue its execution. More specifically, the presence of a message in each input port is mandatory for atomic components that implement functions. Conversely, a composite component can start its execution as soon as it receives the *necessary* input, eventually stopping itself if additional inputs are necessary but are not currently available; similarly, data can be produced on its external output ports as soon as they are available, and not only at completion. This characteristic makes DFO languages more modular than CFO ones, because they allow one to isolate any subset of a process components into a sub-process, while with CFO languages some behavioral-equivalent decompositions are not possible [31].

Moreover, explicit data relations can be used for transparently exploiting parallelism without requiring explicit synchronization, since components free from data dependencies can run in parallel.

5.5.1 The Kepler System

Kepler [81] is an open-source scientific WfMS developed by the members of the Science Environment for Ecological Knowledge (SEEK) project and the Scientific Data Management (SDM) project. It extends Ptolemy II [21], a software system for modeling, simulating, and designing concurrent, real-time systems, developed at UC Berkeley. This system has been chosen as representative for scientific WfMSs because it is one of the most mature and generic one [88].

Kepler inherits from Ptolemy II the support for multiple heterogeneous models of computations (MoCs), captured by the notion of *directors*, that allow the representation of different kinds of systems. The distinctive characteristic of Kepler is the separation between the adopted MoC from the structure of the workflow, which is built combining a set of polymorphic components, called *actors*. This separation is also known as the *actor-oriented modeling paradigm*. An actor implements a functionality of interest for a particular domain, and its behavior can change

on the basis of the adopted MoC. Given the same actors configuration, different execution semantics can be specified through the choice of a particular director (*behavioral polymorphism*). A director defines how actors are executed and how they communicate with each other. The main idea is that a complex model can be built hierarchically by combining different heterogenous models with different MoCs. Five main directors are defined:

- *Synchronous Data-Flow* (SDF) *Director* is designed for representing simple and sequential workflows in which the actors invocation order can be statically determined from the model before its execution. More specifically, before starting a workflow execution, the SDF director pre-computes the order in which actors execute and how many times each actor needs to be fired to complete a single workflow iteration. It requires that the data consumption and production rate of each actor have to be constant and declared.
- *Dynamic Data-Flow* (DDF) *Director*, as the SDF one, executes a workflow in a single execution thread, meaning that tasks cannot be performed in parallel. However, unlike the SDF director, the DDF director does not pre-schedule workflow execution: it determines how to fire actors at run-time, and data production and consumption rates can change during workflow execution.
- *Process Network* (PN) *Director* implements the Process Networks computational model [74], in order to manage workflows that require parallel processing. In a PN workflow each actor has an independent Java thread and the workflow execution is driven by data availability: actors can execute as soon as the necessary input tokens are available in their input ports. Produced tokens are passed to the connected actors through channels of ideally unbounded capacity. These channels are persistent: they retain the received tokens until the actor is able to consume them. A PN workflow terminates when no other components can execute due to the lack of input data, unlike the previous two directors for which the number of desired workflow iterations has to be specified. If tokens are always generated and available to downstream actors, a workflow may not terminate. This director implements the DFO paradigm described in the previous section and is the only director considered in the following.
- *Continuous Time* (CT) *Director* is designed for modeling workflows that predict how a system evolves over time by approximating a mathematical function. For instance, a CT director can be used to define a system that predict the population growth over time. The system is usually described in terms of start conditions and several equations, which are used to predict the state of the system at some specified time in the future.
- *Discrete Event* (DE) *Director* works with timestamps as the CT does. However, timestamps are not used to approximate functions and schedule executions, but to measure average wait times and occurrence rates. Actors send event tokens, which consist of tokens containing data and a timestamp; the director reads these tokens and places them on a global workflow timeline.

Actors communicate by interfaces called *ports*. Ports can be of input, output or mixed type and they are connected through *channels* directed from the output port of an actor to the input port of another one. Each channel can transport a single stream of data. In addition to ports, actors can have a set of *parameters*, which

configure and customize their behavior. Parameters can be statically specified during the workflow design, or dynamically determined through parameter ports.

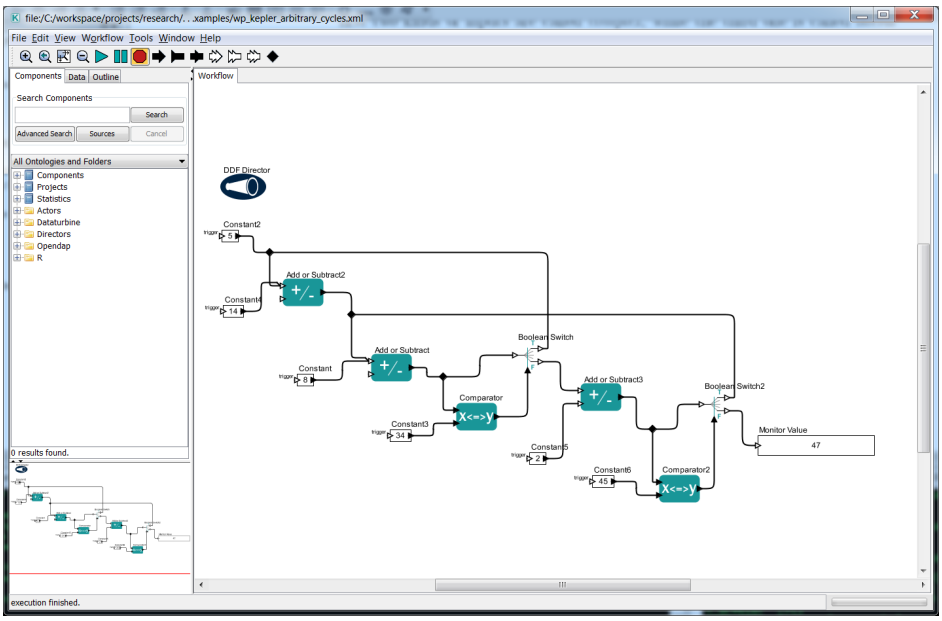


Fig. 5.11. Kepler workflow editor.

Fig. 5.11 shows the Kepler interface: the left panel allows user to search the desired actors to put inside the workflow, the right panel allows user to assemble the workflow and visualize the result of its execution. Workflow execution is managed by the buttons on the top menu bar.

For the purposes of this thesis, in the following the use of the PN director is always assumed. Implicit parallelism and modularity makes DFO languages good candidate for designing distributed and concurrent systems. However, CFO approaches are often preferred to DFO ones. The reasons that usually preclude the adoption of a DFO language for process design can be summarized as follows. Firstly, DFO languages are apparently more complex to use than CFO ones: it is recognized that some broadly used constructions are not easy to define [20, 68]. Moreover, in a DFO process model data dependencies shall be explicitly stated and this activity requires much more efforts than left them unspecified: at some point data dependencies have to be declared in CFO models, but their declaration can be postponed at later design stages or classified as an implementation detail. This approach does not seem a good practice, especially when many concurrent entities are involved, but this simplification initially payoff [68]. Finally, in some cases there is the need to specify control-flow relations in DFO process models for imposing a particular execution order compatible with the existing data dependencies [18]. This can occur when additional information about the system is available, for instance the presence of side effects, or for avoiding the overhead of an undesired parallel execution. Furthermore, when data dependencies impose a

sequential execution, a single control-flow relation can be used to subsume a large set of data-flow dependencies [68, 74].

The strengths and weaknesses of Kepler in implementing geo-processes will be discussed in Sec. 6.2.

5.6 Geo-Processing Workflow Systems

The first attempt to use workflow technologies for geo-processing is due to Weske et al. in [85, 142]. The authors presented a system called *GEO-WASA* that allows the integration of data sources and tools for data analysis, querying and browsing through the use of a WfMS. GEO-WASA is a specialization of the WASA environment for geo-processing. From a very abstract point of view, it can be described as the result of the integration of the process control capabilities of a WfMS and the spatial data handling and visualisation functionalities of an open GIS system. The authors highlighted the benefit of using a WfMS compared with a monolithic GIS system. First of all, GISs are passive systems, in the sense that they respond to users requests, but do not help to coordinate user activities. Second, users need not only more tools, but also facilities to integrate them into a consistent environment. Finally, many applications are highly dependent on human experts: a geo-processing activity is a collaborative and interactive activity.

Similarly, in [9] Alonso and Hagen presented a tool called *Geo-Opera* for supporting the development, execution and management of complex geo-processes. Geo-Opera does not address requirements of spatial data handling such as indexing, storage, or efficient retrieval, for which it relies on existing systems. Its major contribution is bringing together under a single system functionalities that were previously available only as part of isolated tasks.

In recent years, Foester at al. in [43] propose a flexible client application based on OGC standards for processing distributed spatial data over the web. The system has been developed as a plug-in of the User-friendly Desktop Internet GIS (uDig) that supports the connection with distributed data services such as Web Map Service and Web Feature Service. In [113] the authors evolve their client application by considering the possibility of integrating standardised OGC Web Services using WS-BPEL. The aim of their work was to develop an OGC-specific client which is able to: (1) model geo-processing workflows based on OGC web-services using WS-BPEL, (2) execute and visualize single processes but also geo-processing workflows and (3) display the results.

The use of WS-BPEL for chaining OGC web services is widely considered in literature, besides the work presented above it has been adopted also in [123] and [140] for building spatial-related applications by chaining existing services.

A next step towards the adoption of workflow technologies in the geospatial domain is the use of scientific WfMSs, such as Kepler, for chaining geographical web services, as done in [16] and [150]. However, in both these works the aim is to obtain an automatic process by combining existing tools in an efficient way, while no attention is given to collaborative aspects. One of the advantages in using scientific WfMSs is the transparent support of Grid technologies. Several recent studies consider the use of Grid technologies for managing geographic applications

in a distributed environment, such as [35, 61, 62, 79, 86, 103, 119]. The definition of Geospatial Grid is given in [7] where the authors analyze the unique characteristics of spatial data and define the necessary extensions needed for applying the traditional Grid technology in the geospatial domain.

In [23] the authors propose a WfMS that supports the MapReduce approach for geospatial applications. The system is called MRGIS, which stand for MapReduce-enabled GIS and its scheduler exploits two kinds of parallelism. The first one is on *task level* and relies on the fact that many operations can be executed in parallel if they are independent. The other kind of parallelism is on the data level, i.e. by data partitioning. Most GIS operations work block by block, hence a large dataset can be split into multiple independent partitions on which the same task can be executed in parallel. Relatively to this aspect, the difficulty is to estimate how many chunks to split, and how the partial results can be put back together. The authors proposed a simulation based approach for determining the optimal amount of chunks.

The following sections illustrates in more details three main recent contributions about the development of geo-processing WfMSs.

5.6.1 OGC Standards

The *Open Geospatial Consortium* (OGC) is an international voluntary consensus standards organization, originated in 1994 with the aim to define a series of standards for promoting the interoperability in the geographical field. For the purposes of this thesis two standards are of major interest: the web feature service and the web processing service.

The OGC *Web Feature Service* (WFS) *Interface Standard* [100] defines an interface for describing data manipulation operations on geographical features in a standard way. In particular, such data manipulation operations include the ability to (1) retrieve or query features based on spatial and non-spatial constraints, (2) create a new feature instance, (3) delete or update an existing feature instance. The client generates one of this request and posts it to a WFS using HTTP, the WFS executes the request and uses HTTP for distributing the result.

WFS supports two communication models: stateless request-reply and pub-sub. The last one is a messaging system in which clients address messages to a specific node in a content hierarchy, called a topic. The system takes care of distributing the messages arriving from a node publisher to its subscribers.

As regards to the used data format, the Geographical Markup Language (GML) [6] is the default encoding for transporting geographical features, but other formats like shapefiles can also be used. The interface provided by each WFS contains three main operations:

- `GetCapabilities` which returns some metadata about the offered service.
- `DescribeFeatureType` which returns the XML schema of the provided data in order to allow the WFS client to parse the result.
- `GetFeature` which performs the actual query and returns a GML result set containing the full geometry and the feature attributes.

The OGC *Web Processing Service* (WPS) *Interface Standard* [99] defines how to specify the inputs and outputs of a geospatial service in order to invoke it in

a standard way. In particular, it defines how a client can request the execution of a particular process, and how the output of that process is handled. The aim of this standard is to facilitate the publishing, discovery and binding of geospatial processes.

The data required by the WPS can be delivered across a network, embedded in the request, or they can be accessed by URIs. Similarly, the outputs can be embedded in the response, or shared with a URI.

WPS operations are invoked by submitting XML or URI-encoded requests to an online server. WPS can trigger its execution as a web-service, it supports the simultaneous exposure of processes via HTTP GET, HTTP POST, or the Simple Object Access Protocol (SOAP). The interface provided by a WPS is composed of three main operations:

- `GetCapabilities` which returns some metadata about the service.
- `DescribeProcess` which returns a description of the process including its inputs and outputs.
- `Execute` which returns the output(s) of the process execution.

5.6.2 Humboldt Project

The workflow editor developed in the Humboldt project [63] is a lightweight graphical user interface for geo-processing workflow composition. It allows user to graphically compose OGC-conformant web feature services (WFS) and web processing services (WPS) into workflow graphs, which can be directly executed in the editor.

Similarly to Kepler, and in general to all scientific WfMSs, the Humboldt Workflow Editor applies a data-flow approach to workflow composition. Each node inside a workflow can be a computational component (*processing node*) or a data providing component (*data-providing node*). Each edge connects a node associated to a data providing or computational component to another computation component. In case a computational component has multiple inputs, it can have multiple incoming edges, and conversely if a computational component can have multiple outputs, it has multiple outgoing edges. Each edge represents a data-flow dependency: a node can execute only when all input nodes provide a piece of data to the computational node.

In contrast with Kepler, and in general the Process Network model, but in analogy to YAWL, in Humboldt each computational component is required to be function-like, namely it cannot run before all necessary inputs are available and it can produce its output(s) only at completion. Moreover, each processing node is stateless, it has to produce the same output given the same input: the computation cannot depend on some state changes.

In the current implementation, the set of data-providing nodes are OGC WFSs, while the set of processing node are OGC WPSs. Each WFS node does not require any input, the workflow execution engine automatically builds the *invoke-me* messate (`GetFeature`) based on the specified WFS URL and the name of the feature type to be retrieved. The authors do not consider any WFS query and simply assumes that all instances of a particular feature type can be retrieved. The input required by each WPS can be the output produced by a previous WPS node, those offered by a WFS, or those directly defined by the user, for instance a buffer

distance. The first two kinds of inputs are called *complex*, while the third one is called *literal input* and refers to simple datatypes, such as string, integer, double, and so on.

As previously explained, since each Humboldt graph is a data-flow graph, the execution order only depends upon the defined data dependencies. Therefore, if no dependencies are defined between two processing nodes, they can be executed in any order, potentially in parallel.

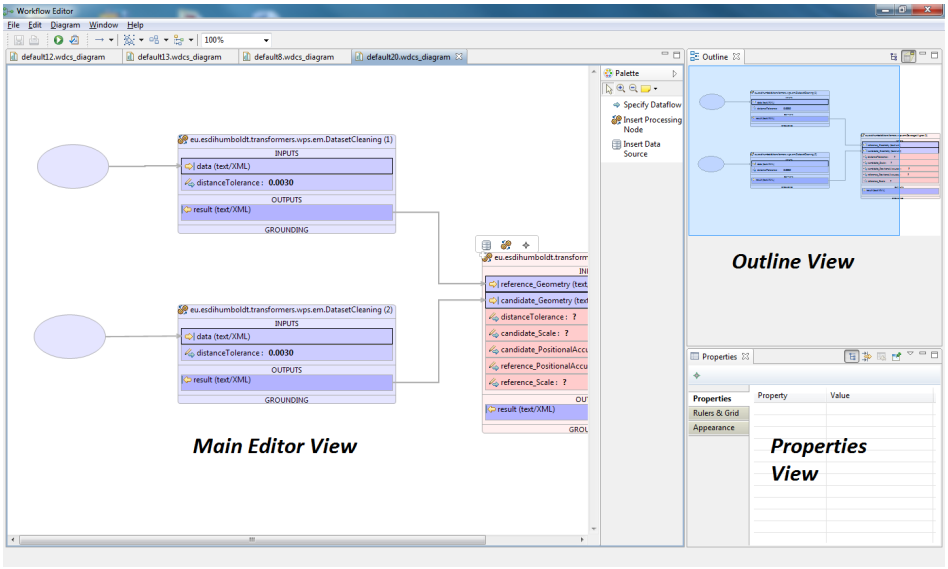


Fig. 5.12. Graphical Interface of the Humboldt Workflow Editor

Fig. 5.12 illustrates the interface of the Workflow Editor: it consists of three views: the main editor view, the outline view and the properties view. The *main editor view* contains the graphical representation of the workflow, in which oval nodes are the data-providing nodes, while the rectangular ones are the processing nodes. In particular, the representation of each processing node contains the specification of input and output parameters, that have to be connected with other processing nodes or data-providing nodes, and of the grounding, namely the information on where the processing node is actually accessible (i.e. for WPS it is the URI and the `ProcessIdentifier` of the process). The *properties view* contains some additional information about the workflow elements that are not included in the main editor view. Finally, the *outline view* contains an overview of the whole workflow diagram in form of tree.

Any time two nodes in the workflow are connected, the system performs a type matching based on the two nodes metadata (i.e. OGC WFS or OGC WPS metadata). However, since such metadata can contain errors, the outcome of the type checking is provided only as a warning for the user and does not prevent

the definition of the data-flow connection. Moreover, the Workflow Editor allows a user to specify:

- a timeout for a single processing node or for the entire workflow,
- whenever the processing nodes have to be executed in parallel when no data-flow dependencies have been specified between them,
- whenever the execution engine will try to execute each processing node in the workflow, even if some nodes have already failed,
- the value passing strategy and the storage location for the computed results; in particular, the value passing strategy can be: pass-by-reference only, pass-by-value only, or try-both.

Before executing a workflow, it is subject to a twofold validation: first, the system checks if all inputs have been specified; secondly, it performs a check on the overall structure of the workflow (e.g. it checks the presence of a tree-like structure, unconnected nodes, etc.). An input is unspecified if it is unclear where its value will come from when the workflow will be executed. A literal input is unspecified, if the designer does not provide any input value, while a complex input is unspecified, if there is no data-flow edge pointing to the corresponding node.

The major limits with respect to existing WfMSs, in particular as regards to the scientific ones, are:

- Inability to insert a whole workflow as node into another workflow, allowing the reuse of workflows and reducing the size and complexity of the workflow graph. This inability dramatically reduces the scalability of the system and the chance to use it for modeling large processes, that usually involve several tasks, task previously defined, compensation activities, and so on. Moreover, even if such possibility will be implemented in the future, the adopted function-like style will reduce the langauge modularity, with respect to Kepler and WS-BPEL which use stateful processes.
- Impossibility to define conditions on the data produced by WFS.
- Only acyclic graph are allowed.
- As for Kepler, no support for resource management is provided, namely the system has been designed main for automating batch processes.

5.6.3 ArcGIS Workflow Manager

The ArcGIS Workflow Manager [42] (previously known as ESRI Job Tracking) is a proprietary WfMS for modeling and executing geographical processes on top of ArcGIS. In particular, it allows one to develop and enforce repeatable GIS workflows, ensuring that the right work is completed correctly, by the right person or team. In particular, it can automate common activities, track the status and progress of jobs, integrate GIS and other business applications, assign activities by geometry, allocate resources and verify who is working on what, manage data coming from different ArcSDE databases, and associate geographical areas to jobs.

Fig. 5.13 shows an example of the system main window: the table on the upper-right illustrates the list of jobs composing a workflow. Each job is characterized by an integer identifier, a name, a type, the name of the user to which it has been

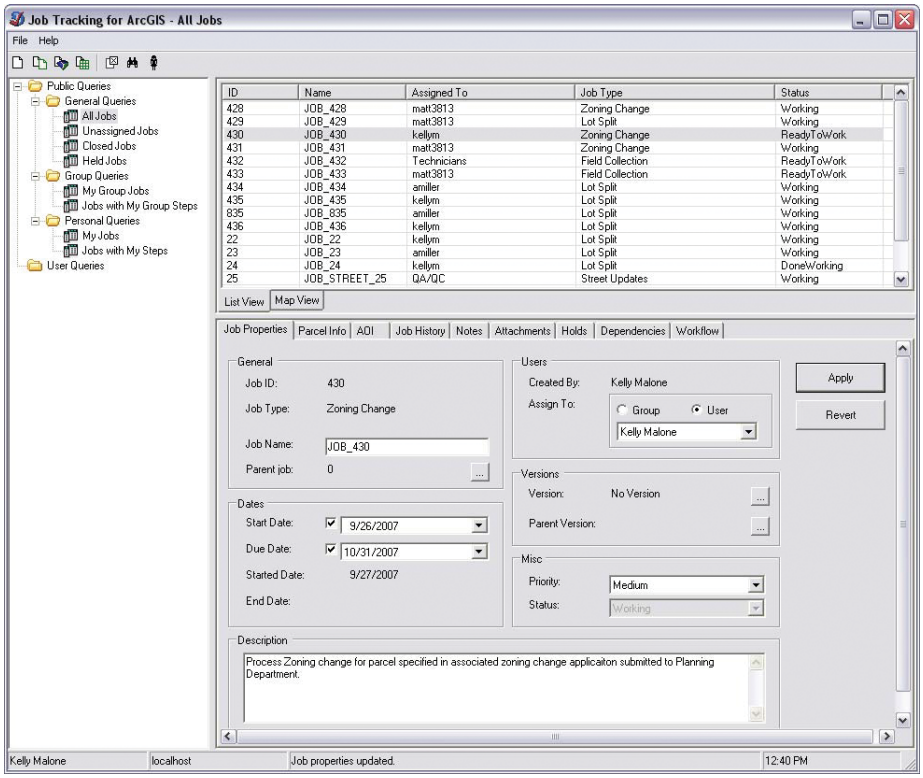


Fig. 5.13. Main window of Arcgis Workflow Manager

assigned, and its status (e.g. done, ready, working, and so on). The bottom part of the window illustrates the details of a job, for instance it allows one to assign the job to a particular user, or set a priority. The job list can be visualized also using an alternative view, called *map view*, which displays the spatial distribution of the jobs, as depicted in Fig. 5.14.

In this system, workflows are composed of steps and paths, Fig. 5.15 illustrates and example of workflow specified in ArcGIS. A *step* is an individual task to be completed by a user. A *path* is a conditional route linking steps. In other words, paths define what steps will be done, and steps define the activity to be done. A complete workflow is a network of steps joined by paths. By default, a step consists of a description of some task that needs to be performed by the user at that particular point in the overall process. It serves as an indicator to the user of what needs to be done. The user performs the task, then indicates through the system interface that the task has been completed, and the workflow engine can advance to the next step. Additionally, steps can be configured with behavior, namely the workflow can be set up to automate certain activities for the user.

Any existing ESRI or non-ESRI executable, application or program can be plugged into the workflow. In particular, developers can use the JTX (Job Tracking for ArcGIS) application programming interface and web-services to build ap-

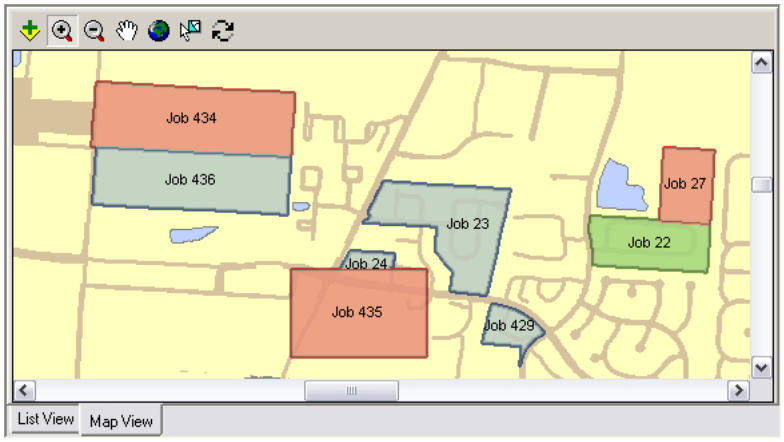


Fig. 5.14. The job list in the main window can be also visualized as a map: each job is related to a particular are, whose color defines the job status.

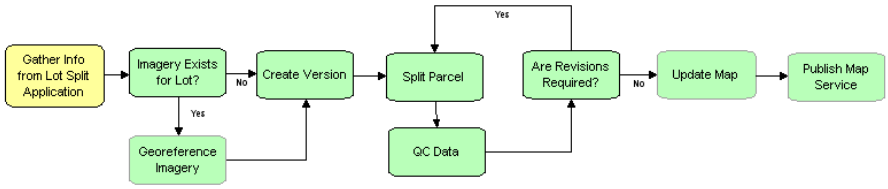


Fig. 5.15. Example of workflow specified in ArcGIS Workflow Manager.

plications that can be plugged into a workflow. On the contrary, data can be only provided using ESRI geodatabase technology.

A very important aspect of any business process is the human resources who will play a role in the initiation, execution, and management of work being performed. The user management functionality offered by ArcGIS Workflow Manager allows one to define how steps are allocated and distributed among the different users. In particular, a job can be initiated by a single user and carried-out entirely by that person, or a job could involve many players who need to interact with the job at various stages in its life-cycle. Moreover, workflows can be configured to automatically assign the job to a user when a specific step is reached; or jobs can be initially assigned to a group of users, placing the job into a virtual queue from which each user of the group can pull a task when he is ready to begin a new job. Finally, a powerful aspect of job management is the use of geographic area to delineate the working extent of a task. Such area is called area of interest (AOI) and tracks where a particular work has to be performed. Fig. 5.16 illustrates an example of an area of interest assignment for the job 440.

Despite ArcGIS Workflow Manager exposes a lot of interesting features, it remains an add-on of the ArcGIS Suite. This thesis focuses on open solutions that can be used to build distributed software systems compliant with the OGC standards.

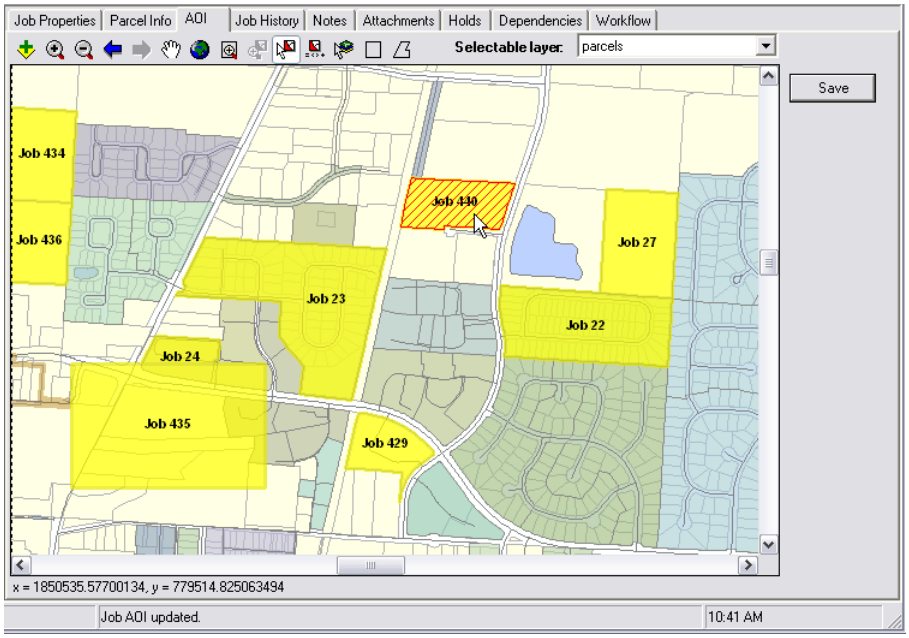


Fig. 5.16. Example of assignment an area of interest for the job 440.

5.7 Summary and Concluding Remarks

Workflow technologies are increasingly applied for building new information systems, especially when these systems cross the boundary of a single organization. The development of a SDI allows one to create a virtual organization that spawns multiple existing spatial agencies. Therefore, we evaluates the applicability of existing WfMSs for supporting geographical processes with the characteristics exposed in the introduction. Therefore, we start in this chapter with a brief overview of the methods and the technological infrastructure underpinning the workflow concept, and then we introduce some representative WfMSs.

In particular, we concentrate on the so-called business and scientific WfMSs, taking BPMN [97], YAWL [127] and Kepler [81] as representatives, and we also take a look at some emerging geo-processing solutions, considering in particular the Humboldt project [63] and the ArcGIS Workflow Manager [42].

These systems can be classified we respect to several criteria, that have also preclude a further analysis of some of them. In particular, Fig. 5.17 shows a classification with respect to the kind of processes and the purposes for which the considered systems have been developed. Some characteristics of geographical processes make them sometimes similar to and sometimes different from both traditional business processes and scientific experiments; hence, we have isolated them into a separate column. As regards to the main purpose of these systems, we can identify two kind of systems: those oriented to the process automation and those oriented to agents coordination. The geographical processes we are interested to are those involving several different human agents that interact together for achieving a predefined goal, as highlighted in Chap. 1. Therefore, in the following we will consider

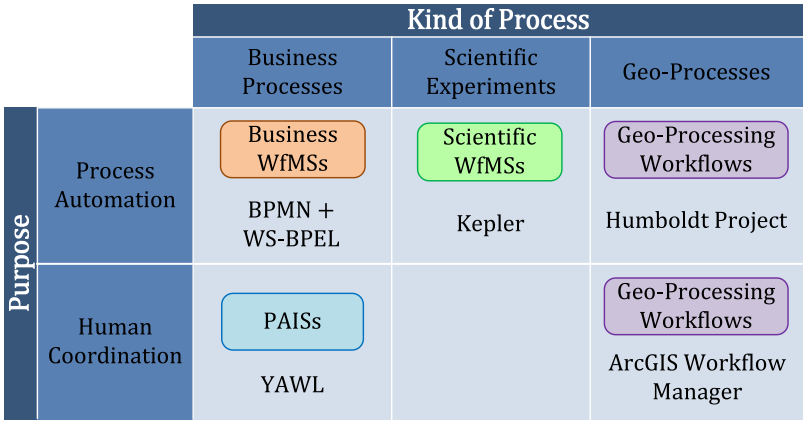


Fig. 5.17. Classification of the considered WfMSs with respect to the kind or process and the purpose for which they have been developed.

as representative for business WfMSs, the YAWL System rather than the BPMN language with a WS-BPEL execution semantics.

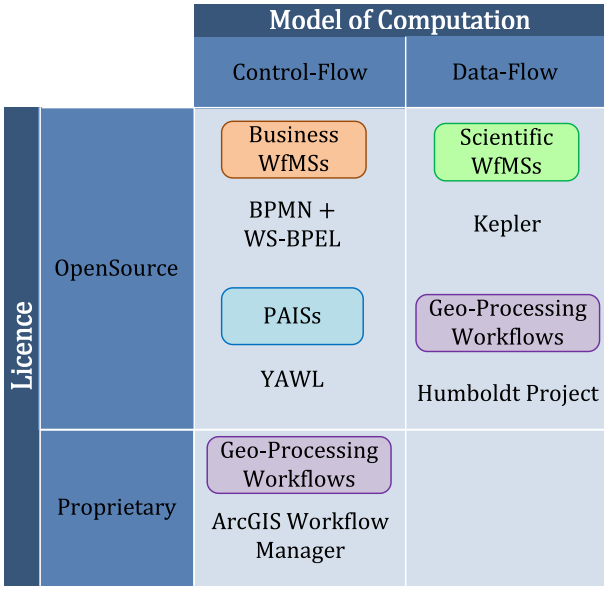


Fig. 5.18. Classification of the considered WfMSs with respect to the implemented model of computation and the distribution licence.

Another classification of the considered system is presented in Fig. 5.18 which compares them with respect to the implemented model of computation and the kind of licence. Relatively to the model of computation, we can observe that the Humboldt Project implements a data-flow model similar to the Kepler one, but the

former one is more prototypical than the second one. For instance, Humboldt does not allow the presence of cycles and tasks can implement only stateless functions. Therefore, any Humboldt workflow can be realized as a Kepler workflow, in which data sources are implemented as WFSs and tasks are implemented as WPSs; hence, in the following we can safely ignore this system.

Finally, as regards to the distribution licence, ArcGIS Workflow Manager is the only considered system with a proprietary licence. Moreover, the system is strictly related to the ArcGIS technology, for instance as regards to the data source implementation, and it cannot be easily integrated with the OGC standards. For these reasons, the system will not be considered in the following, even if it provides several interesting functionalities and is one of the most mature workflow system for geo-processing.

In order to evaluate the applicability of the chosen WfMSs, the following chapter compares them with respect to two criteria: first from a more general point of view, using the well-known framework of workflow patterns, then from a geographical point of view, taking the proposed integration process as use case.

6

A Comparison of the Existing WfMSs

This chapter provides a comparison among the existing WfMSs introduced in the previous chapter. In particular, the comparison is twofold: first the systems are analyzed using the established workflow patterns methodology, then they are evaluated from a geo-information point of view, trying to implement some aspects of the integration process presented in the first part of the thesis.

In the BPM community, the workflow patterns initiative [5] provides a framework to evaluate the suitability of WfMSs for business process design based on a set of recurring constructions (the so-called *patterns*). In literature many offerings have been evaluated against workflow patterns, providing a good starting point for a system comparison. Inspired by these contributions, Sec. 6.1 presents a pattern-based evaluation of Kepler [81], and compares the obtained results with the evaluation of YAWL, BPMN and WS-BPEL contained in [110], [146] and [145], respectively. The section also discusses if some patterns are not directly supported by Kepler because deemed not to be relevant for that application domain, or due to the relatively immaturity of the system.

In Sec. 6.2 these systems are also compared from a more specific geographical point of view, by considering the implementation of the integration process presented in Chap. 4. This process is a good representative of geographical processes, because it has many characteristics highlighted in the introduction. The proposed comparison allows one to identify the strengths and limits of the considered WfMSs in the geographical domain and leads in Sec. 6.3 to the definition of an ideal solution.

6.1 Workflow Pattern Evaluation of the Selected WfMSs

This section presents the pattern-based evaluation of Kepler and compares such system with the existing evaluation of YAWL, BPMN and WS-BPEL performed in [110, 145, 146], respectively. For this evaluation we consider only the standard constructs provided by the default distribution of Kepler and we do not refer to any ad-hoc extension or third-party additional libraries of functionalities. In the following the term *task* is used as a synonymous of Kepler actor, since it

is essentially a workflow component implementing a particular functionality of interest. More details about the various patterns definition can be found in App. B.

6.1.1 Workflow Control-Flow Patterns Evaluation

Workflow Control-Flow Patterns (WCPs) [109] describe a set of recurring constructions that are commonly offered by WfMSs for defining the flow of control among various tasks. In Kepler dependencies among tasks are data dependencies: a relation $A \dashrightarrow B$ (dashed arrow) between A and B means that B may need some data produced by A to complete its execution. A control-flow dependency on the other hand is concerned with the termination of tasks: a relation $A \rightarrow B$ (solid arrow) between A and B means that B can start only when A terminates its execution. A control-flow dependency can always be represented in terms of a data dependency by assuming that A produces an output only just before completing and that output is required to start the following task. Conversely, the opposite assertion does not hold. In order to evaluate the WCPs support of the three systems, we always assume that each task needs all the inputs before its execution and produces its output only at completion.

The following sections analyze how the various WCPs can be realized in Kepler, while the final results and their comparison with the support provided by the three representative business WfMSs will be summarized in Tab. 6.1.

(G1) Basic Control-Flow Patterns

Basic control-flow patterns capture elementary aspects of process control. A Sequence (Wcp-01) relation between two tasks A and B can be obtained by defining a data-flow dependency between A and B.

A Parallel Split (Wcp-02) determines the divergence of an execution branch into two or more branches that execute in parallel. It can be obtained by redirecting the output of a task to different channels, in this manner all subsequent tasks whose input ports are connected to these channel will be activated at the same time. In particular, the Kepler `Relation` operator can be used to replicate the data it receives into different channels.

Conversely, the Synchronization pattern (Wcp-03) represents the synchronization of two or more parallel branches into a single subsequent branch that is enabled only when all the previous branches have completed. In the three considered systems each task can have one or more input ports and it can execute only when a data token is available in each channel connected to those ports. Therefore, any task having more than one input port can be used for synchronizing the execution of the previous tasks.

An Exclusive Choice (Wcp-04) is the divergence of a branch into two or more branches such that only one of them is executed, based on a particular condition. It can be obtained in Kepler through the `Switch` actor which routes the data received on its data input port to only one of the connected output channels, on the basis of a control value received in the control input port. The only drawback of this implementation is the possibility to lose a data token when the control value is out of range, but we can assume that the conditions used to generate the control

values are properly implemented to produce a valid output for the selection. If the exclusive choice regards only two alternatives, you can also model this pattern through a `Boolean Switch` actor.

The Simple Merge pattern (WCP-05) represents the convergence of two or more incoming branches into a single subsequent branch that is enabled as soon as one of the incoming branches completes. As regards to this pattern the Kepler `Relation` operator can be used with the sequential directors (SDF and DDF) to combine the output produced by several previous actors into a unique channel. Similarly, the `Nondeterministic Merge` actor can be used for the same purpose with the process network (PN) director.

(G2) Advanced Branching and Synchronization Patterns

Advanced branching and synchronization patterns characterize more complex branching and merging concepts which arise in business process modeling.

A Multi Choice (WCP-06) extends the Exclusive Choice pattern (WCP-04) by allowing one or more outgoing branches to be enabled, based on the evaluation of a given condition. It can be realized in Kepler by combining a Parallel Split (WCP-02) with several Exclusive Choice constructs (WCP-04), one for each branch of the Parallel Split. In particular, a relation operator can be used to broadcast a value to different outgoing channels, and in each channel a `Boolean Switch` actor is used to decide if the corresponding branch is enable or not, as illustrated in Fig. 6.1. In

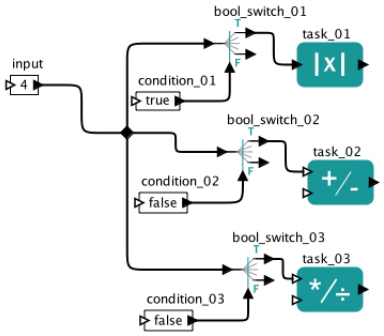


Fig. 6.1. Example of WCP-06 Multi Choice implemented in Kepler

order to ensure the presence of a default branch, the conditions associated with the various exclusive choices have to be defined so that at least one of them evaluates to true. This implementation follows one of the work-around solutions discussed in [109] which is not considered to constitute a native support for the pattern, since the decision process associated with the evaluation of the Multi-Choice is divided across multiple split constructs.

A Structural Synchronizing Merge (WCP-07) determines the convergence of two or more branches, which diverged at a unique identifiable point in the model (represented by a single Multi-Choice construct), into a single subsequent branch

that is enabled when each *active* branch has completed. In other words, this construct does only wait for those incoming branches that will eventually complete.

Similar to the construction presented above for the Multi Choice, several Merge operators (WCP-05) followed by a Synchronization construct (WCP-03) can be used to implement the Structured Synchronizing Merge (WCP-07). In particular, each pair of branches outgoing the same Exclusive Choice will be merged by a `Relation` operator or a `Nondeterministic merge` actor, depending on the chosen director, while the merged outputs are synchronized using a task with one input port for each connected channel. Notice that the various `Relation` operators (or the `Nondeterministic merge` actors) can also be realized with a `DDF Boolean Select` actor with the same enablement condition of the corresponding `Boolean Switch`, as exemplified in Fig. 6.2. These implementations of WCP-07 are also considered a work-around and the pattern cannot be considered directly supported due to the absence of a Multi Choice construct.

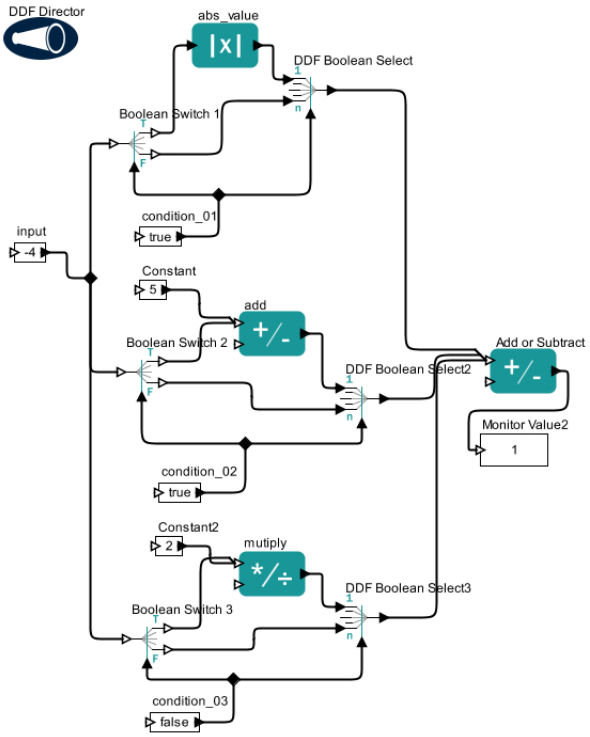


Fig. 6.2. Example of WCP-07 Structured Synchronizing Merge implemented in Kepler.

The unstructured forms of this pattern, Local Synchronizing Merge (WCP-37) and General Synchronizing Merge (WCP-38), require the synchronization of two or more branches, which diverged earlier in the process, into a single subsequent branch using local, respectively, global information about their current and future activations. These patterns cannot be realized in Kepler, since there is no way to

determine the number of active branches nor if a branch that has not yet been enabled will be enabled at any future time.

As regards to Multi Merge (WCP-08), we observe that the difference between it and its safe version Simple Merge (WCP-05) concerns the context in which they are used. In the former more than one incoming branch can be active simultaneously, while in the latter only one incoming branch can be active at a time. For all the three systems, the implementations given for WCP-05 can also be used in an unsafe context where more than one incoming branch is simultaneously active, producing the required behavior. Therefore, WCP-08 can be considered supported.

The discriminator and partial join patterns represent the convergence of two or more incoming branches into a single subsequent branch that is enabled when exactly k of the n incoming branches have completed. For the discriminator the value k is one, while for the partial join it is a number between two and n. If the incoming branches have diverged from a single identifiable point in the model, we are in the presence of the Structured Discriminator (WCP-09) or the Structured Partial Join (WCP-30); otherwise, we refer to Blocking Discriminator (WCP-28) or Blocking Partial Join (WCP-31). Finally, if the execution of the remaining branches is canceled instead of blocked at completion, we have the Canceling Discriminator (WCP-29) or the Canceling Partial Join (WCP-32). In relation to the Structured Discriminator (WCP-09), the Blocking Discriminator (WCP-28), the Structured Partial Join (WCP-30), and the Blocking Partial Join (WCP-31) patterns, we can observe that Kepler lacks any support for the partial synchronization of different tasks. Instances of different tasks that execute in parallel can run independently from each other until completion or can all be synchronized by connecting their output ports to the input ports of the same subsequent task.

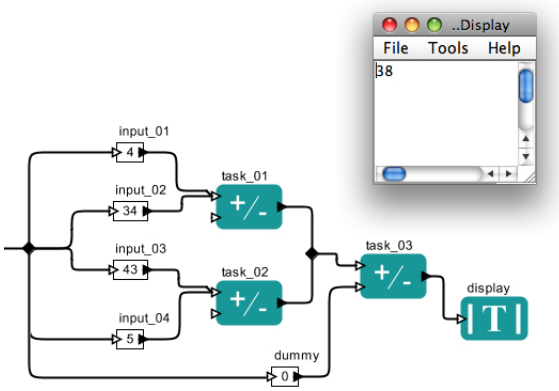


Fig. 6.3. Example of WCP-09 Structured Discriminator implemented in Kepler

Fig. 6.3 shows an attempt to implement WCP-09 where `task_03` represents the discriminator. This actor needs two inputs, the first from a `Relation` operator (or a `Nondeterministic Merge`) that merges the data produced by the branches to be synchronized into a unique channel, and the second from another distinct branch. The second branch (containing the actor `dummy`) produces only one value; therefore,

`task_03` is enabled only one time consuming the first data value produced by one of the branches to be synchronized and blocking all the other data tokens that subsequently arrive. Even if this solution is able to simulate the behavior of a (structured) blocking discriminator in an acyclic context, it is not able to reset the construct when exactly one piece of data is received from each channel connected with the `Relation` operator, removing the unnecessary data from the workflow. Moreover, a dummy constant value or a trigger port has to be used for ensuring a single enablement of the subsequent actor, this input acts as a control value and does not influence the computation. Therefore, the pattern cannot be considered directly supported.

The implementation presented in Fig. 6.3 for the (structured) blocking discriminator cannot be extended for synchronizing an arbitrary number of tasks (WCP-30, WCP-31) by substituting the single dummy value with a list of values of the desired length k, because there is no guarantee that the k synchronized threads come from different branches and in the worst case they can all come from the same branch. For the canceling version of Discriminator (WCP-29) and Partial Join (WCP-32), we can observe that Kepler does not provide a way to withdraw an activity or eliminate data tokens previously produced.

The lack of full support for the partial synchronization of different task instances can be considered a real limitation, because during experiments it may be convenient to start a number of activities in parallel and wait the completion of only one (or a sub-set) of them before proceeding. For instance, to perform a complex operation, such as DNA matching, it is reasonable to apply multiple techniques in parallel: the first available result will be used by the subsequent activities, while the other ones can be discarded when they arrive, or the activities producing them can be canceled altogether. Similarly the absence of mechanisms for canceling a running activity is limiting. The scientific domain is characterized by a high degree of uncertainty: if the result provided by an activity does not satisfy the expectations, the user has to be able to cancel the other running activities and change those that still have to be performed. This limitation probably comes from the relative immaturity of the considered system and in the future they may be expected to integrate several facilities for exception handling and managing compensation activities.

The Generalized And-Join (WCP-33) is an extension of the Synchronization (WCP-03) pattern, where the branches to be synchronized can come from several divergency points. It can be obtained by connecting the output ports of the tasks to be synchronized with the input ports of a synchronizing task t, so that t will wait the completion of all previous tasks before starting, as stated for its structured version (WCP-03). Finally, Thread Merge (WCP-41) and Thread Split (WCP-42) patterns represent the merge, respectively, the generation of several threads of control from a single point in the model. They can be implemented in Kepler using the `SynchOnTerminator`, respectively, the `Repeat` actor.

(G3) Multiple Instance Patterns

Multiple instance patterns describe situations where there are multiple threads of execution related to the same activity. The data-flow paradigm underlying scientific WfMSs provides a natural mechanism for generating multiple instances, but

Kepler is not able to synchronize them at completion: the created instances run independently from each other until the workflow completes. Therefore, only the Multiple Instance without Synchronization pattern (WCP-12) is supported. This choice probably comes from the assumption that several instances of the same experiment can be performed in parallel on different inputs and their executions are mutually independent. When all the experiment instances are terminated, the scientist collects the obtained results and manually derives global outcomes. This also originates from the fact that scientific WfMSs have been developed for automating large-scale experiments, rather than for coordinating the work of a group of agents.

(G4) State-based Patterns

State-based patterns describe situations in which the execution depends on the state of a process instance. The Deferred Choice (WCP-16) pattern allows a choice to be determined by an external input and not by internal data. As such, this pattern requires interaction with the external environment.

In Kepler the `BrowseUI` actor allows one to display a web page containing an HTML form and retrieve the inserted values as XML name/value pairs. This actor allows users to choose the action that the subsequent task will have to perform among different alternatives. However, interactions with the external environment are not limited to user choices: other kinds of interaction can be the arrival of a message or the expiry of a timer. Therefore, the pattern cannot be considered directly supported.

Interleaved Parallel Routing (WCP-17) and Interleaved Routing (WCP-40) are related to the ability of executing each task in a given set at a time until all tasks are executed, where in WCP-17 a partial order can also be defined among the tasks of this set. Therefore, these two patterns can be used to limit the number of running threads to one at a time, and so the number of used resources. In scientific WfMSs available resources are automatically managed by the underlying environment (e.g. a Grid environment) in a transparent way for the user. Therefore, there is no need to provide users with a method to manually control the number of used resources. In Kepler, WCP-17 and WCP-40 can implement by choosing one of the sequential directors.

The Critical section (WCP-39) pattern describes a situation where two or more connected subgraphs in the model are identified as "critical sections", where a critical section is a set of tasks that can only be executed in mutual exclusion with the tasks of other critical sections. This pattern is not supported. Finally, Milestone (WCP-18) represents the situation in which a task is enabled only when the process instance is in a specific state, represented by a specific execution point. This pattern is also not supported by Kepler.

(G5) Cancelation and Force Completion Patterns

Cancelation patterns involve the ability to withdrawn an activity (WCP-19 Cancel Activity), or a set of task instances in the same process instance (WCP-25 Cancel Region) or an entire process instance (WCP-20 Cancel Case). Alternatively, they

can regard the cancelation (WCP-26 Cancel Multiple Instance Activity) or the forced completion (WCP-27 Complete Multiple Instance Activity) of a multiple instance activity. Kepler does not support cancelation patterns: there is no way to cancel or force the completion of an activity that is executing or is scheduled for execution.

(G6) Iteration Patterns

Iteration patterns capture repetitive behavior in a workflow. Arbitrary cycles (WCP-10) are cycles that have more than one entry and/or exit points. They can be implemented in Kepler by using the `Relation` operator or the `Nondeterministic Merge` actor for representing the various entry points, and the `Switch` or `Boolean Switch` actor for representing the various exit points, as exemplified in Fig. 6.4.

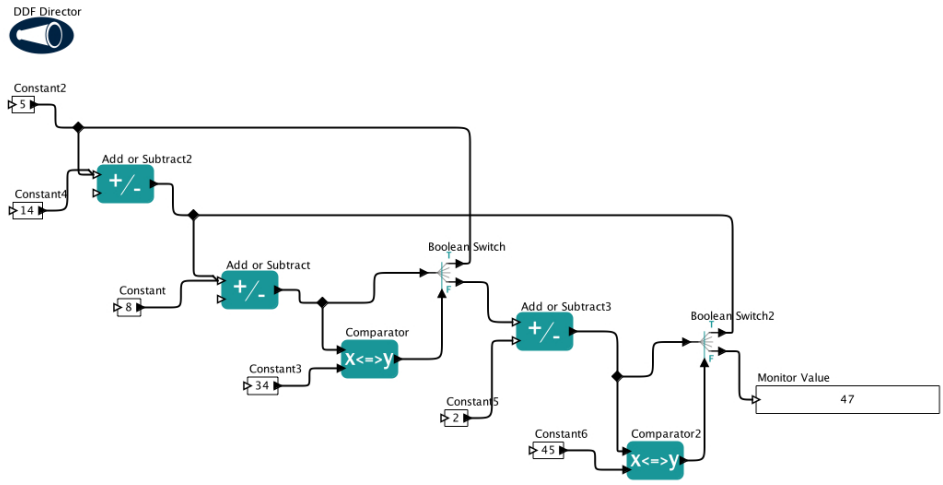


Fig. 6.4. Example of WCP-10 Arbitrary Cycles implemented in Kepler.

The same construction proposed for arbitrary cycles can also be used to implement the Structured Loop (WCP-21), by ensuring the existence of a single entry and a single exit point. However, this pattern is considered to be directly supported only in the presence of a specific loop construct. In Kepler a structured loop can also be obtained using a `Ramp` actor which works like a for-loop allowing the execution of one or more tasks a specified number of times.

Kepler supports the hierarchical decomposition of tasks; however, a composite task cannot contain itself in any of its decompositions, neither directly nor indirectly. Therefore, the recursion pattern (WCP-22) is not supported in terms of workflow; recursive definitions can be obtained only in a programmatic way through the underlying implementation programming language.

(G7) Termination Patterns

Termination Patterns capture various ways in which a workflow can complete. In scientific WfMSs a workflow execution terminates when there are not sufficient data tokens left to execute any task, thus WCP-11 Implicit Termination is the commonly supported termination pattern. Kepler also directly supports Explicit Termination (WCP-43) by providing a specific construct to explicitly terminate a workflow execution that reaches a certain point.

(G8) Trigger Patterns

Trigger Patterns capture external signals that may be required to start certain tasks. A trigger can be obtained by adding an additional input port to a task. The input value received through this port is not used during the computation, but it is useful only for synchronization purposes and not during computation. Kepler natively support this concept by means of trigger ports and trigger messages, respectively. However, channels are persistent: tokens that flow through them are retained until the connected task is able to consume them. Therefore, only Persistent Trigger (WCP-24) is supported.

Final Observations

Table 6.1 summarizes the WCPs support of Kepler and compares it to the other three representative business process modeling languages considered here. Following the evaluation criteria established in [109], a "+" rating (direct support) or a "±" rating (partial support) is assigned when the system provides a construct that completely, respectively, partially satisfies the description of the pattern when used in a case satisfying the context assumptions. Otherwise, a "−" rating (no support) is assigned. In particular, given the difference between the computational models of scientific and business WfMSs, we give a + to a system either if it can realize the control-flow dependency expressed by that pattern via a data-flow dependency between computational tasks and/or routing constructs (e.g. an If, Loop or Merge), under the assumption that a task instance requires its inputs only before starting and produces its outputs only at completion. We did not consider work-arounds such as hard-coding the desired behavior of a pattern in a task as direct support for that pattern. More details on the definition and evaluation criteria of the various WCPs can be found in [88].

The support provided by all the considered systems for the basic control-flow patterns is substantially the same. Conversely, the support of these systems is quite different with respect to the advanced branching and synchronization patterns. In particular, the various Discriminator and Partial join patterns are not supported by Kepler and by WS-BPEL, while BPMN and YAWL provides some support for them. This is because BPMN and YAWL are defined according to a graph-oriented paradigm, whereas WS-BPEL is essentially block-structured, except for the `flow` construct. On the other hand, some other patterns are fully or partially supported by the three business WfMSs, but they are not supported at all by Kepler, as in the cases of the two patterns dealing with the multiple enablement or synchronization

Table 6.1. Workflow Control-Flow Patterns support in the considered WfMSs. The first column reports the patterns group, G1: basic control-flow, G2: advanced branching and synchronization, G3: multiple-instance, G4: state-based, G5: cancelation and force completion, G6: iteration, G7: termination, and G8: trigger.

	Pattern	BPMN	WS-BPEL	YAWL	Kepler
G1	WCP-01 Sequence	+	+	+	+
	WCP-02 Parallel Split	+	+	+	+
	WCP-03 Synchronization	+	+	+	+
	WCP-04 Exclusive Choice	+	+	+	+
	WCP-05 Simple Merge	+	+	+	+
G2	WCP-06 Multi Choice	+	+	+	-
	WCP-07 Structured Synchronizing Merge	+	+	+	-
	WCP-08 Multi Merge	+	-	+	+
	WCP-09 Structured Discriminator	±	-	+	-
	WCP-28 Blocking Discriminator	±	-	+	-
	WCP-29 Canceling Discriminator	+	-	+	-
	WCP-30 Structured Partial Join	±	-	+	-
	WCP-31 Blocking Partial Join	±	-	+	-
	WCP-32 Canceling Partial Join	±	-	+	-
	WCP-33 Generalized And-Join	+	-	+	+
	WCP-37 Local Synchronizing Merge	-	+	+	-
	WCP-38 General Synchronizing Merge	-	-	+	-
	WCP-41 Thread Merge	+	±	+	+
	WCP-42 Thread Split	+	±	+	+
G3	WCP-12 M.I. without Synchronization	+	+	+	+
	WCP-13 M.I. with a priori design-time know.	+	-	+	-
	WCP-14 M.I. with a priori run-time know.	+	-	+	-
	WCP-15 M.I. without a priori run-time know.	-	-	+	-
	WCP-34 Static Partial Join for M.I.	±	-	+	-
	WCP-35 Canceling Partial Join for M.I.	±	-	+	-
	WCP-36 Dynamic Partial Join for M.I.	-	-	+	-
G4	WCP-16 Deferred Choice	+	+	+	-
	WCP-17 Interleaved Parallel Routing	-	±	+	+
	WCP-18 Milestone	-	-	+	-
	WCP-39 Critical Section	-	+	+	-
	WCP-40 Interleaved Routing	-	+	+	+
G5	WCP-19 Cancel Activity	+	+	+	-
	WCP-20 Cancel Case	+	+	+	-
	WCP-25 Cancel Region	±	±	+	-
	WCP-26 Cancel Multiple Instance Activity	+	-	+	-
	WCP-27 Complete Multiple Instance Activity	-	-	+	-
G6	WCP-10 Arbitrary Cycles	+	-	+	+
	WCP-21 Structured Loop	+	+	+	+
	WCP-22 Recursion	-	-	+	-
G7	WCP-11 Implicit Termination	+	+	-	+
	WCP-43 Explicit Termination	+	-	+	+
G8	WCP-23 Transient Trigger	-	-	+	-
	WCP-24 Persistent Trigger	+	+	+	+

of branches, such as the Multi Choice (Wcp-06) and Structured Synchronizing Merge (Wcp-07).

Relatively to the Multiple Instance patterns, we can observe that WS-BPEL and Kepler are only able to support the M.I. without Synchronization (Wcp-12) pattern, while BPMN and YAWL provide wider support for such patterns, especially these languages are also able to synchronize multiple instances of the same task.

The major difference in the State-Based patterns is the support for the Deferred Choice (Wcp-16) offered by the three business WfMSs, and for the Critical Section (Wcp-39) offered by WS-BPEL and YAWL. While for the Cancelation and Force Completion patterns the lack of support found in Kepler is in contrast with the support offered by the three business WfMSs for many of them.

Finally, the support for Termination, Iteration and Trigger patterns is substantially the same in all the systems.

6.1.2 Workflow Data Patterns Evalutation

Workflow Data Patterns (WDPs) [108] capture those language features that are useful for describing and managing data resources during process execution. In Kepler data are carried only by data tokens and there are no shared variables. The realization of each data pattern in Kepler is analyzed in the following sections, while Tab. 6.2 compares the obtained results with the support offered by the chosen business WfMS representatives.

(G1) Data Visibility Patterns

Data visibility patterns identify potential contexts in which a data construct can be defined and utilized. In scientific WfMSs data are contained only inside tokens, variables are local to each task instance, there are no global variables and tasks can only communicate by providing a value to their output ports and reading a value from their input ports. Therefore, Kepler supports the Task Data pattern (Wdp-01) and the Multiple Instance Data pattern (Wdp-04). These patterns cater for the definition of data elements accessible only within an execution instance of a task, that is eventually able to execute multiple times inside the same case. They are supported since the variables used by tasks are specific to each individual execution instance. Moreover, the Environment Data pattern (Wdp-08), which allows a workflow instance to access data elements from the external operating environment, is directly supported via several predefined tasks that allow access to a local or remote database, or a local or remote file system (e.g. Kepler's actors `Database Writer` or `Directory Listing`).

(G2) Data Interaction Patterns

Data interaction patterns examine between which types of components data can be exchanged and which component takes the initiative. Data can be exchanged within the same process (internal data interaction patterns), or with the external environment (external data interaction patterns).

In scientific WfMSs tasks can communicate only through channels that connect the output port of a task with the input port of another task, and this holds also for a composite task and its sub-workflow decomposition. Therefore, the patterns directly supported are: Task to Task (WDP-09), regarding the communication between tasks of the same case, Block Task to Sub-Workflow Decomposition (WDP-10), regarding the communication between a composite task and its sub-workflow tasks, and Sub-Workflow Decomposition to Block Task (WDP-11), regarding the inverse communication between sub-workflow tasks and a block task. In particular, in WDP-09 the same channels are used to pass both control-flow and data tokens (*integrated control and data channels* [108]), as control-flow is directly specified in terms of data dependencies; while for WDP-10 and WDP-11 the communication between a block task and its component, or vice-versa, is also performed through channels (*explicit data passing via data channels* [108]).

Patterns Data Interaction - to Multiple Instance Task WDP-12 and Data Interaction - from Multiple Instance Task WDP-13 capture data passed to/from a multiple instance task. In Kepler a new task instance is spawn as soon as the required input data is available. This instance receives distinct input data through tokens and works on its own data, without side effects for the other instances, while the output produced by each task instance is queued in the outgoing channel. Therefore, WDP-12 is directly supported through the approach classified as *instance-specific data passed by value or by reference* [108]. Conversely, WDP-13 cannot be considered directly supported, because it requires the aggregation of data elements produced by the various instances before passing them to the subsequent task. This problem is correlated with the impossibility of synchronizing multiple instances of the same task at completion. Finally, Kepler does not provide a mechanism to communicate between two different instances of the same workflow (Case to Case WDP-14). The interaction with the external environment can only be initialized by the task, case or workflow which can request or provide external data, for instance by writing the computed results into an external database or a local or remote file system. Therefore, the patterns Task/case/workflow to environment - push oriented (WDP-15, WDP-19, WDP-23) and Environment to task/case/workflow - pull oriented (WDP-16, WDP-20, WDP-24) are supported; while the patterns Task/case/workflow to environment - pull oriented (WDP-18, WDP-22, WDP-26) and Environment to task/case/workflow - push oriented (WDP-17, WDP-21, WDP-25) are not directly supported.

(G3) Data Transfer Patterns

Data transfer patterns consider the manner in which the actual transfer of data elements occurs between one process component and another. In scientific WfMSs data are passed to each task by means of tokens on channels that connect output and input ports. Each task operates on a copy of the received values, thus the patterns Data Transfer by Value - Input (WDP-27) and Data Transfer by Value - Output (WDP-28), regarding the ability to receive or send data elements by value, are directly supported. Moreover, the received data can also be the name of a file or the address of an external (remote) resource, and generally no concurrency restrictions are applied to the shared data. Therefore, the pattern Data Transfer

by Reference - Unlocked (WDP-30), considering the communication between tasks by passing a reference to the location of the data elements, is supported, while the pattern Data Transfer by Reference - With Lock (WDP-31), which also requires the ability to define privileges restrictions or dedicated access, is not directly supported. A task can copy into its local address space the data value collected by an external resource (e.g. a remote database), providing support for the Data Transfer - Copy In / Copy Out pattern (WDP-29), which captures the ability to copy the values of a set of data elements from an external source to an address space local to the task. Finally, the data transformation patterns (WDP-32, WDP-33), which capture the ability to apply some transformations on data prior to pass these to/from a component, are not supported: data can only be transformed by a task during its execution and not immediately prior or after its execution.

(G4) Data-based Routing Patterns

Data-based Routing Patterns capture the various ways in which data elements can interact with other perspectives and influence the overall operation of a process instance [108].

In scientific WfMSs data availability drives the computation, but data dependencies influence only task activation, not its completion. Therefore, the pattern Task Precondition - Data Existence (WDP-34), requiring the presence of some data at the time of a task execution, is supported, while the pattern Task Postcondition - Data Existence (WDP-36), requiring the presence of some data at task completion, is not supported. Moreover, data dependencies only concern the availability of data and not their value, thus the patterns Task Precondition - Data Value (WDP-35) and Task Postcondition - Data Value (WDP-37), requiring a particular value for specific parameters at the time of execution, respectively, of completion, are not supported. As per the data interaction patterns, interactions with the external environment can only be triggered by a task within the workflow and there is no way for an external event to initialize a task. Therefore, pattern Event-based Task Trigger (WDP-38), representing the ability for an external event to initiate a task, is not supported. Similarly, a task is enabled only when the necessary inputs are available, and no constraints can be specified about their values. Hence, Data-based Task Trigger (WDP-39), which allows the triggering of a specific task when an expression evaluates to true, is also not supported. Finally, the Data-based Routing pattern (WDP-40), capturing the ability to alter the control-flow based on the evaluation of data expressions, is supported by Kepler with the same mechanism as the one described for the Exclusive Choice (WCP-04) and the Multi Choice (WCP-06) patterns.

Final Observations

Table 6.2 summarizes the WDPs support of Kepler and compares it with YAWL, BPMN 1.0 and WS-BPEL 1.1.

Business WfMSs provide some additional way to define and use data constructs, for instance at the case or scope level, while in Kepler data are mainly local to each task instance and there are no shared variables.

Table 6.2. Data patterns support in the considered WfMSs. The first column denotes the patterns group, G1: data visibility, G2: data interaction, G3: data transfer and G4: data-based routing.

	Pattern	BPMN	BPEL	YAWL	Kepler
G1	WDP-01 Task Data	+	±	+	+
	WDP-02 Block Data	+	-	+	-
	WDP-03 Scope Data	-	+	+	-
	WDP-04 Multiple Instance Data	±	-	+	+
	WDP-05 Case Data	+	+	+	-
	WDP-06 Folder Data	-	-	+	-
	WDP-07 Workflow Data	-	-	+	-
	WDP-08 Environment Data	-	+	+	+
G2	WDP-09 Task to Task	+	+	+	+
	WDP-10 Block Task to Sub-Workflow Decomp.	+	+	+	+
	WDP-11 Sub-Workflow Decomp. to Block Task	+	+	+	+
	WDP-12 to Multiple Instance Task	+	+	+	+
	WDP-13 from Multiple Instance Task	-	-	+	-
	WDP-14 Case to Case	-	-	+	-
	WDP-15 Task to Environment – Push-Oriented	+	+	+	+
	WDP-16 Environment to Task – Pull-Oriented	+	+	+	+
	WDP-17 Environment to Task – Push-Oriented	-	-	+	-
	WDP-18 Task to Environment – Pull-Oriented	-	-	+	-
	WDP-19 Case to Environment – Push-Oriented	+	+	+	+
	WDP-20 Environment to Case – Pull-Oriented	+	+	+	+
	WDP-21 Environment to Case – Push-Oriented	-	-	+	-
	WDP-22 Case to Environment – Pull-Oriented	-	-	+	-
	WDP-23 Workflow to Environment – Push-Or.	+	+	+	+
	WDP-24 Environment to Workflow – Pull-Or.	+	+	+	+
	WDP-25 Environment to Workflow – Push-Or.	-	-	+	-
	WDP-26 Workflow to Environment – Pull-Or.	-	-	+	-
G3	WDP-27 Data Transfer by Value - Incoming	+	+	+	+
	WDP-28 Data Transfer by Value - Outgoing	+	+	+	+
	WDP-29 Data Transfer - Copy In/Copy Out	+	+	+	+
	WDP-30 Data Transfer by Ref. - Unlocked	+	+	+	-
	WDP-31 Data Transfer by Ref. - With Lock	-	-	+	±
	WDP-32 Data Transformation - Input	-	-	+	+
	WDP-33 Data Transformation - Output	-	-	+	+
G4	WDP-34 Task Precondition - Data Existence	+	+	+	+
	WDP-35 Task Precondition - Data Value	-	-	+	+
	WDP-36 Task Postcondition - Data Existence	-	-	+	+
	WDP-37 Task Postcondition - Data Value	-	-	+	+
	WDP-38 Event-based Task Trigger	-	-	+	+
	WDP-39 Data-based Task Trigger	-	-	+	+
	WDP-40 Data-based Routing	+	-	+	+

Relatively to the data interaction patterns, we can observe that business WfMSs allow some form of interactions in which the external environment can proactively offer data to a task, has happens in Environment to Task – Push Oriented (WDP-17), or in Task to Environment – Pull Oriented (WDP-18), while in Kepler only the tasks inside the workflow can start the connection with the external environment.

BPMN and YAWL are able to perform some transformations before passing a data element to/from a task. Moreover, the three considered business WfMSs provide some ways to prescribe a mutual exclusive access to data.

Finally, as regards to the data-based routing patterns, the three business WfMSs under consideration allow the definition of task triggers at the event or data level, while Kepler is not able to deal with triggers. Moreover, BPEL and YAWL allow the definition of task preconditions based on data values, while Kepler allows only the definition of task preconditions based on data existence. BPMN and YAWL allow also the definition of task postconditions, while in Kepler no postcondition can be specified neither on data existence nor on data value.

6.1.3 Workflow Resource Patterns

Workflow Resource Patterns (WRPs) [111] capture the various ways in which resources (e.g. human or computational resources) are represented and used in workflows. Scientific WfMSs consider processes that are usually enacted by only one user at a time, thus they do not have to manage different agents or different roles with related authorization and authentication issues [82]. Moreover, little or no user interaction is needed to perform an activity, no work is assigned to human agents and human intervention is usually limited to perform run-time decisions. As a result, only few resource patterns are supported, as discussed in the following and summarized in Tab. 6.3.

(G1) Creation Patterns

Creation Patterns correspond to limitations on the manner in which a work item may be executed. They are specified at design time, usually in relation to a task, and serve to restrict the range of resources that can undertake work items that corresponding to a task [111]. As stated above, the considered scientific WfMSs do not provide a mechanism for identifying and distinguishing resources, in particular with respect to human agents. Work items are automatically executed as soon as they have the necessary input without the need to be explicitly allocated to a particular resource. Therefore, only Automatic Execution (WRP-11) is supported.

(G2) Push Patterns

Push Patterns characterize situations where newly created work items are proactively offered or allocated to resources by the system. These may occur indirectly by advertising work items to selected resources via a shared work list or directly with work items being allocated to specific resources [111]. In Kepler work items are directly allocated by the system and executed as soon as they are enabled

by the availability of the necessary inputs. Therefore, the only supported push pattern is Distribution on Enablement (Wrp-19).

(G3) Pull Patterns

Pull patterns correspond to the situation where individual resources are made aware of specific work items, that require execution, either via a direct offer from the system or indirectly through a shared work list. The commitment to undertake a specific task is initiated by the resource itself rather than the system [111]. In Kepler it is the director that schedules the execution of work items and these automatically start to execute when the necessary inputs are available. It follows that none of the pull patterns are supported by the three systems.

(G4) Detour Patterns

Detour patterns refer to situations where work item distributions that have been made for resources are altered either by the system or at the instigation of the resource. As a consequence of this event, the normal sequence of state transitions for a work item is varied [111]. Examples of these patterns are the delegation or escalation of a work item to a given resource, or the deallocation of a work item from a resource, so that the work item is offered again. These patterns are not supported because work items are automatically executed and cannot be intentionally suspended, re-routed and re-allocated.

(G5) Auto-Start Patterns

Auto-start patterns relate to situations where execution of work items is triggered by specific events in the lifecycle of the work item or the related process definition. Such events may include the creation or allocation of the work item, completion of another instance of the same work item or a work item that immediately precedes the one in question [111].

The Commencement on Creation (Wrp-36) pattern is directly supported by Kepler, it refers to the ability of a resource to commence execution on a work item as soon as this is created. In Kepler a work item is executed immediately after its enablement.

(G6) Visibility Patterns

Visibility patterns classify the various scopes in which work item availability and commitment are able to be viewed by resources [111]. These patterns are not supported: Kepler does not provide a facility to visualize the list of available or committed work items.

Table 6.3. Support of the WRPs by the exisinting business WfMSs considered in this thesis.

	Pattern	BPMN	BPEL	YAWL	Kepler
G1	WRP-1 Direct Distribution	+	+	+	-
	WRP-2 Role-base Distribution	+	+	+	-
	WRP-3 Deferred Distribution	-	+	+	-
	WRP-4 Authorization	-	-	+	-
	WRP-5 Separation of Duties	-	-	+	-
	WRP-6 Case Handling	-	+	-	-
	WRP-7 Retain Familiar	-	+	+	-
	WRP-8 Capability-based Distribution	-	+	+	-
	WRP-9 Organization-based Distribution	-	±	+	-
	WRP-10 History-based Distribution	-	±	+	-
	WRP-11 Automatic Execution	+	+	+	+
G2	WRP-12 Distribution by Offer – Single Reseouce	-	+	+	-
	WRP-13 Distribution by Offer – Multiple Resources	-	+	+	-
	WRP-14 Distribution by Allocation – Single Resource	+	+	+	-
	WRP-15 Random Allocation	-	±	+	-
	WRP-16 Round Robin Allocation	-	±	+	-
	WRP-17 Shortest Queue	-	±	+	-
	WRP-18 Early Distribution	-	-	-	-
	WRP-19 Distribution on Enablement	+	+	+	+
	WRP-20 Late Distribution	-	-	+	-
G3	WRP-21 Resource-Initiated Allocation	-	-	+	-
	WRP-22 Resouce-Initiated Execution – Allocated	-	+	+	-
	WRP-23 Resource-Initiated Execution – Offered	-	+	+	-
	WRP-24 System Determined Work Queue Content	-	-	+	-
	WRP-25 Resource Determined Work Queue Contet	-	+	+	-
	WRP-26 Selection Autonomy	-	+	+	-
G4	WRP-27 Delegation	-	+	+	-
	WRP-28 Escalation	-	+	+	-
	WRP-29 Deallocation	-	+	+	-
	WRP-30 Stateful Reallocation	-	+	+	-
	WRP-31 Stateless Reallocation	-	-	+	-
	WRP-32 Suspension/Resumption	-	+	+	-
	WRP-33 Skip	-	+	+	-
	WRP-34 Redo	-	-	-	-
	WRP-35 Pre-Do	-	-	-	-
G5	WRP-36 Commencement on Creation	+	-	+	+
	WRP-37 CommencementOnAllocation	-	-	+	-
	WRP-38 Piled Execution	-	-	+	-
	WRP-39 Chained Execution	+	-	+	+
G6	WRP-40 Configurable Unallocated Work Item Visib.	-	-	+	-
	WRP-41 Configurable Allocated Work Item Visibility	-	-	+	-
G7	WRP-42 Simultaneous Execution	+	+	+	+
	WCP-43 Additional Resources	-	+	-	+

(G7) Multiple Resource Patterns

Multiple Resource patterns capture the ability of a resource to simultaneously work on different work items (Simultaneous Execution - Wrp-42) and the ability of a resource to request additional resources to assist in the execution of a work item currently undertaken by that resource (Additional Resources - Wcp-43). In each of the three systems the same resource can work on different work items simultaneously, hence the Simultaneous Execution pattern is supported. However, the Additional Resources pattern is not supported.

Final Observations

The resource pattern analysis confirms that Kepler provides a limited attention to human coordination aspects. In particular, tasks are automatically executed and there is no way to model task assignment or delegation. As previously mentioned, this is due in part to the fact that scientific WfMSs have been originally developed for automating large scale experiments, and in part to the relative immaturity of these systems. Conversely, the system that provides most attention to agents coordination is YAWL which support almost all patterns.

6.2 Requirements of a Geo-Processing WfMS

In [105] the authors recognise a new emerging approach in the construction of SDIs across the world. They observe a shift from the first generation, which was production oriented and focused on the development of integrated databases, to the second generation, which is more process oriented and emphasizes partnership and stakeholders involvement. In this emerging context the coordination of the involved agents has become one of the most important aspects to consider.

The main question remains if SDIs are distinct from other kind of information infrastructures, like those for the health care, namely if they have some new and peculiar requirements that prevent the adoption of standard solutions. SDIs are special mainly because they apply specialised tools and concepts for handling spatial data, and because their implementation and use require understanding of basic geographic and cartographic principles [95]. SDIs rely both on spatial data and on the technology and concepts to handle these data. Hence, they may be different from other information infrastructures, but are these differences fundamental?

The following sections analyze the adoption of existing workflow technologies for supporting distributed geo-processing activities. In particular, it considers the benefit of exploiting existing WfMSs for the design and implementation of the integration framework presented in the first part of this thesis.

6.2.1 Integration Process Implementation with BPMN

Most of the benefits of PAISs came from the ability to interpret high-level process specifications, separating the business logics from the software system logics: when a change occurs in the organization functioning, due to contingent needs or

a proactive application of the BPM approach, the new processes are documented and translated into operational activities. An information system can be easily realigned to the new conditions by simply changing the interpreted process specifications and implementing the missing functionalities as services. Actually, this theoretical alinement is now far from real: an huge gap exists between the documentation of operational processes and their executable specifications, and this is even worse in GIS construction.

In order to exemplify such situation, let us consider the integration process proposed in Part I, whose BPMN representation is depicted in Fig. 6.5. The process is triggered by a start message which communicates the availability of new or updated information. On the basis of the relation between the existing database and the new one, the integration case is determined. If the determined scenario is valid, namely is one of those defined in Sec. 4.1, the process continues, otherwise it is terminated. In the first case, the execution continues with the integration of the thematic classes and of the objects; then, on the basis of the previously determined integration scenario a choice is taken between a union integration, or a more complex Kalman integration. Such condition evaluation is represented by the first empty diamond symbol. In the second case, two parallel computations are started: one related to the position integration, and the other regarding the computation of variance-covariance information for the result. Diamond symbols containing a cross enclose parallel activities, notice that these two parallel branches are synchronized at a certain point, because the computation of the Kalman matrix is required by both parallel computations. After the metric integration has been performed, the integration of logical information is executed and the generated inconsistencies are determined. Such inconsistencies are solved by the final loop, which iterates until new inconsistencies are found. Finally, the result is assembled using the computed information and sent to another process being stored.

The actual process is more complex than the one depicted in Fig. 6.5: for sake of simplicity many details have been omitted and some aspects have not been included at all, for instance exception handling and compensation. For obtaining a specification sufficiently detailed to be executed in some sort of WfMS, some incremental refinements have to be applied to the process model in Fig. 6.5. However, obtaining an executable specification from such model is not a simple matter: a huge gap exists between the documentation of operational processes and their executable specifications. The first unavoidable refinement is the definition of data necessary at run-time. A BPMN model can be annotated with data aspects necessary at run-time; however, the data model is fairly primitive and inadequate for describing data-intensive processes, like the geographical ones. Moreover, let us notice that in order to describe a minimal set of data-flow interactions between the integration process and another process which initially provides the updated information, and finally stores the computed result, these processes have been placed into two different pools, even if this construct was originally intended for representing interactions among processes that belong to different organizations. This is because a message flow can never be used to connect activities or events within the same pool. For modelling purposes, some authors [141] suggest to break the standard and use data-flow interactions whenever needed, even if they are not well defined.

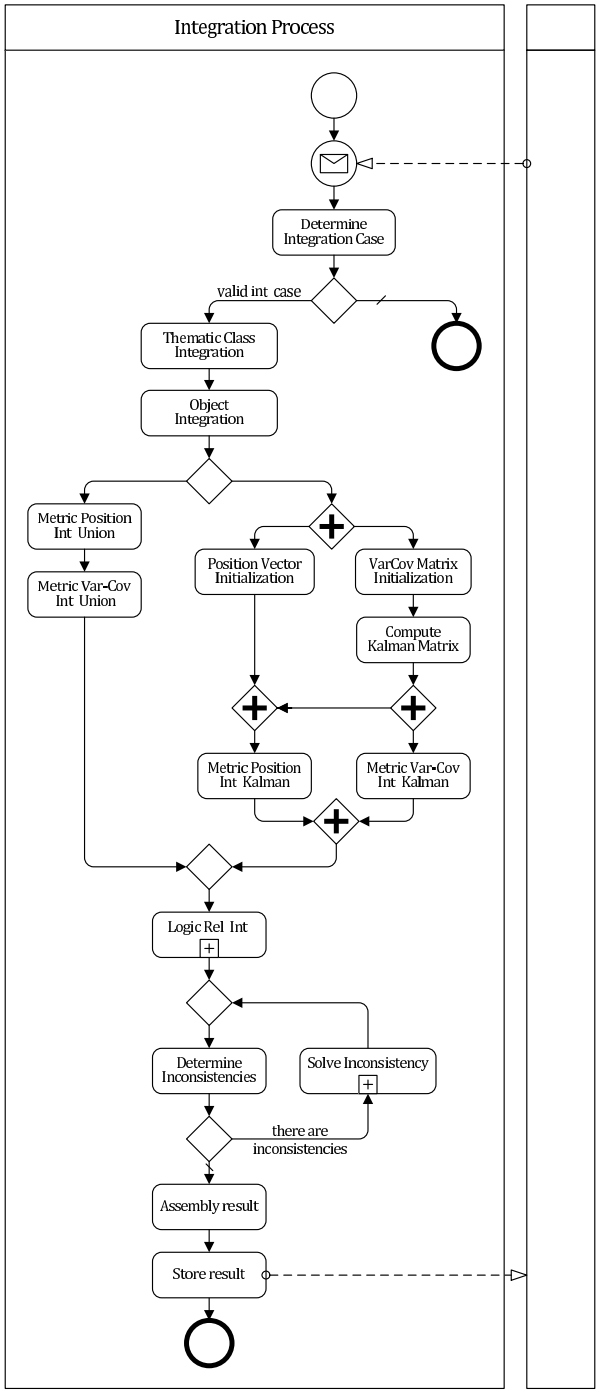


Fig. 6.5. Integration process design in BPMN.

As stated in Sec. 5.4.2, BPMN is commonly used a graphical representation for WS-BPEL, but not all the BPMN constructs can be translated in WS-BPEL and viceversa [101, 139]. For instance, the process in Fig. 6.5 will be translated into different uncorrelated WS-BPEL processes. Moreover, WS-BPEL is far from be an high-level language, its strengths reside on creating new services by assembling existing ones, not to deal with humans, hence any user interaction described in BPMN cannot be directly translated in WS-BPEL.

As a consequence, many WfMSs leverage existing technologies for providing advanced tools that can fulfill the gap. The following section considers the use of PAISs and scientific WfMSs for implementing the integration process. The example proposed in this section and in the following two ones constitute a primary study on the applicability of existing workflow systems for geo-processing. The processes have not be completely realized, due to the limits found in the considered systems. The main purpose of this examples are to define the characteristics of an ideal solution which will be described in Sec. 6.3.

6.2.2 Integration Process Implementation with YAWL

As explained in the previous chapter, YAWL provides a good framework for managing human resources, coordinating their works and monitoring the process evolution, while it is less suitable for performing long-running intensive computations dealing with complex data structures.

The YAWL specification of the integration process is depicted in Fig. 6.6, notice that routing constructs (splits and joins) can only be associated with a task, which can eventually be a dummy automatic task that does not compare in any work list. Such tasks are depicted in gray in the diagram of Fig. 6.6.

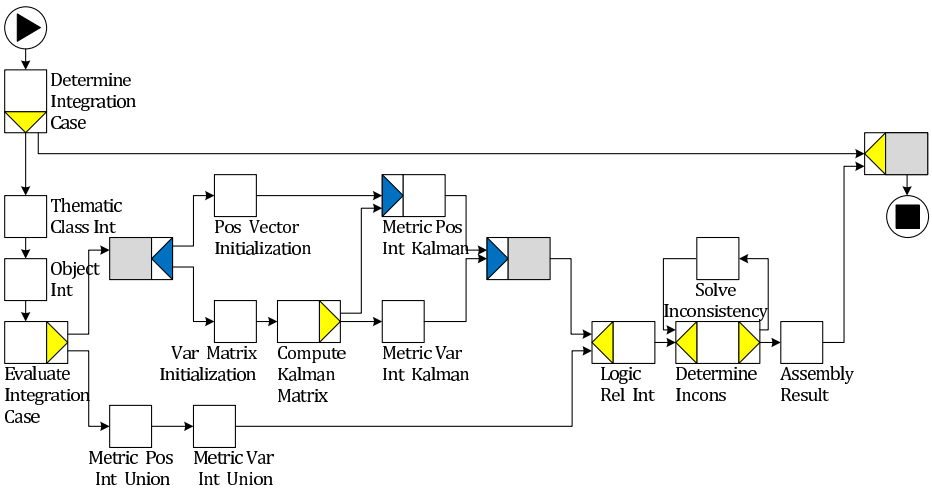


Fig. 6.6. Integration process design in YAWL.

The first mentioned strength regards the management of user privileges through the definition of a role hierarchy. However, relatively to this aspect, YAWL allows

one to define privileges only at task level, but in the considered example we have the need to specify user privileges also at data level: the same task can be performed by different participant but only with reference to a certain portion of the territory. For instance, for solving inconsistencies related to topological relations, some changes on the input geometries can be necessary, in this case only the user responsible for the particular object can decide how to perform the changes.

The second difficulty in the implementation of the considered process regards the data management. In YAWL variables can be represented using XML or by connecting the process specification to a PostgreSQL database. Even if GML is an XML-based language and PostGIS is a PostgreSQL extension for spatial data management, a problem remains: conditions on workflow variables are expressed using XPath and no spatial conditions can be specifies using this language (e.g. based on topological relations). Moreover, in the considered scenario it should be useful to natively represent more complex data structures, such as those regarding MACS positions, hard topological relations, and using a database management system means escaping from the language constructs, breaking the available validation facilities.

Moreover, even if the YAWL system can automatically generate a graphical user interface for the distributed application, no support is provided for the visualization and management of spatial information. This is a real limitation, since in the geographical field most of the user operations are performed through the interaction with a map.

6.2.3 Integration Process Implementation with Kepler

The last considered WfMS is Kepler, which has been taken as a reference for scientific WfMSs and the data-flow paradigm. As highlighted in the previous chapter, it is more suitable for representing repetitive and intensive computation on huge amount of data, but it provides few facilities for managing human resources and the interaction with multiple users that concurrently interact.

The Kepler implementation of the integration process is depicted in Fig. 6.7. The behaviour of the *boolean switch* (e.g. between the actor `Evaluate Integration Case` and actor `Thematic Class Integration`) can be explained as follows: it receive from the white input multi-port any type of token which is redirected to the true (`T`) or false (`F`) output port, on the basis of the value of the control token received from the other input port. Let us notice that thanks to the implicit data parallelism offered by this system, as soon as the actor `LogicRelInt` produces an integrated topological relation, an instance of `DetermineInconsistency` can start its execution. Therefore, many instances of this actor can be in execution while the integration of topological relations is still in progress.

Unfortunately, Kepler does not provide any support for the human resource management and coordination. The fundamental assumption in this kind of workflows is that an experiment is executed by only one scientist at time. Conversely, processes performed inside an SDI typically involve several agents with different competencies and roles. Moreover, the support for human activities is very limited, it is usually reduced only to provide an input to an automated activity or perform a choice among several alternatives. Therefore, the implementation of the

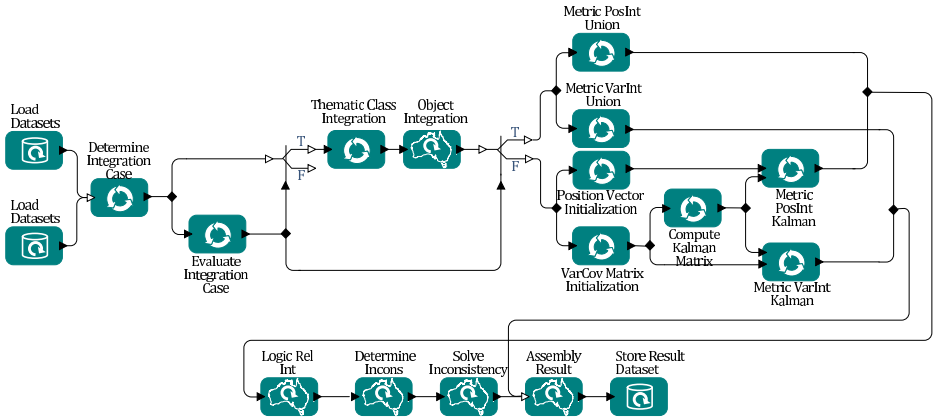


Fig. 6.7. Integration process design in Kepler.

necessary interactions during the integration process have to be programmatically solved, escaping the available language constructs.

6.3 The Ideal Solution

Previous sections have exposed the strength and weakness of existing WfMSs as tools for implements flexible GISs. Even if these systems offer attractive ready-to-use features, none of them seems to be completely satisfactory from an architectural and functional points of view. Not surprisingly, these limitations present some similarities with those found in the use of classical relational databases for storing spatial data:

- lack of a geometric model,
- lack of tools supporting spatial data visualization and directly interaction through maps,
- difficulties in exploiting spatial data characteristics during geo-processing.

As regards to the first point, spatial data have several unique characteristics compared with data processed in other fields, as deeply analysed in [41] and recalled in Chap. 2. The main one is the presence of one or more geometric attributes that require the existence of an underlying geometric model, namely the definition of some geometric types and operations. Therefore, the modeling of spatial data requires to augment existing WfMSs with a support for geometric data models.

The visualization of spatial data in forms of maps is an essential characteristics of any GIS application. A numeric representation of spatial object has no meaning for a user, as the information is given by the shape and positions of the objects. Moreover, users usually perform their operation by manipulating a map representation and not directly object coordinates. Most of the available systems provides a web-based interface, due to the involved web-based technologies and the easiness of deploy. These interfaces have to be enhanced with functionalities for the visualization and interaction with spatial data through maps: the advances

in web technologies make web interfaces even more rich, but they can be limited for manipulating an huge amount of vector data in an effective and precise way.

These first two points require some improvement of the existing systems, but they are not particularly hard to solve. The real challenge resides on the last point regarding the processing of spatial data. In this kind of processing two forms of parallelism can be distinguished: functional and data decomposition [57]. Functional decomposition techniques deal with the distribution of operations among the available resources. The complexity of some analysis requires that services addressing isolated tasks are combined into multiple processing steps to achieve the desired results. Therefore, functional parallelism can be achieved by decomposing complex computation activities into smaller parts (tasks), defining the execution order of these operations and the dependencies between them, so that some operations can be performed in parallel while others in sequence. On the other hand, data decomposition techniques subdivide data into independent chunks, so that multiple instances of the same activity can be executed in parallel on different inputs. The partial results produced by each activity instance are finally combined to form the overall output.

As highlighted before, PAISs are more suitable for representing the functional decomposition of processes and the interactions among different agents. At any step the workflow engine can monitor the overall state of the process, which activities have been performed and which are executing. On the other hand, this kind of systems provides a poor support for the data decomposition which is essential for intensive computations. Optimizations at such level have to be addressed in an ad-hoc manner by the underlying software layer. Conversely, scientific WfMSs have been developed for supporting intensive computations and can easily exploit the use of Grid technologies in a transparent way for the user. Scientific workflow systems have been studied to simplify the composition of computational blocks and hence they offer a better chance to provide a library for geo-processing. However, some problematic aspects remain, for instance how to provide to the user a complete overview of the process execution, or how to seamlessly integrate cross-cutting concerns like resource management. GIS engineering needs WfMSs able to combine the strengths of PAISs and scientific WfMSs with a model of computation suitable for geo-processing.

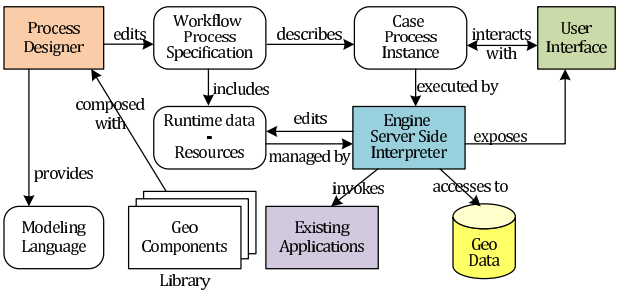


Fig. 6.8. Architecture of a geo-processing WfMS.

The architecture of a possible WfMS for geo-processing is depicted in Fig. 6.8. As regards to the needed extensions, the first one is the development of a set of components to handle geographical operations. For instance, Kepler allows one to perform spatial operations by invoking web-services and some research has been done for integrating external dedicated applications, like GRASS [149]. However, as discussed in [133], this solution limits workflow transparency, thus it is preferable to enhance a WfMS with a core library of GIS functionalities. Many APIs providing geographical functions have been developed, such as the Java Topology Suite [136]. Anyway, the functionalities offered by these libraries can be currently combined only in a programmatic way; the idea is to use them for developing a set of pluggable and standard-based components (e.g. ISO TC211) that can be graphically assembled by users with limited programming skills.

Moreover, in order to exploit the data parallelism offered by scientific WfMSs, some specific functions have to be developed for partitioning spatial data into independent chunks of information and combining the partial results. The specific characteristics of spatial data pose some additional constraints in the development of such functions: (1) spatial data do not follow a preconceived pattern, but it can vary considerably in complexity across a dataset. (2) Most attention is usually given to the decomposition of data, but in the geographical domain the combination phase is equally important and in some cases even more difficult, since during the reconstruction many properties, like topological relations, have to be preserved.

Relatively to the computational aspect, Kepler supports many different computational models, some of them specifically developed for modeling dynamic physical systems. However, in the geographical domain not all these computational models are necessary. We can safely restrict to the one in which components run in parallel exchanging data through channels of ideally unbounded capacity, known as Process Networks [149], and enhance it with coarse-grained control-flow constructs, in order to express the control-flow logics that emerges from fine-grained data-flow relations. This can be useful for mitigating the main drawback of using a data-driven approach with respect to the control-flow one offered by PAISs, namely the loss of easy to grasp information about the overall process execution.

Finally, the integration of cross-cutting concerns like security and resource management plays a central role. Relatively to this aspect, the engine has to be decoupled from the user interface that will become part of a distributed application for supporting the collaborative execution of workflows by several agents in different locations.

6.4 Summary and Concluding Remarks

This chapter has compared the chosen WfMS representatives first from a general point of view, using the workflow pattern methodology, and then from a geographical point of view, considering the proposed integration process as use case.

In the BPM community, workflow patterns are a framework frequently used to compare the suitability of WfMSs in describing business processes. Many existing offerings have been evaluated in literature using this tool. In this thesis we present

the evaluation of Kepler and compare it with respect to the evaluation of the other considered WfMSs contained in [127, 145, 146]. We observe that in some cases a pattern is not directly supported by Kepler because it is considered not relevant for that application domain; in other cases, it can be useful but it is not supported probably due to the relative immaturity of the system. In particular, this system does not support many of the resource patterns related to the coordination of human agents, this is one of the main weakness of the system in designing geographical processes with the characteristics highlighted in Chap. 1. Another importation limitation regards the management of multiple instance tasks, if from one hand the adopted computational model allows one to transparently exploit parallelism, since multiple instances of the same task can concurrently execute on different data, the synchronization of these instances can be quite difficult.

As regards to the geographical comparison, we can observe that processes in the geographical field can benefit from both the adoption of business and scientific WfMSs, but none of them is completely satisfactory. The found limitations regard three different aspects: modeling, visualization and processing of spatial data. The latter is the most serious one, because it cannot be solved by simply adding features to existing WfMSs. The interactive nature of long-running geo-processing activities, the importance of domain expert knowledge in driving the computation and the need to coordinate the effort of different agents, are better addressed by business WfMSs. On the contrary, scientific WfMSs can provide a support for intensive long-running computations required by geo-processes. The ideal WfMS for geo-processing has to combine the characteristics of both approaches in a coherent system. In particular, the data-flow computation model adopted by scientific WfMSs enhanced with coarse-grained control-flow structures can be the most suitable solution for geo-processing. For these reasons, the last chapter presents a different solution, based on a novel modeling language, called NestFlow, which combines both approaches in a coherent way.

7

The NestFlow Solution

The previous chapter evaluates the applicability of workflow technologies for developing software systems that support distributed geo-processing. The conclusion is that none of the available systems is completely satisfactory: the ideal WfMS for geo-processing has to combine the characteristics of both business and scientific WfMSs in a coherent way. For this purpose, this chapter introduces a different modeling language, called NestFlow, which has been conceived to explore a particular PML design solution in which structured control-flow constructs are tightly-coupled with Asynchronous Message Passing (AMP) connections. This is clearly a different approach from the established CFO solutions, in which unstructured control-flow constructs are coupled with parameter passing and shared task variables. In NestFlow control-flow constructs can be composed only in properly nested structures and task variables cannot be shared among concurrent entities. In this way, the use of AMP is made mandatory in the right place, i.e. to describe the interaction among concurrent entities. Data-flow constructs are promoted as first-class citizens because they are invaluable for offering a uniform hierarchical decomposition mechanism that increases modularity. Conversely, data-flow constructs are a marginal feature in existing CFO languages that focus mainly on control-flow, in an attempt to design simpler languages. The NestFlow rationale is to fuse control-flow and data-flow aspects for offering a structured control-flow without any loss of expressiveness and with positive effects on modularity.

This PML has been developed by our research group in the context of another PhD thesis, and has been extended and applied in this thesis for designing and developing the proposed framework. In addition to the language presentation, this chapter discusses the design of the integration process presented in Part I using this language. The structure of the chapter is as follows: Sec. 7.1 introduces the control-flow aspects of the language, while Sec. 7.2 presents the data-flow ones. Sec. 7.3 summarizes two important properties for well-formed NestFlow models and Sec. 7.4 explains the main semantics of the NestFlow constructs. Finally, Sec. 7.5 discusses the realization of the proposed integration process in NestFlow.

7.1 Control-Flow Aspects

This section introduces the basic graphical elements of NestFlow and its syntax. NestFlow elements are obtained from graphical primitives, like rounded boxes, diamonds, or lines, connected together in heterogeneous groups called `blocks`. In contrast with existing PMLs, each block is intended to be manipulated as a whole by a graphical editor that can also enforce the essential syntactical rules. The NestFlow blocks are summarized in Fig. 7.1 using a BNF-like notation that encodes the basic compositional rules. In particular, a rule $\langle S \rangle ::= X|Y|Z$ means that the meta-symbol $\langle S \rangle$ can be replaced by one construct chosen among X, Y or Z.

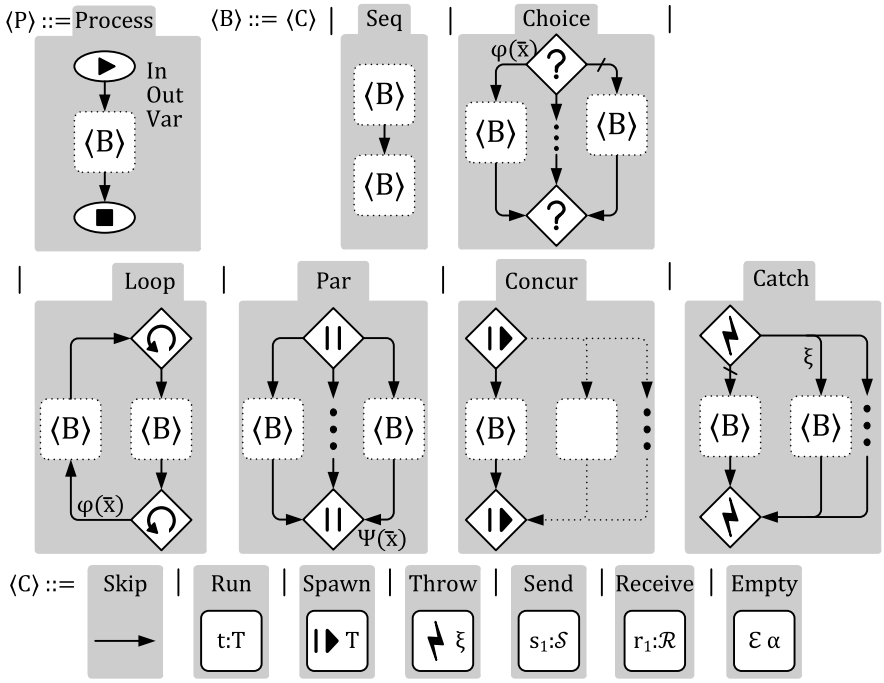


Fig. 7.1. The NestFlow modeling language constructs.

In Fig. 7.1 three main syntactical categories can be distinguished: the main block $\langle P \rangle$, called `Process`, non-terminal blocks $\langle B \rangle$, `Seq`, `Choice`, `Loop`, `Par`, `Concur`, and `Catch`, and terminal blocks $\langle C \rangle$, `Skip`, `Run`, `Spawn`, `Throw`, `Send`, `Receive`, and `Empty`. In the following the term *block* may refer to both terminal and non-terminal blocks, while the term *command* is used as an alias for terminal blocks. Excluding the last three commands that deal with data, the other blocks are used to describe the main control-flow relations among tasks.

A *task* is essentially a more or less complex pre-existing process specification invoked using a `Run` command or created at run-time with a `Spawn` command. An invoked process can be a *native* procedure implemented with a general-purpose

programming language or any other previously defined NestFlow process model. The ability of reusing an existing process specification is essential for supporting a uniform hierarchical decomposition that in turn enhances the overall system modularity. A native procedure can be used to implement an automatic task, drive an external program or interact with human agents by means of a graphical user interface.

A process specification is obtained starting from the main process $\langle P \rangle$ and by recursively nesting multiple blocks $\langle B \rangle$ and commands $\langle C \rangle$ following the grammar in Fig. 7.1 and some additional syntactical rules that are explained in Sec. 7.3. A sequence of blocks of arbitrary length n can be obtained by nesting $n-1$ `Seq` blocks. The limit of only two children has been introduced to simply the exposition, but nothing prevent one to define a more general construct with multiple children. `Choice` and `Par` blocks can have two or more branches. Each branch i of a `Choice` block is annotated with a condition $\varphi_i(\bar{x})$ over a set of variables $\bar{x}$, except the last one which is called default branch and is marked with an oblique bar /. A `Par` block is used to execute in parallel two or more inner blocks that have to synchronize at the end of the block before leaving it. A `Loop` block has two branches, both of them can be expanded with further inner blocks or they can be also used alone closing the other branch with a `Skip` command. `Concur` is a block with initially two branches, one depicted as a solid line that represents the main flow, and one with a shaded line that denotes the possibility of adding one more branches at run-time with a `Spawn` command, as explained in Sec. 7.4. NestFlow is designed not only for creating process models but also for displaying their execution: the `Concur` block offers a basic mechanism to make visible the creation and destruction of dynamic entities by growing and shrinking the displayed model instance. PMLs usually do not offer any representation of this dynamic behavior and concurrent instances are usually left implicit. `Catch` is a block with two or more branches. The first branch, depicted as a solid line, represents the main flow; the other branches, each one annotated with a type ξ_i, represent the alternative execution of the block if an exception of type ξ_i occurs on the first branch.

NestFlow distinguishes between a task type and its instances. A task instance is denoted as $t : T$, where $t \in \mathcal{I}$ is an identifier chosen among the set of valid identifiers $\mathcal{I}$, and $T \in \mathcal{I}$ is its type. In the graphical representation the textual identifier can be left implicit, because a task instance is uniquely identified by its place in the model, as further explained in the following sections.

7.2 Data-Flow Aspects

A process specification shall be accompanied with the declaration of its input and output streams as well as local variables by using the keywords `in`, `out` and `var`, respectively. Input and output streams represent the interface of the process, while variables capture the main part of its internal state.

It is assumed that every stream, variable and process has a type and an identifier, for instance `x:Int` denotes a variable with identifier `x` and type `Int`. The same notation holds for streams, but for convenience a subscript `in` or `out` is added to the identifier, especially when no declaration is given. As stated above, a process

instance invoked in a `run` command is declared in a similar way with the difference that its identifier can be omitted in the graphical representation.

A *stream* is simply a queue for objects of the same type that can be used for modeling, not only the information flow, but also the set of objects needed and produced by a task. An instance of A can refer to one of its own interface stream α through the dot notation `this`.α, where `this` is a language keyword. Similarly, a stream α of an internal component instance $b{:}B$ is referred by $b.\alpha$, where b is the instance identifier and α is an interface stream of B. The dot-notation ensures that all streams in a component are uniquely identified and the keyword `this` can be left implicit.

As regards to the available stream and variable types, NestFlow has been extended with all geometric types defined in the JTS library [136], which are those defined by the OGC Simple Features Specification for SQL, and the Macs concepts presented in Chap. 3, for instance, Macs feature, hard topological relation, Macs database, etc. In the feature other complex types, such as topological structures, linear networks, and so on, will be added.

The flow of objects among internal tasks of a model is expressed through the notion of *link* that is a unidirectional connection between two streams. A link is graphically denoted using a dashed line with an hollow arrow pointing to the task that exposes the input stream. Depending on involved commands, link ends are also annotated with stream or variable identifiers as summarized in Fig. 7.2.

Links can be distinguished in internal and external ones, as in Fig. 7.2.a, Fig. 7.2.b and Fig. 7.2.d, respectively. Internal links connect two streams inside the same process, while external links are dangling dashed lines that represent an interaction with the environment.

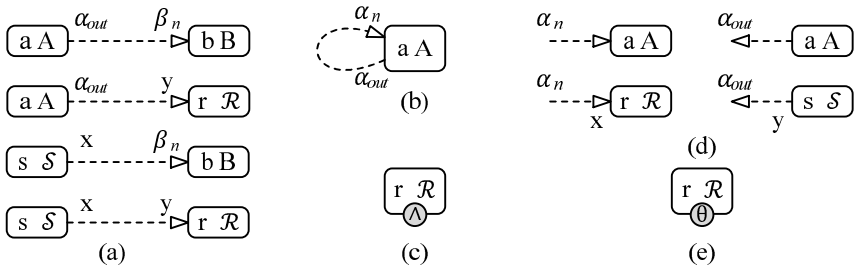


Fig. 7.2. Different combinations of NestFlow links. Tasks are denoted by A and B and the corresponding task instance identifiers by lower-case letters. Streams are denoted by the initial Greek letters α and β, while x and y are variables. $\mathcal{S}$ and $\mathcal{R}$ denote commands `Send` and `Receive`, respectively.

One strength of NestFlow resides on the possibility to hide links and their related commands any time they can be subsumed by the main control-flow. For instance, internal links with the same source and target can be grouped into a unique *collapsed link* and then subsumed by a control-flow relation with the same direction; moreover, `Receive` and `Send` commands with hidden links can be subsumed as well in a well-formed model. The remaining links mostly describe interactions among concurrent tasks. Message passing is sufficiently expressive to encode pa-

rameter passing mechanism, hence parameter passing notation can be considered as a syntactic abbreviation for representing message exchange.

A process can declare zero or more variables with their own type. Variables are visible only inside the component where they have been declared and their scope does not extend to components contained in it. For shortening the presentation, NestFlow does not include operators to express computations on variables: it can be assumed without losing generality that any computation can be implemented as a native task eventually labeled with the respective expression. For example, a task denoted with `x` ← `x` + `y`, as in Fig. 7.3, means that exists a native task with two input streams $\texttt{x}_{in}$`:Int`, $\texttt{y}_{in}$`:Int` and an output stream $\texttt{x}_{out}$`:Int` that taken the value of the variables `x` and `y` in the same scope computes their sum and stores it again in `x`. The connection between variables and streams is obtained by means of `Send`/`Receive` commands and links. Thanks to this kind of encoding, the setof constructs can be reduced without losing expressiveness.

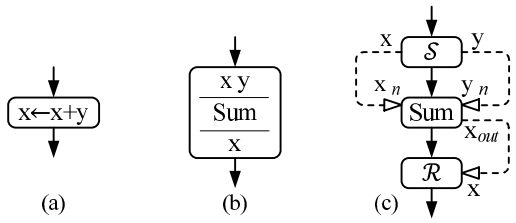


Fig. 7.3. (a) Example of an expression involving two variables x and y. Both variables are the input of a task implementing the function $f(x, y) = x = y$. The computed result is stored in the same variable x. (b) Functional-style parameter passing. (c) The two notations can be implemented the basic with message passing mechanism.

7.3 Well-Formed Models

Properties of well-formed models are construction rules that can be statically checked without executing a model. These properties are important because they can guarantee the presence of a good run-time behavior that can be hard to prove otherwise. A type system is the prime example of these properties: if types in a model are coherent, then many faulty states cannot be reached when it is executed. For instance, a link can connect only an output stream with an input stream of the same type or a super-type; a `Spawn` command can be placed only inside a `Concur` block; and so on. An exhaustive formalization of all NestFlow well-formedness properties is out of the scope of this thesis, it can be found in [47]; instead, this section wants to introduce the most relevant ones, namely:

- *Only one place per component instance* – it guarantees that there are no concurrent executions of the same component which may leave the process in an inconsistent state. At the same time it simplifies data-flow graphical specification, because the source and the target of a link are uniquely identified without explicitly stating task identifiers.

- *No shared variables among parallel branches* – it prunes away non-deterministic executions caused by the exact timing of events that are not completely under the designer's control. It also forces the use of AMP constructs when they are more suitable.

Both properties can be relaxed at the expense of making their check and the graphical representation more complex. For example, two or more parallel branches can access in read-only mode to the same variable without causing inconsistencies; read-only access can be easily verified by checking whenever a variable appears as an output or a left-value in an assignment statement.

7.4 Run-time Behavior

The complete state of a process is given by the state of its streams, its variables, and the component instances contained in it. In NestFlow a component instance is *stateful*: it retains its state over multiple executions. This characteristics is in contrast with many existing PMLs, such as YAWL. Any new instance starts from an initial state, which may be modified by sequential executions of the component internal blocks. NestFlow blocks have the following behavior:

- `Seq` – It runs the first inner block $\langle B \rangle$ until completion, then it executes the next one.
- `Choice` – It evaluates conditions $\varphi_i(\overline{x})$ associated to each branch in a fixed order. As soon as a condition is true, the corresponding branch is executed, otherwise the default (rightmost) one is chosen.
- `Loop` – It executes blocks contained in its two branches multiple times. After the execution of the right branch in Fig. 7.1, condition $\varphi(\overline{x})$ is evaluated. If the condition is false the loop exits, otherwise it executes the left and the right branch in sequence.
- `Par` – It executes the specified branches in parallel each with its own thread of control. The condition $\psi(\bar{x})$ at the end of the parallel block can be used to define a generalized partial join. This condition is evaluated whenever possible on the available variables every time a parallel branch completes. A variable $x \in \bar{x}$ is available if is not involved in a running parallel branch, otherwise it is considered unknown and so the part of ψ concerning x. When $\psi(\bar{x})$ is true, the remaining running branches are cancelled by raising an exception. In any case, a parallel block is leaved only when all threads have been completed or reverted.
- `Concur` – It is the scope of a dynamic component creation. It initially executes the main body $\langle B \rangle$ but one or more parallel branches can be added at run-time using a `Spawn`; all threads join before exiting the block.
- `Catch` – It executes the default branch and if an exception of type ξ_i is raised inside it, the current execution is interrupted and resumed from the branch annotated with the proper exception type ξ_i to handle the exceptional situation. For example, in Fig. 7.1 the non-terminal block $\langle B \rangle$ in the second branch is executed when an exception of type ξ is raised by a `Throw` command in the default branch. An exception raised inside a parallel branch that does not

contain a corresponding `Catch` block, causes also the interruption of all the other tasks that are running in the parallel branches, through the raise of an interrupt exception on those branches, in order to revert the entire block. All these different exceptions (i.e. the initial exception and the various interrupt ones) are grouped into a single exception before leaving the parallel block.

- `Skip` – It is useful for obtaining specific control-flow structures from generic ones; for instance, the usual *while* and *repeat-until* loops can be obtained replacing the right or the left branch of a `Loop` with it, respectively.
- `Run` – It executes a component instance and the current thread of control is suspended until the component completes. The command has no special symbol, it is only labeled with the invoked component type and optionally with the component instance identifier.
- `Spawn` – It creates a new task instance $t{:}T$ that is immediately executed into a new parallel branch added to the inner `Concur` block containing the command.
- `Throw` – It raises an exception of the specified type, reverting recursively all blocks that contain it until a proper handler is reached.
- `Send` – It inserts the value of one or more variables into one or more corresponding output streams. A `Send` is non-blocking: the execution continues with the next tasks without waiting.
- `Receive` – It stores into one variable an object extracted from one of the available input streams. The `Receive` temporally suspends the current thread of control until an object arrives or a timeout θ expires. A `Receive` with a timeout θ can be annotated as in Fig. 7.2(d). A multiple `Receive` stores the first arrived object from a stream α_{in}^{i} into the corresponding variable x_i, resets the others to unbounded and continues the execution; this behavior is called *or-receive*. A sequence of `Receive` commands can be grouped into a unique *and-receive* which waits an object from each connected stream before proceeding and is denoted as in Fig. 7.2.c.
- `Empty`. It removes all objects in a specified stream α.

A `Send` has only output streams and a `Receive` only input ones: regardless of its name, a stream can have a different direction depending on the internal or external perspective. Furthermore, `Send` and `Receive` commands can be viewed as special component instances with their own identifiers, hence each variable x involved in a `Send` $s{:}\mathcal{S}$ can be considered as an output stream $s.x_{out}$, while each variable y of a `Receive` $r{:}\mathcal{R}$ can be considered as an input stream $r.y_{in}$.

7.5 Design of the Integration Process

This section discusses the NestFlow design of the integration process proposed in Part I, both in its local and distributed version. This process is a good representative of the geographical processes considered in this thesis, as it presents many of the characteristics exposed in the introduction. Indeed, it is a distributed application that requires the coordination and collaboration of several SDI members with different competencies, and in which not all the operations can be automated.

From the comparison performed in the previous chapter, we can observe that processes in the geographical field can benefit from both the adoption of business

and scientific WfMSs, but none of them are completely satisfactory. The ideal WfMS for geo-processing has to combine the characteristics of both approaches in a coherent system. In particular, the most suitable solution for geo-processing seems to be the DFO approach adopted by scientific WfMSs enhanced with coarse-grained control-flow structures.

NestFlow is an example of modeling language in which control-flow constructs are coupled with data-flow abstractions. Through the models of the following sections it will be clear the importance of having data-flow constructs for representing data interactions and dependencies among objects, together with control-flow constructs that help to enforce scheduling constraints and simplify the representation.

Let us notice that in many cases the abbreviations introduced in Fig. 7.3 are used to represent message passing in a compact way without introducing additional constructs. Moreover, tasks are assumed to exchange references to a Macs database, but for brevity in some cases only the term database is used.

7.5.1 The Main Integration Process

Fig. 7.4 depicts the main steps of the integration process presented in Chap. 4, the colored tasks are compound tasks whose decompositions are discussed in the following sub-sections. The process starts when two messages are received, containing the reference to the two Macs databases to be integrated. The first `Receive` is decorated with a logic *and* $\wedge$ symbol, denoting the fact that it waits the arrival of both messages before continuing the process execution. Such `Receive` puts the references obtained from the input streams $\texttt{in}_1$ and $\texttt{in}_2$ into the variables $\texttt{macs}_1$ and $\texttt{macs}_2$, respectively.

The subsequent `Catch` block manages the situation in which an invalid integration case is detected. More specifically, task `DetermineIntegrationCase` receives in input the two databases to be integrated and determines, on the basis of their characteristics, the integration scenario, which is stored in the variable `case`. In particular, such evaluation is performed by considering the intersection between the set of thematic classes, objects, positions and relations contained in the two databases, as exemplified by Tab. 4.1 in Chap. 4. If the determined integration case is one of those defined in Sec. 4.1, then the process execution continues, otherwise an `InvalidIntegrationCaseEx` exception is thrown, which is captured by the `Catch` block and the corresponding branch is executed. Such branch contains only a `send` command that inserts the value of the `case` variable into the output stream $\texttt{out}_1$.

The first integration operation `TypeInt` regards the integration of the thematic classes contained into the two source Macs databases. In particular, it receives in input the two source sets of thematic classes and produces in output a set containing their union, which is stored into the variable `ty`. Subsequently, task `ObjectInt` is performed, it receives in input the set of source objects and produces in output a set of integrated objects that is stored in the variable `obj`. This is a composite task whose details are discussed in the following section. Let us notice that task `TypeInt` and `ObjectInt` have no data dependencies, thus they could potentially execute in parallel. However, since the first task is very simple and the

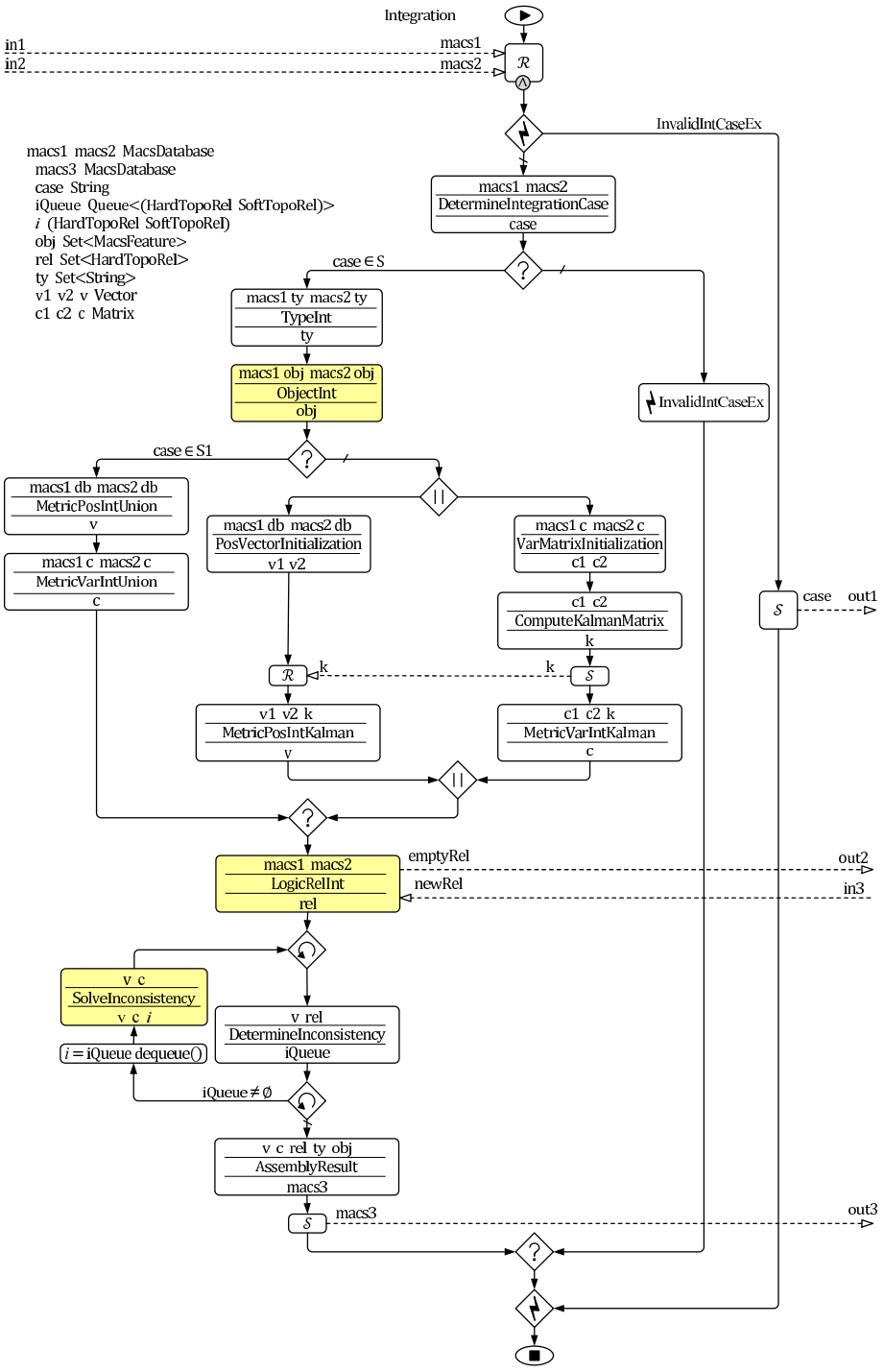


Fig. 7.4. The NestFlow representation of the main integration process. For brevity, $S = \{\text{A.0, A.1, A.2, A.3, A.4, A.5, B.1, B.2, B.3, B.4, B.5}\}$, and $S_1 = \{\text{A.0, A.1, B.4}\}$.

time required by its execution is very limited, they have been constrained to run in sequence.

After the integration of thematic classes and of objects, the positions integration and the update of accuracy information have to be performed. In order to do these, the integration case is evaluated again: if the integration case belongs to the set $S_1 = \{\text{A.0, A.1, B.4}\}$, then the simple union integration is performed, otherwise the Kalman integration is computed. In the first case, tasks `MetricPosIntUnion` and `MetricVarIntUnion` compute Eq. 4.4 and store the resulting position vector and covariance matrix into variables `v` and `c`, respectively. In the other case, before performing the Kalman integration, the vectors of positions and the matrices of covariances used during the computation have to be initialized, as illustrated in Alg. 4.4. More specifically, such initializations are performed in parallel, the two position vectors are stored into variables $\texttt{v}_1$ and $\texttt{v}_2$, while the two covariance matrices are stored into variables $\texttt{c}_1$ and $\texttt{c}_2$. After the initialization of the covariance matrices, the Kalman matrix is computed and stored into variable `k` by task `ComputeKalmanMatrix`.

The computed Kalman matrix is needed for performing both the integration of the position vectors, and the computation of the new covariances for the integrated result. Therefore, a synchronization is needed between the two parallel branches, which is represented by a message passing interaction between them: after the Kalman matrix have been stored into variable `k` in the right branch, a `send` is performed which puts its value into another variable $\texttt{k}'$ used by the left branch. Let us notice that in NestFlow parallel branches cannot share variables; therefore, two distinct variables have been used for storing the Kalman matrix. Notice that this interaction has been realized in BPMN and YAWL using a control-flow dependency which has lead to an unstructured composition, as shown in Fig. 6.5 and Fig. 6.6.

Task `MetricPosIntKalman` and `MetricVarIntKalman` perform the Kalman integration of the position vectors and of the covariance matrices; the result of these operations is stored into variables `v` and `c`, respectively. After the metric integration, task `LogicRelInt` computes the sets of topological relations that are valid in the resulting database. The details of this compound task will be discussed in Sec. 7.5.3; it can observed that the task can send and receive messages to and from the external environment if empty relations are determined at this stage.

The following loop is responsible to determine and solve existing inconsistencies between the topological relations in `rel` and those that can be derived from the integrated objects geometry in `v`. Initially, task `DetermineInconsistency` searches for existing inconsistencies and stores them into the queue `iQueue`. If this queue is not empty, the left branch of the loop is performed which takes the first inconsistency in `iQueue` and try to solve it through the task `SolveInconsistency`. The details of such task will be discussed in Sec. 7.5.4. The loop is performed again until all inconsistencies have been solved.

When all inconsistencies have been solved, task `AssemblyResult` is performed which builds the resulting Macs database from the previously computed information and stores it into the variable $\texttt{macs}_3$. The content of this variable is inserted into the output stream $\texttt{out}_3$ by the final `Send` command.

7.5.2 The ObjectInt Process

Fig. 7.5 depicts the details of the process that performs the integration of the two source object sets. The process structure is quite simple: it receives from the input

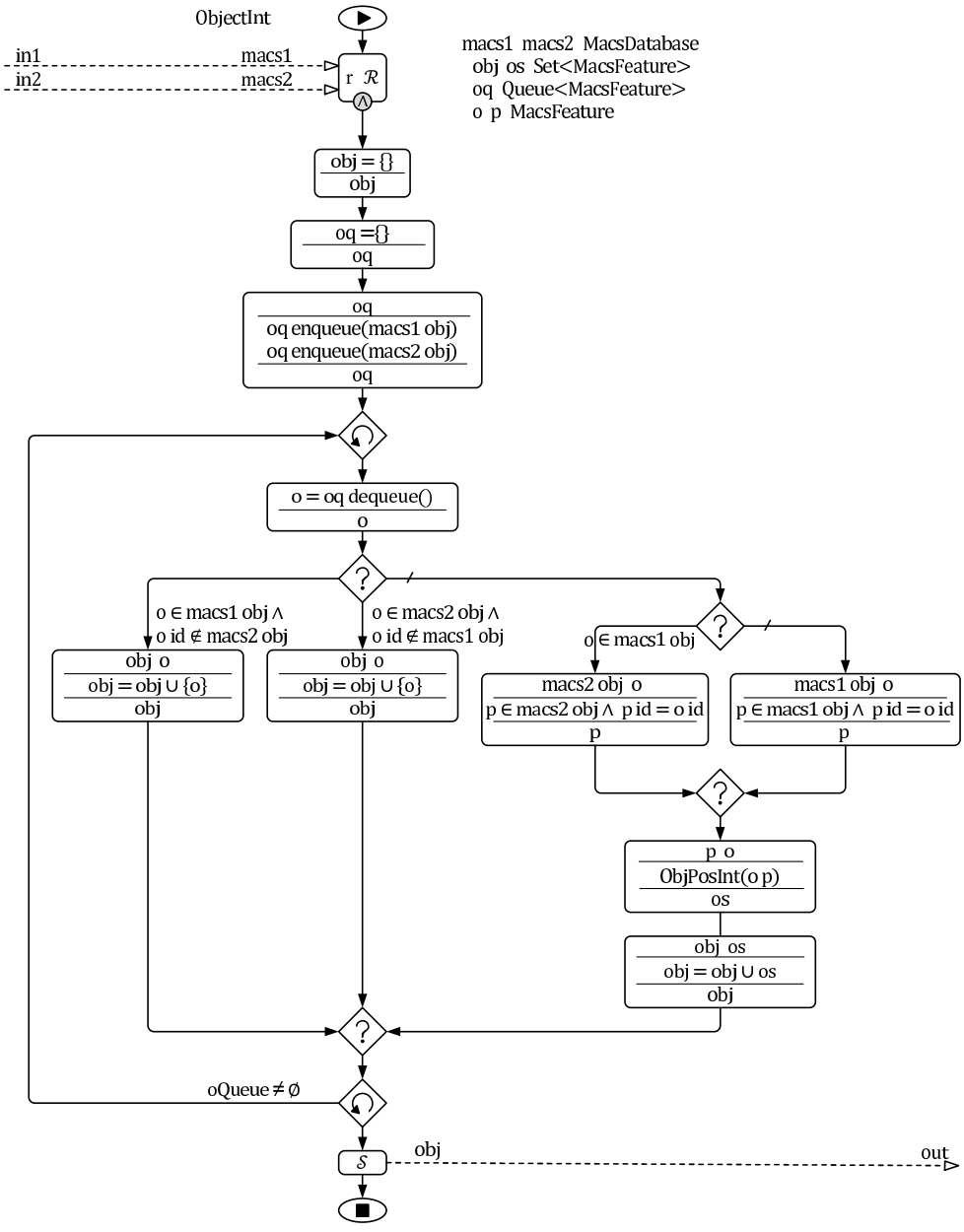


Fig. 7.5. The NestFlow representation of the ObjectInt process.

streams $\texttt{in}_1$ and $\texttt{in}_2$ the references to the two Macs databases, then two internal variables `obj` and `oq` are initialized. The first one is used to build the process result, while the second one is used to store the objects that have to be analyzed. The main `Loop` block processes the queue `oq` containing the union of the source objects: for each object `o` in `oq` a different operation is performed for adding it to the result, depending on the fact that it is contained only in one or both the source databases. More specifically, if `o` is contained in only one source database, then it is simply added to `obj`. Otherwise the corresponding object `p` contained in the other source database is retrieved and task `ObjectPosInt` is performed, in order to determine the set of positions that will be considered during the integration. The final result is inserted into the output stream `out` by the final `Send`.

7.5.3 The LogRelInt Process

Fig. 7.6 illustrates the details of the process that integrates the topological information contained into the two source Macs databases. The initial `Receive` command waits the availability of data in all three input channels $\texttt{in}_1$, $\texttt{in}_2$ and $\texttt{in}_3$. The first two input channels are used to receive the source Macs databases, while the last one is used to receive a value denoting the current integration scenario. In particular, if the value of the variable `case` is one among A.2, A.3, A.5, B.1, B.5, then the left branch is executed; otherwise, the execution continues with the right one.

In the left branch of the `Choice` block, task `TopFromSupp` is initially performed. It implements the algorithm defined in Alg. 4.4, which computes the set of topological relations that can be derived from the support of the objects in the two source databases. The computed result is stored into the variable `rel`, which is used by the following task that assemblies the result by performing the union of the original sets of topological relations and the set `rel`, and stores such result into the variable `relSet`.

Conversely, the right branch contains a `Par` block with two branches. The first branch of the `Par` block computes two sets of objects: one containing the objects of the first database that are not contained in the second one, and the other containing the objects of the second database that are not contained in the first one. The first set of objects is stored into the variable `o1`, while the second one into the variable `o2`. Using the value of such variables, task `TopFromSupp` is executed and the produced result is stored in the variable $\texttt{r}_1$. At the same time, the second branch of the `par` block computes two sets of objects: one containing the objects of $\texttt{macs}_1$ whose identifiers are contained also in $\texttt{macs}_2$, and the other containing the objects of $\texttt{macs}_2$ whose identifiers are contained also in $\texttt{macs}_1$. The first set is stored in variable $\texttt{oInt}_1$, while the second one in variable $\texttt{oInt}_2$. These variables are used by task `MergeTopoRel` which computes Eq. 4.10, namely it merges the topological relations contained into the source databases by performing the intersection of the relations defined for the same pair of objects. As highlighted in Sec. 4.3 these intersections can produce empty sets any time discordant relations are stored in the source databases. In this case task `MergeTopRel` sends a message for notifying the user that an inconsistent situation has been reached, then its execution stops until a message is received that contains information about how to solve such

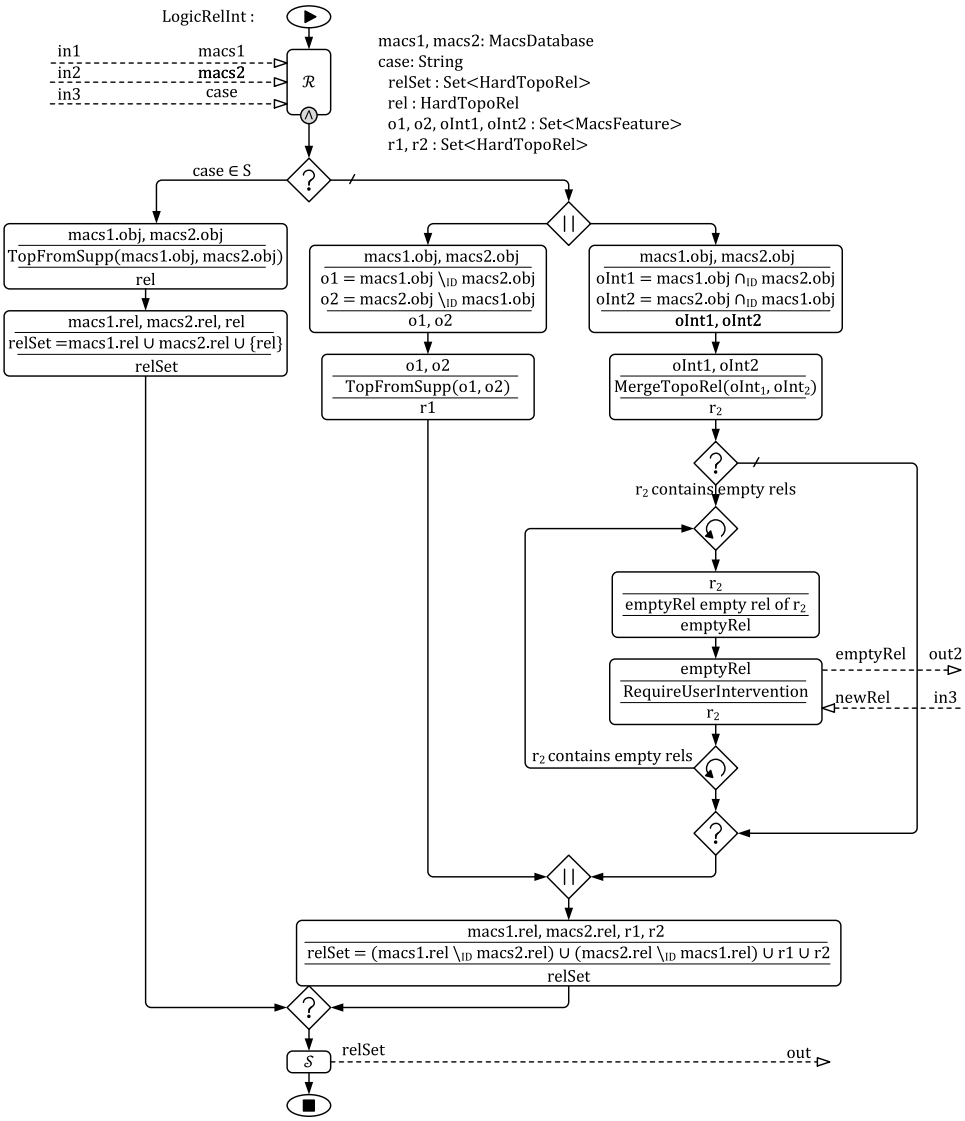


Fig. 7.6. The NestFlow representation of the LogicRelInt process. For brevity, set $S = \{A.2, A.3, A.5, B.1, B.5\}$

situation. The result computed by `MergeTopRel` is stored into the variable $\mathtt{r}_2$. This task is an example of situation in which not all inputs are necessary before starting a compound task execution, and not all outputs are produced at the end. Moreover, in some cases such input/output values can be not necessary at all. In order to represent the same behaviour with BPMN or YAWL, the tasks composing `MergeTopRel` could not be isolated into a unique task, but they shall be represented explicitly.

After the completion of the two parallel branches, the subsequent task computes the resulting set of topological relations by performing the union between the set of relations contained only in the first database, the set of relations contained only in the second database, and the two computed sets $\texttt{r}_1$ and $\texttt{r}_2$. The result of this task is stored into the variable `relSet`. Finally, the content of the variable `relSet` is inserted into the output stream `out` by the last `send` command.

7.5.4 The SolveInconsistency Process

The details of the process for solving inconsistencies between hard and soft topological relations are depicted in Fig. 7.7. This process receives from its input streams a reference to the two source databases, stored respectively into the variables $\texttt{macs}_1$ and $\texttt{macs}_2$, a reference to the database obtained from a previous metric integration, contained into variable $\texttt{macs}_3$, and an inconsistency contained into the variable i. Each inconsistency is represented as a pair formed by a hard and a soft topological relation (R, r_{soft}) such that $r_{soft} \notin R$.

The first task `IdentifyRelTransition` identifies the transition $t = r_A \rightarrow r_B$, such that r_A is the topological relation derivable from the current object geometries, while r_B is a new topological relation such that $r_B \in R$. Subsequently the applicability of t is verified and stored in the boolean variable `b`. If the identified transition is not applicable, relatively to what specified in Tab. 4.2-4.10, a `NotApplTransitionEx` exception is thrown, which is captured by the right branch of the `Catch` block that sends the transition `t` to the output stream $\texttt{out}_1$. Otherwise if the transition can be applied, the computation continues with the left branch of the first `Choice` block.

In case the desired relation r_B is valid in one of the two source databases, task `ChangeCovariances` determines the necessary modification to be applied in the source covariance matrices, in order to make the relation satisfied also in the integrated database. Then a new metric integration is performed between the modified source databases $\texttt{macs}_1$ and $\texttt{macs}_2$, by tasks `MetricPosIntKalman` and `MetricVarIntKalman` that update the corresponding components of $\texttt{macs}_3$.

Conversely, if the relation r_B is not valid in any of the two source databases, task `DetermineGeomChanges` initially determines the necessary changes to apply to the object geometries, as specified in Alg. 4.6-4.9. Then the identified changes are applied by task `ApplyGeomChanges`, obtaining a new Macs database that is stored in variable $\texttt{macs}_4$. Then a new metric integration is performed between $\texttt{macs}_3$ and the newly generated $\texttt{macs}_4$. The results of such integration are stored in the variable $\texttt{macs}_3$. Finally, the updated content of $\texttt{macs}_3$ is sent to the output stream $\texttt{out}_2$ by the last `send` command.

7.5.5 The DistributedIntegration Process

This section discusses the NestFlow representation of the distributed version of the process presented in Chap. 4. Let us notice that in this process different and geographically distributed agents are involved: one SDI manager and several SDI members. Fig. 7.8-7.9 illustrates the process performed by each SDI member, while Fig. 7.10 reports the process of the SDI manager.

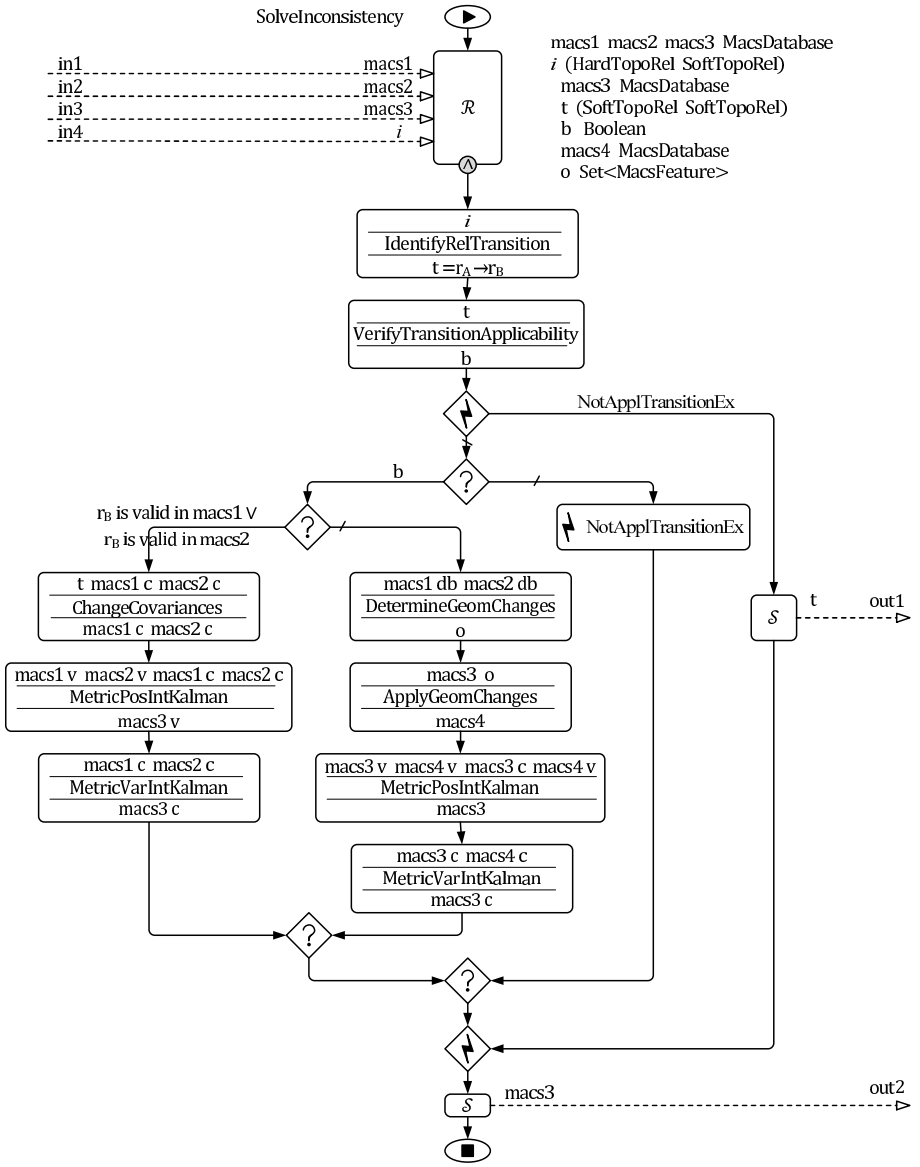


Fig. 7.7. The NestFlow representation of the SolveInconsistency process.

The first task performed by each SDI member consists in the specification of a remote URI representing the address of the SDI manager process. This address will be subsequently used for sending messages to the SDI manager and invoking its services. This operation is manually performed by a human user, as suggested by the icon decoration placed inside the task `SpecifySdiManagerUri`. This is a startup operation that has to be performed prior to all the other ones.

The `Concur` block contained in the second branch of the `Par` block denotes the fact that many local integration processes can be performed in parallel by each SDI

member. More specifically, as soon as new or updated data are received trough the input stream $\texttt{in}_1$, the `Spawn` command inside the `Loop` block starts a new instance of task `Lip` which is responsible for performing the local integration of these new data with the existing ones. Notice that the execution of a `Spawn` command simply places into a dynamically created parallel branch a new instance of the spawned task. The output of the `Spawn` command is the identifier of the created instance which is placed into the variable ℓ. The `Concur` dashed branch is a graphical representation of the dynamic creation of a parallel branch with a new instance of the spawned task. Each execution of the `Spawn` command generates at run-time a

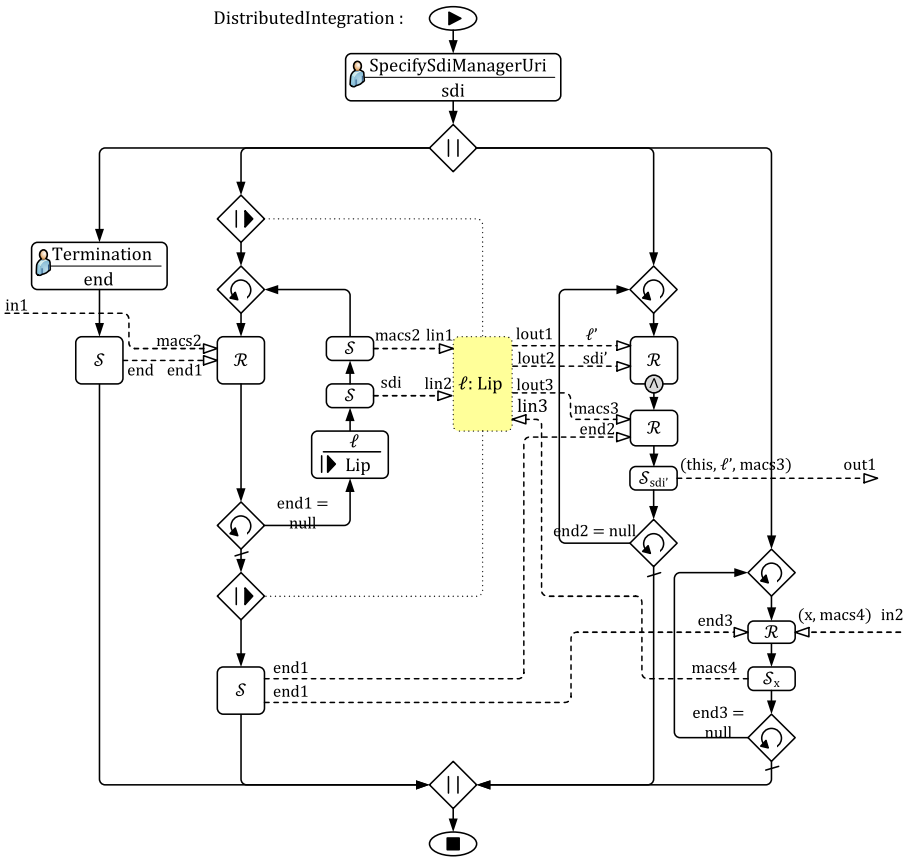


Fig. 7.8. The NestFlow representation of th DistributedIntegration process.

new parallel branch containing a new instance of `Lip`. Such task instance receives from the input stream $\texttt{lin}_1$ a reference to the new Macs database, and from the other input stream $\texttt{lin}_2$ its identifier.

The `Loop` blocks in the third and fourth branch are used for assembling the result produced by each dynamic instance without breaking the encapsulation. More specifically, the output produced by each dynamic `Lip` instance is sent to the output stream $\texttt{out}_1$ which belongs to the global process, and similarly the input of each

dynamic `Lip` instance is received through the input stream $\texttt{in}_2$ of the global process. In other words, the external environment does not known the existence of such dynamic instances and communicates only with task `DistributedIntegration`.

The `Loop` block in the third branch is responsible for sending the result of the local integration to the SDI manager. In particular, the first two `Receive` commands collect the instance identifier, stored into variable ℓ', the address of the SDI manager, stored into variable $\texttt{sdi}'$, and the result of the local integration, stored in `macs3`. The subsequent `send` command sends to $\texttt{sdi}'$ the tuple `(this,` ℓ', $\texttt{macs}_3$`)` through the output stream $\texttt{out}_1$, where `this` is a language keyword representing the identifier of the current process. The SDI manager will use the reference in `this` for communicating with the current process, while the reference in ℓ' will be returned together with the result, in order to correctly redirect the received data.

The `Loop` in the fourth branch does a similar work for the input. The process receives from the input stream $\texttt{in}_2$ the result of the global integration performed by the SDI manager. More specifically, it receives a tuple `(x,` $\texttt{macs}_4$`)` containing a reference `x` to the specific dynamic instance of `Lip` and the integrated Macs database $\texttt{macs}_4$. The returned reference `x` is one of those previously sent to the SDI manager by the `send` command in the third branch.

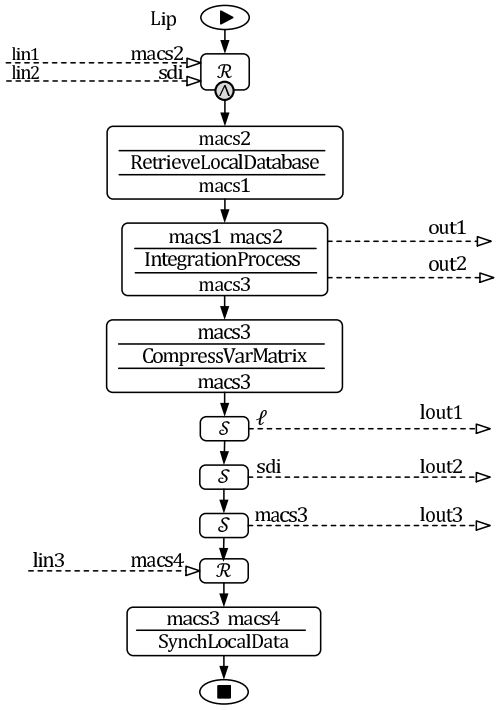


Fig. 7.9. NestFlow representation of the Lip process.

The details of the `Lip` process are illustrated in Fig. 7.9. The process starts by receiving the Macs database to be integrated and the address of the SDI man-

ager. The database $\texttt{macs}_2$ is used by task `RetrieveLocalDatabase` to retrieve the information of the local database that has to be integrated, such information are stored into the variable `macs1`. An instance of task `IntegrationProcess` (depicted in Fig. 7.4) is then executed. Notice that, while its output streams $\texttt{out}_1$ and $\texttt{out}_2$ becomes output streams of the `Lip` process, the output streams $\texttt{out}_3$ is connected to the subsequent task `CompressVarMatrix`. This task compress the covariance matrix as explained in Sec. 4.7 and updates the Macs database referenced by $\texttt{macs}_3$. The reference ℓ' to the current instance, the address `sdi` of the SDI manager and the result of the local integration are sent to the outer process by the following three `send` commands. Finally, through the input stream $\texttt{lin}_3$ it receives from the outer process, the result of the global integration performed by the SDI manager. This result is used to synchronize the local data by the last task `SynchLocalData`.

Fig. 7.10 illustrates the global integration performed by the SDI manager. It initially receives a tuple `(p`, ℓ, `macs)` containing the reference of the invoking process `p`, the reference to the particular instance ℓ and the Macs database to be integrated `macs`. Task `Gid` performs the global integration presented in Sec. 4.6, then the result is sent to process `p` together with the reference to the instance ℓ, and to all the other involved SDI members whose address has been identified and stored into the list `mlist`.

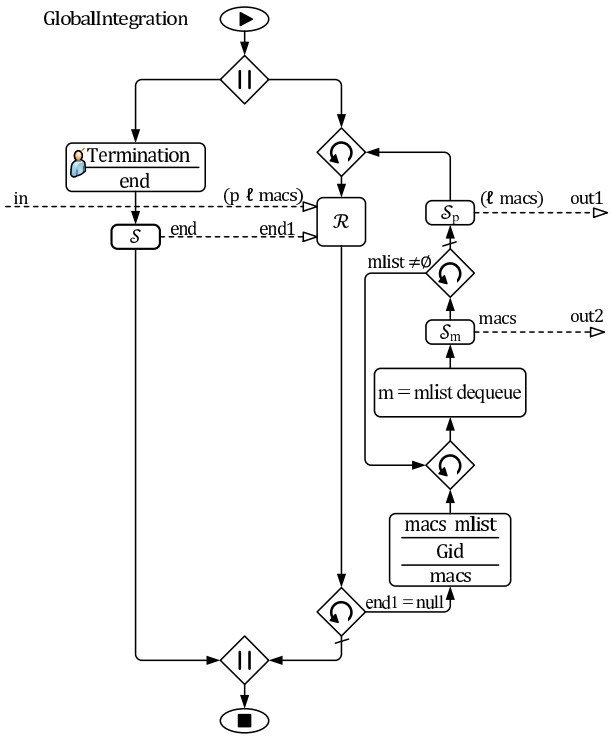


Fig. 7.10. NestFlow representation of the GlobalIntegration process performed by the SDI manager.

7.6 Summary and Concluding Remarks

In the previous chapter we have highlighted the need in the geographical domain for a WfMS that combines the strengths of both business and scientific WfMSs in a coherent way. In particular, it should inherit from scientific WfMSs a data-oriented computational model, in which data aspects are considered a primary aspect to be modeled, but this model has to be enhanced with some high-level control-flow constructs in order to allow more sophisticated constructions.

NestFlow is conceived to explore a particular modeling language design solution in which structured control-flow constructs are tightly-coupled with asynchronous message passing connections. This language has been developed by our University in the context of another PhD thesis, and has been extended and applied here for designing the proposed integration framework. The NestFlow design of such process is presented at the end of the chapter, where it becomes clear how the explicit representation of data-flow dependencies together with the use of advanced control-flow structures can ease the design and development of geo-processes.

In this thesis only the core constructs of NestFlow have been used. A future work of this part regards the further extension and specialization of the NestFlow language for the geographical domain. In particular, the development of a library of standard components implementing geographical operations, such as the evaluation of topological relations, as well as the definition of components for the visualization of the process execution from a geographical point of view. Relatively to the management of users and roles, the system can be extended by consider the Geo-RBAC model proposed in [32], which allows one to defined privileges for a task considering also the area in which that task has to be performed.

8

Conclusions

This thesis deals with two important problems that usually affect an SDI: the integration of spatial data coming from different sources and characterized by different quality levels, and the development of a framework for supporting the construction and maintenance of a distributed and integrated global SDI database.

Relatively to the first contribution, this thesis can be considered as a starting point to deeply study and analyze the role of accuracy in any operation regarding spatial data. In particular, the thesis starts by introducing a model for representing accuracy information of spatial data and by proposing an integration framework based on this model. The model considers a statistical treatment of measurements and the integration framework is an application of the Kalman filter in a static context, where estimates do not change due to the time passage, but only for the presence of new and updated observations. The integration is also driven by the knowledge of the existing topological relations, which are treated with a logical approach since they are not subject to measurement errors. In other words, in some cases the integrated measurements can be adjusted in order to satisfy a particular topological relation.

The proposed integration procedure has been applied to solve some typical situations encountered during the construction of a regional SDI in Lombardy (Italy), while its massive application to two real-world databases require to face several additional questions. First of all, the computational complexity of the approach is mainly determined by the cost of inverting the involved variance-covariance matrices during the application of the Kalman filter. If the database contains n positions, then the dimension of the variance-covariance matrix is approximately n^2, and this dimension can be very high for real databases. For this reason, we propose some techniques for reducing the number of covariance information that have to be stored and eventually transferred in case of a distributed environment. However, the matrix keeps its full dimension during the computation; hence, the first future extension can be the development of other approaches to reduce the cost of the inversion operation.

Another critical aspect for the massive application of the proposed integration procedure is the first assumption about the preliminary database alignment. Indeed, the approach assumes that the two databases share the same schema, and corresponding objects are characterized by the same identifier. This matching op-

eration is far from being simple and is still an open research problem. In some cases the source databases can be characterized by different scales, or level of details, several objects can be aggregated into a unique one, or vice-versa. Therefore, a future work regards the study of the preliminary matching phase that allows one to label corresponding objects with the same identifier.

The last integration phase aligns metric and logical observations together, if some inconsistencies exist between the desired topological relations and the integrated positions, the last ones can be slightly modified in order to satisfy the first ones. In case the desired topological relation is satisfied by the geometry of one source database, then the relative distances between the involved objects in this database are made more accurate; moreover, we known that a configuration exists that satisfy the desired properties. Conversely, if the desired topological relation is not satisfied by any source database, then the geometry has to be properly modified ad hoc, and in such case we cannot exclude that new inconsistencies can be generated elsewhere. However, we can observe that thanks to the role taken by the accuracy of relative distances, the topological relations that are valid in the source database are substantially preserved during the metric integration. Therefore, the number of generated inconsistencies, that have to be treated with the last integration phase, is very limited.

Another important extension regards the use of the accuracy information contained into a MACS database in the query and visualization of spatial objects. For instance, let us consider the distance queries presented in Sec. 2.2.3, in this case a certain level of required accuracy can be specified for the result. Similarly, queries involving topological relations can require a certain level of certainty. As regards to the visualization aspects, the possibility to give a graphical overview of object uncertainty, in terms of its confidence region, can become a powerful tool in the geographical field, where user interactions are generally performed through maps, and not directly on numeric coordinates. In a first stage, the approximation introduced in Def. 3.12 can be applied, which estimates the confidence region of an object as the smallest buffer region containing the confidence region of all its defining positions.

The second part of the thesis deals with the realization of the proposed integration framework. This framework is a good example of geographical process with the characteristics highlighted in the introduction; therefore we decide to evaluate the applicability of existing workflow technologies for its realization. Many WfMSs are available, anyway we decide to concentrate on two representatives YAWL and Kepler which well exemplified two different approaches to process design. In particular, YAWL is a representative of the control-flow oriented approach, in which processes are described by defining the execution order to the constituent tasks. This system gives also much attention to the coordination aspects and to the management of human resources. Conversely, Kepler is a representative of the data-flow oriented approach, in which processes are described by explicitly representing the existing data dependencies among tasks. This system is also oriented to the automation of repetitive processes, rather than to the coordination aspects.

YAWL and Kepler are compared from two perspective: on the one hand from a general point of view by considering the widely accepted method of workflow patterns, and on the other hand from a spatial point of view by considering the

design of the integration process as use case. The obtained results highlight that both systems provides interesting and promising functionalities for the realization of geographical process, but none of them is completely satisfactory. Several deficiencies have to be filled in both kind of systems, and the most important one regards the adopted computational model. The final conclusion is that the data-flow computational model provided by Kepler is more satisfactory, because it is more adequate in case of long-running and data-intensive computations, but in some cases the presence of high-level control-flow structure can be desirable. For this reason, the last chapter introduces a novel modeling language called NestFlow in which structured control-flow constructs are tightly-coupled with Asynchronous Message Passing (AMP) connections. Such language has be extended in this thesis with some data types for the geometry representation, and applied for the realization of the proposed integration process. In particular, each integration step has been implemented in Java and assembled using NestFlow. Clearly many other extensions of this language are necessary for obtaining a complete geographical WfMS, for instance as regards to the management of human resources, but the initial results are satisfactory.

For future work we plan to extend the system with a library of graphical components for geo-processing, taking them from existing API like JTS [136], in order to allow a domain expert (which is not an IT expert) to assemble its necessary processes. Another important extension regards the support for an automatic composition of geographical services on the basis of their semantic description. In other words, the designer can be driven in the workflow construction by a description of the activity performed by the task and its interface. In this way only tasks with a compatible interfaces can be concatenated and tasks performing similar activities can be easily identified. Some research work about the semantic description of geographical services are present in [75–77], while Kepler provides an example of ontology-based classification of the available functionalities for ease their search and retrieval.

References

1. GlassFish. http://glassfish.org/ Accessed March 2012.
2. INSPIRE Directive. http://inspire.jrc.ec.europa.eu Accessed March 2012.
3. JBoss Application Server. http://www.jboss.org/jbossas/ Accessed March 2012.
4. The Open Geospatial Consortium, Inc. (OGC). http://www.opengeospatial.org/ Accessed March 2012.
5. Workflow Patterns Initiative. http://workflowpatterns.com. Accessed March 2011.
6. *Geography Markup Language (GML) Implementation Specification, version 2.1.2*, 2002. https://portal.opengeospatial.org/files/?artifact_id=11339 Accessed March 2012.
7. Geospatial Grid. In S. Rana and J. Sharma, editors, *Frontiers of Geographic Information Technology*, pages 121–137. 2006.
8. *OpenGIS Implementation Specification for Geographic Information - Simple Feature Access - Part 1: Common Architecture, version 1.2.1*, 2011. http://www.opengeospatial.org/standards/sfa Accessed March 2012.
9. G. Alonso and C. Hagen. Geo-Opera: Workflow Concepts for Spatial Processes. In *Proceedings of the 5th International Symposium on Advances in Spatial Databases (SSD'97)*, pages 238–258, 1997.
10. R. B. Altman. A Probabilistic Algorithm for Calculating Structure: Borrowing from Simulated Annealing. In *Proceedings of 9th International Conference on Uncertainty in Artificial Intelligence*, 1993.
11. C. K. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-Based Information Mediation with MIX. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 597–599, 1999.
12. A. Belussi, B. Catania, and P. Podestà. Towards Topological Consistency and Similarity of Multiresolution Geographical Maps. In *Proceedings of 13thACM Int. Workshop on Geographic Information Systems*, pages 220–229, 2005.
13. A. Belussi and Migliorini. Integrating Multi-Accuracy Spatial Data. Technical Report 85/2009, Department of Computer Science, University of Verona, 2011.
14. A. Belussi and S. Migliorini. A Framework for Integrating Multi-Accuracy Spatial Data in Geographical Applications. *GeoInformatica*, 16(3):523–561, 2012.
15. A. Belussi and S. Migliorini. Distributed Integration of Spatial Data with Different Positional Accuracies. In *Proceedings of the 13th AGILE International Conference on Geographic Information Science*, 2012.
16. B. D. Best, P. N. Halpin, E. Fujioka, A. J. Read, S. S. Qian, L. J. Hazen, and R. S. Schick. Geospatial Web Services within a Scientific Workflow: Predicting Marine

Mammal Habitats in a Dynamic Environment. *Ecological Informatics*, 2:210–223, 2007.
17. B. Bhanu, R. Li, C. Ravishankar, M. Kurth, and J. Ni. Indexing Structure for Handling Uncertain Spatial Data. In *Proceedings of 6th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, 2004.
18. L. Bic. A process-oriented model for efficient execution of dataflow programs. In *Data flow computing*, pages 332–347. 1992.
19. O. Boucelma, J. Garinet, and Z. Lacroix. The virGIS WFS-based spatial mediation system. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03)*, pages 370–374, 2003.
20. S. Bowers, B. Ludäscher, S. H. H. Ngu, and T. Critchlow. Enabling Scientific Workflow Reuse through Structured Composition of Dataflow and Control-Flow. In *Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDE 2006)*, page 70, 2006.
21. J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems. *Int. Journal of Computer Simulation*, 4:155–182, 1994.
22. T.B. Buyong, A.U. Frank, and W. Kuhn. A Conceptual Model of Measurement-Based Multipurpose Cadastral Systems. *Journal of the Urban and Regional Information Systems Ass. URISA*, 2(3):35–49, 1991.
23. Q. Chen, L. Wang, and Z. Shang. MRGIS: A MapReduce-Enabled High Performance Workflow System for GIS. In *Proceedings of the 4th IEEE International Conference on eScience (ESCIENCE'08)*, pages 646–651, 2008.
24. L. Chun and T. Xiaohua. Relationship of uncertainty between polygon segment and line segment for spatial data in GIS. *Geo-Spatial Information Science*, 8(3):183–188, 2005.
25. E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proceedings of 3rd International Symposium on Advances in Spatial Databases (SSD'93)*, pages 277–295, 1993.
26. E Clementini and P Di Di Felice. *An Algebraic Model for Spatial Objects with Indeterminate Boundaries*, pages 155–169. London: Taylor & Francis, 1996.
27. M. A. Cobb, M. J. Chung, H. Foley, F. E. Petry, K. B. Shaw, and H. V. Miller. A Rule-based Approach for the Conflation of Attributed Vector Data. *GeoInformatica*, 2(1):7–35, 1998.
28. A. G. Cohn and N.M. Gotts. The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries. In P. Burrough and A. Frank, editors, *Geographical Objects with Undetermined Boundaries*, pages 171–187. Taylor and Francis, 1996.
29. C. Combi, M. Gambini, and S. Migliorini. The NestFlow Interpretation of Workflow Control-Flow Patterns. In *Proceedings of 15th International Conference on Advances in Databases and Information Systems (ADBIS'11)*, pages 316–332, 2011.
30. C. Combi, M. Gambini, S. Migliorini, and R. Posenato. Modelling Temporal, Data-Centric Medical Processes. In *Proceedings of 2nd ACM SIGHIT International Health Informatics Symposium, IHI 2012*, pages 141–150, 2012.
31. Carlo Combi and Mauro Gambini. Flaws in the Flow: The Weakness of Unstructured Business Process Modeling Languages Dealing with Data. In *On the Move to Meaningful Internet Systems (OTM 2009) Confederated International Conferences*, pages 42–59, 2009.
32. M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. *ACM Trans. Inf. Syst. Secur.*, 10(1), 2007.
33. B. V. Dasarathy. Information Fusion - What, Where, Why, When, and How? *Information Fusion*, 2(2):75–76, 2001.

34. T. Devogele, C. Parent, and S. Spaccapietra. On Spatial Database Integration. *International Journal of Geographical Information Science*, 12(4):335–352, 1998.
35. L. Di, A. Chen, W. Yang, Y. Liu, Y. Wei, P. Mehrotra, C. Hu, and D. Williams. The Development of a Geospatial Data Grid by Integrating OGC Web Services with Globus-based Grid technology. *Concurrency and Computation: Practice and Experience*, 20(14):1617–1635, 2008.
36. M. Duckham, J. Lingham, K. T. Mason, and M. F. Worboys. Qualitative reasoning about consistency in geographic information. *Information Science*, 176(6):601–627, 2006.
37. M. Duckham and M. F. Worboys. An algebraic approach to automated geospatial information fusion. *International Journal of Geographical Information Science*, 19(5):537–557, 2005.
38. M. J. Egenhofer and R. Franzosa. Point-set Topological Spatial Relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.
39. M. J. Egenhofer and R. D. Franzosa. On the Equivalence of Topological Relations. *International Journal of Geographical Information Systems*, 9(2):133–152, 1995.
40. M. J. Egenhofer and D. M. Mark. Modelling Conceptual Neighbourhoods of Topological Line-Region Relations. *International Journal of Geographical Information Systems*, 9(5):555–565, 1995.
41. Max J. Egenhofer. What's special about spatial?: database requirements for vehicle navigation in geographic space. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 398–402, 1993.
42. ESRI. *ArcGIS Workflow Manager A Workflow Management Solution*, 2007. http://www.esri.com/library/whitepapers/pdfs/jtx-workflow-mgmt.pdf Accessed March 2012.
43. T. Foerster and B. Schäffer. A Client for Distributed Geo-processing on the Web. In *Proceedings of Web and Wireless Geographical Information Systems*, pages 252–263, 2007.
44. F. T. Fonseca, C. A. Davis, and G. Câmara. Bridging Ontologies and Conceptual Schemas in Geographic Information Integration. *GeoInformatica*, 7(4):355–378, 2003.
45. F. T. Fonseca and M. J. Egenhofer. Ontology-Driven Geographic Information Systems. In *Proceedings of the 7th International Symposium on Advances in Geographic Information Systems (ACM-GIS'99)*, pages 14–19, 1999.
46. F. T. Fonseca, M. J. Egenhofer, P. Agouris, and G. Câmara. Using Ontologies for Integrated Geographic Information Systems. *Transactions in GIS*, 6(3):231–257, 2002.
47. M. Gambini and S. Migliorini. NestFlow Process Modeling Language Specification. Technical report, 2010. http://www.nestflow.org/downloads/nestflow_spec.pdf.
48. M. Gambini and S. Migliorini. NestFlow Workflow Control-Flow Patterns Support. Technical report, 2010. http://www.nestflow.org/downloads/nestflow_wcps.pdf.
49. F. Gielsdorf, L. Gruending, and B. Aschoof. Positional Accuracy Improvement - A Necessary Tool for Updating and Integration of GIS Data. In *Proceedings of the FIG Working Week 2004*, pages 1–13, 2004.
50. M.F. Goodchild. Measurement-based GIS. In W. Shi, P.F. Fisher, and M.F. Goodchild, editors, *Spatial Data Quality*, pages 5–17. Taylor and Francis, 2002.
51. R. K. Goyal. *Similarity Assessment for Cardinal Directions between Extended Spatial Objects*. PhD thesis, 2000.

52. R. K. Goyal and M. J. Egenhofer. The Direction-Relation Matrix: A Representation of Direction Relations for Extended Spatial Objects. In *Proceedings of UCGIS Annual Assembly and Summer Retreat*, 1997.
53. R. K. Goyal and M. J. Egenhofer. Consistent Queries over Cardinal Directions Across Different Levels of Detail. In *Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA'00)*, pages 876–, 2000.
54. R.V.L. Hartley. Transmission of information. *Bell System Techical Journal*, 7:535–563, 1928.
55. H.R. Hashemipour, S. Roy, and A.J. Laub. Decentralized Structures for Parallel Kalman Filtering. *IEEE Transactions on Automatic Control*, 33(1):88–94, 1988.
56. K. A. Hawick, P. D. Coddington, and H. A. James. Distributed Frameworks and Parallel Algorithms for Processing Large-Scale Geographic Data. *Parallel Computing*, 29(10):1297–1333, 2003.
57. R. G. Healey, M. J. Minetar, and S. Dowers. *Parallel Processing Algorithms for GIS*. Taylor & Francis, Inc., 1997.
58. S. Hope. *Integration of Vector Datasets*. PhD thesis, Department of Geomatics, University of Melbourne, 2008.
59. S. Hope and A. Kealy. Using Topological Relationships to Inform a Data Integration Process. *Transactions in GIS*, 12(2):267–283, 2008.
60. S. Hope, A. Kealy, and G. Hunter. Improving Positional Accuracy and Preserving Topology through Spatial Data Fusion. In *Proceedings of 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, pages 675–684, 2006.
61. Z. Huang, Y. Fang, B. Chen, and X. Wu. Development of a Grid GIS Prototype for Geospatial Data Integration. In *Proceedings of the 7th International Conference on Grid and Cooperative Computing (GCC'08)*, pages 628–631, 2008.
62. Z. C. Huang, G. Q. Li, and Y. Zeng. Building Data Infrastructure for Geocomputation by Spatial Information Grid. In *Proceedings of the 6th International Conference on Grid and Cooperative Computing (GCC'07)*, pages 630–635, 2007.
63. Humboldt Project. *The HUMBOLDT Workfow Editor*, 2010. http://community.esdi-humboldt.eu/attachments/114/Workflow_Editor_User_Manual.pdf Accessed March 2012.
64. International Organization for Standardization. *ISO 19113:2002. Geographic Information – Quality Principles*, 2002. http://www.iso.org/iso/catalogue_detail.htm?csnumber=26018 Accessed March 2012.
65. International Organization for Standardization. *ISO 19109:2005 Geographic information – Rules for application schema*, ISO/TC 211 edition, 2005. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39891 Accessed March 2012.
66. International Organization for Standardization. *ISO 3534-1:2006. Statistics – Vocabulary and symbols – Part 1: Probability and general statistical terms.*, 2006. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=40145 Accessed March 2012.
67. International Organization for Standardization. *ISO/FDIS 19152 Geographic information – Land Administration Domain Model (LADM)*, ISO/TC 211 edition, January 2011. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51206 Accessed March 2012.
68. W. M. Johnston, J. R. P. Hanna, and R. J. Millar. Advances in dataflow programming languages. *ACM Comput. Surv.*, 36(1):1–34, 2004.
69. R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal Of Basic Engineering*, 82:35–45, 1960.

70. M. Kavouras and M. Kokla. A Method for the Formalization and Integration of Geographical Categorizations. *International Journal of Geographical Information Science*, 16(5):439–453, 2002.
71. M. Kavouras, M. Kokla, and E. Tomai. Comparing categories among geographic ontologies. *Computers & Geosciences*, 31:145–154, 2005.
72. G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty, and Information*. Prentice Hall, 1988.
73. M. Kokla and M. Kavouras. Fusion of top-level and geographical domain ontologies based on context formation and complementarity. *International Journal of Geographical Information Science*, 15(7):679–687, 2001.
74. E. A. Lee and T. M. Parks. Dataflow Process Networks. *Proceedings of the IEEE*, 83(5):773–801, 1995.
75. R Lemmens. *Semantic Interoperability of Distributed Geo-Services*. PhD thesis, Delft University of Technology, 2006.
76. R. Lemmens, R. De By, M. Gould, A. Wytzisk, C. Granell, and P. van Oosterom. Enhancing Geo-Service Chaining through Deep Service Descriptions. *Transactions in GIS*, 11(6):849–871, 2007.
77. R. Lemmens, A. Wytzisk, R. De By, C. Granell, M. Gould, and van Oosterom. P. Integrating Semantic and Syntactic Descriptions to Chain Geographic Services. *IEEE Internet Computing*, 10:42–52, 2006.
78. R. Li, B. Bhanu, C. Ravishankar, M. Kurth, and J. Ni. Uncertain Spatial Data Handling: Modeling, Indexing and Query. *Computers and Geosciences*, 33(1):42–61, 2007.
79. X. Liu, X. Huang, and Z. Zhao. Research on GIS Spatial Database Based on Grid Computing. In *Proceedings of the 2009 WRI International Conference on Communications and Mobile Computing (CMC'09)*, pages 156–159, 2009.
80. P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind. *Geographic Information Systems and Science*. John Wiley & Sons, 2005.
81. B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computatition: Practice and Experience*, 18(10):1039–1065, 2006.
82. B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers. Scientific Workflows: Business as Usual? In *Proceedings of the 7th International Conference on Business Process Management (BPM'09)*, pages 31–47, 2009.
83. S. Majithia, M. Shields, I. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. pages 514–521, 2004.
84. S. Manoah, O. Boucelma, and Y. Lassoued. Schema Matching in GIS. In *Artificial Intelligence: Methodology, Systems, and Applications*, volume 3192, pages 500–509. 2004.
85. C. B. Medeiros, G. Vossen, and M. Weske. GEO-WASA - Combining GIS Technology with Workflow Management. In *Proceedings of the 7th Israeli Conference on Computer-Based Systems and Software Engineering*, page 129, 1996.
86. W. S. Meyer, J. Moreira De Souza, and M. R. Ramirez. Secondo-grid: An Infrastructure to Study Spatial Databases in Computational Grids. In *In Computational Grids 7th Brazilian Symposium on Geoinformatics*, 2005.
87. S. Migliorini, M. Gambini, A. Belussi, M. Negri, and G. Pelagatti. Workflow Technology for Geo-Processing: the Missing Link. In *Proceedings of the 2nd International Conference and Exhibition on Computing for Geospatial Research & Application, COM.Geo 2011*, page 36, 2011.
88. S. Migliorini, M. Gambini, M. La Rosa, and A.H.M. ter Hofstede. Pattern-Based Evaluation of Scientific Workflow Management Systems. Technical Report 39935, Queensland University of Technology, 2011.

89. M. Molenaar and P.J.M. van Oosterom. Conceptual Tools for Specifying Spatial Object Representations. In M. Madden, editor, *Manual of Geographic Information Systems*, pages 47–69. American Society for Photogrammetry and Remote (ASPRS), 2009.
90. Martien Molenaar. *An Introduction to the Theory of Spatial Object Modelling for GIS (Research Monographs in GIS)*. Taylor & Francis Ltd, 1998.
91. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4), 1989.
92. G. Navratil, M. Franz, and E. Pontikakis. Measurement-Based GIS Revisited. In *Proceedings of 7th AGILE Conference on Geographic Information Science*, pages 771–775, 2004.
93. J. Ni, C. V.. Ravishankar, and B. Bhanu. Probabilistic Spatial Database Operations. In *Advances in Spatial and Temporal Databases*, pages 140–158, 2003.
94. OASIS. *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, 2007. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf Accessed March 2012.
95. N.J. Obermeyer and J.K. Pinto. *Managing Geographic Information Systems*. The Guilford Press, 1994.
96. Object Management Group. *Unified Modeling Language 2.1.2 Super-Structure Specification*, 2007. http://www.omg.org/docs/formal/07-11-04.pdf Accessed March 2012.
97. Object Management Group (OMG). *Business Process Modeling Notation (BPMN) 2.0 (Beta 1)*, 2009. http://www.omg.org/spec/BPMN/2.0/ Accessed March 2012.
98. T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li. Taverna: a Tool for the Composition and Enactment of Bioinformatics Workflows. *Bioinformatics*, 20:3045–3054, 2004.
99. Open Geospatial Consortium Inc. *OpenGIS Web Processing Service*, 2007. http://portal.opengeospatial.org/files/?artifact_id=24151 Accessed March 2012.
100. Open Geospatial Consortium Inc. *OpenGIS Web Feature Service 2.0 Interface Standard*, 2010. http://portal.opengeospatial.org/files/?artifact_id=39967 Accessed March 2012.
101. C. Ouyang, M. Dumas, S. Breutel, and A. H. M. ter Hofstede. Translating Standard Process Models to BPEL. In *Proceedings of 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, pages 417–432, 2006.
102. J. Perkal. On the Length of Empirical Curves: Discussion Paper 10. Michigan Inter-University Community of Mathematical Geographers, Ann Arbor.
103. M. E. Pierce. Special Issue Editorial Introduction: Grids and Geospatial Information Systems. *Concurr. Comput.: Pract. Exper.*, 20(14):1611–1615, 2008.
104. M. Radwan, L. Alvarez, R. Onchaga, and J. Morales. The Changing Role of the Geo-Data Infrastructure: from a Data Delivery Network to a Virtual Enterprise Supporting Complex Services. In *Proceedings of ISPRS 2004 Photogrammetry and Remote Sensing*, pages 194–199, 2004.
105. A. Rajabifard, M. F. Feeney, and I. P. Williamson. Future directions for SDI development. *International Journal of App. Earth Observation and Geoinformation*, 4:11–22, 2002.
106. M. Reichert and B. Weber. *Enhancing Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Springer-Verlag, 2012.
107. P. Rigaux, M. Scholl, and Voisard A. *Spatial Databases with Application to GIS*. Morgan Kaufmann Publishers Inc., 2002.
108. N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. In *Proceedings of the 24th International Conference on Conceptual Modeling (ER 2005)*, pages 353–368, 2005.

109. N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. Technical Report BPM-06-22, BPM Center Report, 2006. http://www.bpmcenter.org. Accessed March 2011.
110. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P van der Aalst. newYAWL: Achieving Comprehensive Patterns Support in Workflow for the Control-flow, Data and Resource Perspectives. Technical Report BPM-07-05, BPM Center, 2007. http://www.bpmcenter.org Accessed March 2012.
111. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In *Proceedings of the 17th International Conference on Advanced Information Systems Engineering (CAiSE'05)*, 2005.
112. A. Saalfeld. Conflation: Automated Map Compilation. *International Journal of Geographical Information Systems*, 2(3):217–228, 1988.
113. B. Schäffer and T. Foerster. A Client for Distributed Geo-processing and Workflow Design. *Journal of Location Based Services*, 2(3):194–210, 2008.
114. Markus Schneider. Uncertainty Management for Spatial Data in Databases: Fuzzy Spatial Data Types. In *Proceedings of 6th International Symposium on Advances in Spatial Databases*, pages 330–351, 1999.
115. C.E. Shannon. A matematical theory of communication. *Bell System Techical Journal*, 27:379–423, 623–656, 1948.
116. W. Shi. A Generic Statistical Approach for Modelling Error of Geometric Features in GIS. *International Journal of Geographical Information Science*, 12(2):131–143, 1998.
117. W. Shi. *Principles of modeling uncertainties in spatial data and spatial analyses.* CRC Press, 2010.
118. Wenzhong Shi. A Generic Statistical Approach for Modelling Error of Geometric Features in GIS. *International Journal of Geographical Information Science*, 12(2):131–143, 1998.
119. Y. Shu, J. F. Zhang, and X. Zhou. A Grid-Enabled Architecture for Geospatial Data Sharing. In *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*, 2006.
120. S. Skiadopoulos and M. Koubarakis. Composing Cardinal Direction Relations. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases (SSTD'01)*, pages 299–320, 2001.
121. S. Skiadopoulos and M. Koubarakis. Consistency Checking for Qualitative Spatial Reasoning with Cardinal Directions. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming (CP'02)*, pages 341–355, 2002.
122. S. Spaccapietra, C. Parent, and Y. Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *VLDB Journal*, 1(1):81–126, 1992.
123. B. Stollberg and A. Zipf. OGC Web Processing Service Interface for Web Service Orchestration Aggregating Geo-processing Services in a Bomb Threat Scenario. In *Proceedings of Web and Wireless Geographical Information Systems*, pages 239–251, 2007.
124. G. Strang and K. Borre. *Linear algebra, geodesy, and GPS.* Wellesley-Cambridge, 1997.
125. G. Stumme and A. Maedche. FCA-MERGE: Bottom-Up Merging of Ontologies. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 225–234, 2001.
126. X. Tanga, W. Kainzb, and H. Wangc. Topological Relations between Fuzzy regions in a Fuzzy Topological Space. *Spatial Analysis-Modeling, Methodology and Applications*, 12(2):S151–S165, 2010.

127. A. H. M. ter Hofstede, W. M. P. van der Aalst, M. Adams, and N. Russell. *Modern Business Process Automation: YAWL and its Support Environment*. Springer-Verlag, 2009.
128. E. Tossebro and M. Nygaard. Abstract and Discrete Models for Uncertain Spatiotemporal Data. In *Proceedings of 14th International Conference on Scientific and Statistical Database Management SSDBM'02*, page 240, 2002.
129. E. Tossebro and M. Nygaard. An Advanced Discrete Model for Uncertain Spatial Data. In *Proceedings of 3th International Conference on Advances in Web-Age Information Management (WAIM'02)*, pages 37–51, 2002.
130. E. Tossebro and M. Nygaard. Uncertainty in Spatiotemporal Databases. In *Proceedings of 2nd International Conference on Advances in Information Systems (ADVIS02)*, pages 43–53, 2002.
131. E. Tossebro and M. Nygaard. A Medium Complexity Discrete Model for Uncertain Spatial Data. In *Proceedings of 7th Int. Database Engineering and Applications Symposium (IDEAS'03)*, pages 376–384, 2003.
132. E. Tossebro and M. Nygaard. A Discrete Model for Topological Relationships between Uncertain Spatial Objects. In *Developments in Spatial Data Handling*, pages 395–406, 2004.
133. C. J. Tuot, M. Sintek, and A. R. Dengel. IVIP - A Scientific Workflow System to Support Experts in Spatial Planning of Crop Production. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM'08:)*, pages 586–591, 2008.
134. H. Uitermark, P. van Oosterom, N. J. I. Mars, and M. Molenaar. Ontology-Based Geographic Data Set Integration. In *Proceedings of the International Workshop on Spatio-Temporal Database Management (STDBM'99)*, pages 60–78, 1999.
135. H. T. J. A. Uitermark. *Ontology-Based Geographic Data Set Integration*. PhD thesis, University of Twente, 2001.
136. Vivid Solutions. *JTS Topology Suite Technical Specifications*. http://www.vividsolutions.com/jts/bin/JTS%20Technical%20Specs.pdf Accessed March 2012.
137. W3C. *Web Services Choreography Description Language Version 1.0*, 2005. http://www.w3.org/TR/ws-cdl-10/ Accessed March 2012.
138. L Wald. Definitions and Terms of Reference in Data Fusion. *International Archives of Photogrammetry and Remote Sensing*, 32(7):651–654, 1999.
139. M. Weidlich, G. Decker, A. Grösskopf, and M. Weske. BPEL to BPMN: The Myth of a Straight-Forward Mapping. In *Proceedings of the On the Move to Meaningful Internet Systems (OTM) Confederated International Conferences*, pages 265–282, 2008.
140. A. Weiser and A. Zipf. Web Service Orchestration of OGC Web Services for Disaster Management. In *Geomatics Solutions for Disaster Management*, pages 239–254, 2007.
141. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag New York, Inc., 2007.
142. M. Weske, G. Vossen, C. B. Medeiros, and F. Pires. Workflow Management in Geoprocessing Applications. In *Proceedings of the 6th ACM International Symposium on Advances in Geographic Information Systems (GIS'98:)*, pages 88–93, 1998.
143. Stephen A. White. *Using BPMN to Model a BPEL Process*. IBM Corp. http://www.omg.org/bpmn/Documents/Mapping_BPMN_to_BPEL_Example.pdf Accessed March 2012.
144. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25(3):38–49, 1992.

145. P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In *Proceedings of the 22nd International Conference on Conceptual Modeling (ER 2003)*, pages 200–215, 2003.
146. P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In *Proceedings of the 4th International Conference Business Process Management (BPM 2006)*, pages 161–176, 2006.
147. P. R. Wolf and C. D. Ghilani. *Adjustment Computations: Statistics and Least Squares in Surveying and GIS*. John Wiley & Sons, 1997.
148. M. F. Worboys and M. Duckham. Integrating Spatio-Thematic Information. In *Proceedings of 2nd International Conference on Geographic Information Science (GIScience 2002)*, pages 346–362, 2002.
149. J. Zhang, D. Pennington, and W. Michener. Automatic Transformation from Geospatial Conceptual Workflow to Executable Workflow using Grass GIS Command Line Modules in Kepler. In *Proceedings of International Conference on Computational Science*, pages 912–919, 2006.
150. J. Zhang, D. D. Pennington, and W. K. Michener. Using Web Services and Scientific Workflow for Species Distribution Prediction Modeling. In *Advances in Web-Age Information Management*, pages 610–617, 2005.

Part III

Appendices

A

Statistical Background

A.1 Probability Theory

An *event* is a collection of outcomes of an experiment. Events consisting of a single outcome are known as simple or elementary events. The collection of all possible outcomes for an experiment is defined as *sample space*. Sure and impossible events are denoted by Ω and $\varnothing$, respectively.

Definition A.1 (Probability). Given a random experiment, the probability of an element A can be approximated by the following ratio:

$$\Pr(A) = lim_{n\to\infty} \frac{m}{n}$$

where n is the number (sufficiently large) of times the experiment is repeated, while m is the number of occurrence of A in the experiment. □

From the definition it follows that $0 \leq \Pr(A) \leq 1$, in particular $\Pr(\Omega) = 1$ and $\Pr(\varnothing) = 0$. Moreover, let A and B two events, the following axioms hold:

1. $0 \leq \Pr(A) \leq 1$ and $0 \leq \Pr(B) \leq 1$
2. $\Pr(\Omega) = 1$ and $\Pr(\varnothing) = 0$
3. $\Pr(A \cup B) = \Pr(A) + \Pr(B) - \Pr(A \cap B)$
4. If $A \supset B$, $\Pr(A) \geq \Pr(B)$ and $\Pr(A) - \Pr(B) = \Pr(A|B)$
   where $\Pr(A|B)$ is the probability that the event A occurs and the event B does not occurs.
5. For any event A, $\Pr(A^C) = 1 - \Pr(A)$
   where A^C is the event "A does not occur" and is called *complement event*.
6. If events $A_1, A_2, \ldots, A_n$ are mutually exclusive and exhaustive events (i.e. at least one of them occurs), the probability of their union is equal to 1: $\Pr(A_1 \cup \cdots \cup A_n) = \Pr(A_1) + \ldots \Pr(A_n) = 1$

Definition A.2 (Random variable). Given a random experiment with sample space Ω, a random variable X is a function $X : \Omega \to \mathbb{R}$.

- The probability $\Pr(X = x)$ of the random variable X equal to a constant x is called **probability density function** of X.

- The probability $\Pr(X \leq x)$ of the random variable X not greater than the constant x is called **distribution function** of X. □

Definition A.3 (Expectation). Let X a random variable with probability density function $f(x)$. The expectation of this random variable is denoted as $E(X)$ and is equal to the weighted average of its outcomes.

$$E(X) = \int_{-\infty}^{\infty} x f(x) dx$$

□

Definition A.4 (Variance). Let X a random variable with probability density function $f(x)$. The variance of this random variable is denoted as $\sigma^2(X)$ or simply σ_X^2. It corresponds to the expectation of the random variable $(X - E(X))^2$.

$$\sigma^2(X) = E[(X - E(X))^2]$$

□

The variance of a random variable X represents the *dispersion* of the distribution around its mean value.

The expectation and the variance have the following properties:

1. $\sigma^2(X) = E(X)^2 - [E(X)]^2$
2. For any constant a, $E(a) =$ a and $\sigma^2(a) = 0$
3. For any constant c, $E(cX) = cE(X)$ and $\sigma^2(xC) = c^2\sigma^2(X)$
4. $E(XY) = E(X)E(Y) + E[(x - E(X)) - (y - E(y))]$
5. Let $x_1, x_2, \ldots, x_n$ be n random variables, the expectation and the variance of their sum are

$$E(x_1 + x_2 + \cdots + x_n) = E(x_1) + E(x_2) + \cdots E(x_n)$$

$$\sigma^2(x_1 + x_2 + \cdots + x_n) = \sum_{i,j=1}^{n} E\left[(x_i - E(x_i))(x_j - E(x_j))\right]$$

6. Let $x_1, x_2, \ldots, x_n$ be n random variables, the expectation of their product is

$$E(x_1, x_2, \cdots, x_n) = E(x_1)E(x_2)\cdots E(x_n)$$

7. Let $x_1, x_2, \ldots, x_n$ be n random variables. Suppose that the expectation and the variance of each of these random variables are $E(x_k) = \mu$ and $\sigma^2(x_k) = \sigma^2$. The expectation and the variance of the random variable $y = \frac{1}{n} \cdot \sum_{k=1}^{n} x_k$ are:

$$E(y) = \mu$$

$$\sigma^2(y) = \frac{\sigma^2}{n}$$

Definition A.5 (Covariance). Let X and Y two random variables, the covariance σ_{XY} of these two variables is given as

$$\sigma_{XY} = E\left[(X - E(X))(Y - E(Y))\right]$$

□

The covariance has the following properties:

1. An alternative computation method is:

$$\begin{aligned}\sigma_{XY} &= E\left[(X - E(X))(Y - E(Y))\right] \\ &= E\left[XY - XE(Y) - YE(X) + E(X)E(Y)\right] \\ &= E(XY) - E(X)E(Y)\end{aligned}$$

2. Let X and Y two *independent* random variables

$$\sigma_{XY} = E(XY) - E(X)E(Y) = 0$$

A.2 Fuzzy Theory

Fuzzy set theory can be applied to represent spatial objects with vague properties.

Definition A.6 (Fuzzy set). Given a universe $\mathcal{U}$, a fuzzy set A in $\mathcal{U}$ is defined by a membership function $\mu_A : \mathcal{U} \to [0,1]$ that associate to each element $u \in \mathcal{U}$ the degree of membership of u belonging to A. If $\mu_A(u) = 0$, element u does not belong to A, while $\mu_A(u) = 1$ means that u definitely belongs to A. The set $\{x \in \mathcal{U} \mid \mu_A(x) > 0\}$ is called support of A.

Let A and B two fuzzy set with membership functions μ_A and μ_B, respectively. Five fundamental operators on fuzzy sets are defined as follows:

1. Inclusion: $A \supseteq B$ if and only if $\forall u \in \mathcal{U} \,.\, \mu_A(u) \geq \mu_B(u)$
2. Equivalence: $A = B$ if and only if $A \supseteq B \wedge B \supseteq A$
3. Union: $C = A \cup B$ if and only if $\forall u \in \mathcal{U} \,.\, \mu_c(u) = max\{\mu_A(u), \mu_B(u)\}$
4. Intersection: $D = A \cap B$ if and only if $\forall u \in \mathcal{U} \,.\, \mu_D(u) = min\{\mu_A(u), \mu_B(u)\}$
5. Complement: $E = A^C$ if and only if $\forall u \in \mathcal{U} \,.\, \mu_E(u) = 1 - \mu_A(u)$

For any nonempty fuzzy set A, B and C the following propositions are true:

1. Indempotency: $A \cup A = A$ and $A \cap A = A$
2. Commutation: $A \cup B = B \cup A$ and $A \cap B = B \cap A$
3. Association: $(A \cup B) \cup C = A \cup (B \cup C)$ and $(A \cap B) \cap C = A \cap (B \cap C)$
4. Absorption: $(A \cap B) \cup A = A$ and $(A \cup B) \cap A = A$
5. Distributivity: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ and $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
6. Identity: $A \cup \mathcal{U} = \mathcal{U}$ and $A \cup \varnothing = A$ and $A \cap \mathcal{U} = A$ and $A \cap \varnothing = \varnothing$
7. Involution: $(A^C)^C = A$
8. De Morgan's law: $(A \cup B)^C = A^C \cap B^C$ and $(A \cap B)^C = A^C \cup B^C$
9. $A \cup A^C \neq \mathcal{U}$
10. $A \cap A^C \neq \varnothing$

Definition A.7 (Fuzzy topology). Let X be a nonempty set and I the closed unit interval. A family δ of fuzzy sets in X is called *fuzzy topology* if and only if:

1. $\varnothing, X \in \delta$
2. if $A, B \in \delta$, then $A \cap B \in \delta$
3. $\forall i \in I \,.\, A_i \in \delta \,.\, \bigcup A_i \in \delta$

The pair (X, δ) is called *fuzzy topological space*. Every member of δ is called an *open* fuzzy set, while its complement, denoted by δ' is a closed fuzzy set.

Definition A.8 (Interior and closure). Given a fuzzy set A:

1. Set A° is said to be the interior of A, if it is an open subset of A; i.e. $A^\circ = \{B \in \delta : B \subseteq A\}$
2. The closure $\overline{A}$ of A is the intersection of all closed subsets of A: $\overline{A} = \bigcap\{B \in \delta' : B \supseteq A\}$

Definition A.9 (Boundary). The boundary of a fuzzy set A is defined as: $\partial A = \overline{A} \cup (\overline{A^C})$.

Definition A.10 (I-Fuzzy subset). Let X be a nonempty set, I be the closed interval $[0, 1]$. An I-fuzzy subset A on X is a mapping (called membership function on A) $\mu_A : X \to I$. The family of all the $[0, 1]$-fuzzy or I-fuzzy subsets on X is denoted as I^X and consists of all mappings from X to I.

The set I^X is called I-fuzzy topological space, X is called the carrier domain of each I-fuzzy subset on it, and I is called the value domain of each I-fuzzy subset on X, while $A \in I^X$ is called crisp subset on X.

B

Workflow Patterns

B.1 Control-Flow Patterns

This section briefly recalls the control-flow patterns presented in [109].

- *Basic Control-Flow Patterns* capture elementary aspects of process control.
  - *Sequence* (WCP-01): a task in a process is enabled after the completion of a preceding activity in the same process.
  - *Parallel Split* (WCP-02): the divergence of a branch into two or more parallel branches each of which execute concurrently.
  - *Synchronization* (WCP-03): the convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled.
  - *Exclusive Choice* (WCP-04): the divergence of a branch into two or more branches such that when the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logic expression associated with the branch.
  - *Simple Merge* (WCP-05): the convergence of two or more branches into a single subsequent branch such that each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.
- *Advanced Branching and Syncronization Patterns* this group characterize more complex thread branching and merging concepts which arise in business processes.
  - *Multi-Choice* (WCP-06): the divergence of a branch into two or more branches. When the incoming branch is enabled, the thread of control is immediately passed to one or more of the outgoing branches based on the outcome of distinct logic expressions associated to each branch.
  - *Structured Synchronizing Merge* (WCP-07): the convergence of two or more branches (which diverged earlier in the process at a uniquely identifiable point) into a single subsequent branch. The thread of control is passed to the subsequent branch when each active incoming branch has been enabled. The Structured Synchronizing Merge occurs in a structured context, i.e. there must be a single Multi-Choice construct earlier in the process model with which the Structured Synchronizing Merge is associated and it must merge

all of the branches emanating from the Multi-Choice. These branches must either flow from the Structured Synchronizing Merge without any splits or joins or they must be structured in form (i.e. balanced splits and joins).

- *Multi-Merge* (WCP-08): the convergence of two or more branches into a single subsequent branch such that each enablement of an incoming branch results in the thread of control being passed to the subsequent branch.
- *Structured Discriminator* (WCP-09): the convergence of two or more branches into a single subsequent branch following a corresponding divergence earlier in the process model such that the thread of control is passed to the subsequent branch when the first incoming branch has been enabled. Subsequent enablements of incoming branches do not result in the thread of control being passed on. The Structured Discriminator construct resets when all incoming branches have been enabled. The Structured Discriminator occurs in a structured context, i.e. there must be a single Parallel Split construct earlier in the process model with which the Structured Discriminator is associated and it must merge all of the branches emanating from the Structured Discriminator. These branches must either flow from the Parallel Split to the Structured Discriminator without any splits or joins or they must be structured in form (i.e. balanced splits and joins).
- *Blocking Discriminator* (WCP-28): the convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the first active incoming branch has been enabled. The Blocking Discriminator construct resets when all active incoming branches have been enabled once for the same process instance. Subsequent enablements of incoming branches are blocked until the Blocking Discriminator has reset.
- *Cancelling Discriminator* (WCP-29): the convergence of two or more branches into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when the first active incoming branch has been enabled. Triggering the Cancelling Discriminator also cancels the execution of all of the other incoming branches and resets the construct.
- *Structured Partial Join* (WCP-30): the convergence of two or more branches (say m) into a single subsequent branch following a corresponding divergence earlier in the process model such that the thread of control is passed to the subsequent branch when k of the incoming branches have been enabled, where k is less than m. Subsequent enablements of incoming branches do not result in the thread of control being passed on. The join construct resets when all active incoming branches have been enabled. The join occurs in a structured context, i.e. there must be a single Parallel Split construct earlier in the process model with which the join is associated and it must merge all of the branches emanating from the Parallel Split. These branches must either flow from the Parallel Split to the join without any splits or joins or be structured in form (i.e. balanced splits and joins).
- *Blocking Partial Join* (WCP-31): the convergence of two or more branches (say n) into a single subsequent branch following one or more corresponding

divergences earlier in the process model. The thread of control is passed to the subsequent branch when k of the incoming branches has been enabled (where $2 < k < m$). The join construct resets when all active incoming branches have been enabled once for the same process instance. Subsequent enablements of incoming branches are blocked until the join has reset.
  - *Cancelling Partial Join* (WCP-32): the convergence of two or more branches (say n) into a single subsequent branch following one or more corresponding divergences earlier in the process model. The thread of control is passed to the subsequent branch when k of the incoming branches have been enabled where k is less than n. Triggering the join also cancels the execution of all of the other incoming branches and resets the construct.
  - *Generalised AND-Join* (WCP-33): the convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled. Additional triggers received on one or more branches between firings of the join persist and are retained for future firings.
  - *Local Synchronizing Merge* (WCP-37): the convergence of two or more branches which diverged earlier in the process into a single subsequent branch such that the thread of control is passed to the subsequent branch when each active incoming branch has been enabled. Determination of how many branches require synchronization is made on the basis on information locally available to the merge construct. This may be communicated directly to the merge by the preceding diverging construct or alternatively it can be determined on the basis of local data such as the threads of control arriving at the merge.
  - *General Synchronizing Merge* (WCP-38): the convergence of two or more branches which diverged earlier in the process into a single subsequent branch such that the thread of control is passed to the subsequent branch when either (1) each active incoming branch has been enabled or (2) it is not possible that any branch that has not yet been enabled will be enabled at any future time.
  - *Thread Merge* (WCP-41): at a given point in a process, a nominated number of execution threads in a single branch of the same process instance should be merged together into a single thread of execution.
  - *Thread Split* (WCP-42): at a given point in a process, a nominated number of execution threads can be initiated in a single branch of the same process instance.
- *Multiple Instance Patterns* describe situations where multiple concurrent instances of a task or sub-process execute simultaneously and may need to be synchronized upon completion.
  - *Multiple Instances without Synchronization* (WCP-12): within a given process instance, multiple instances of a task can be created. These instances are independent of each other and run concurrently. There is no requirement to synchronize them upon completion.
  - *Multiple Instances with a Priori Design-Time Knowledge* (WCP-13): within a given process instance, multiple instances of a task can be created. The required number of instances is known at design time. These instances are

independent of each other and run concurrently. It is necessary to synchronize the task instances at completion before any subsequent tasks can be triggered.

- *Multiple Instances with a Priori Run-Time Knowledge* (WCP-14): within a given process instance, multiple instances of a task can be created. The required number of instances may depend on a number of run-time factors, including state data, resource availability and inter-process communications, but is known before the task instances must be created. Once initiated, these instances are independent of each other and run concurrently. It is necessary to synchronize the instances at completion before any subsequent tasks can be triggered.
- *Multiple Instances without a Priori Run-Time Knowledge* (WCP-15): within a given process instance, multiple instances of a task can be created. The required number of instances may depend on a number of runtime factors, including state data, resource availability and inter-process communications and is not known until the final instance has completed. Once initiated, these instances are independent of each other and run concurrently. At any time, whilst instances are running, it is possible for additional instances to be initiated. It is neccesary to synchronize the instances at completion before any subsequent tasks can be triggered.
- *Static Partial Join for Multiple Instances* (WCP-34): within a given process instance, multiple concurrent instances of a task (say m) can be created. The required number of instances is known when the first task instance commences. Once n of the task instances have completed (where n is less than m), the next task in the process is triggered. Subsequent completions of the remaining m-n instances are inconsequential, however all instances must have completed in order for the join construct to reset and be subsequently re-enabled.
- *Cancelling Partial Join for Multiple Instances* (WCP-35): within a given process instance, multiple concurrent instances of a task (say m) can be created. The required number of instances is known when the first task instance commences. Once n of the task instances have completed (where n is less than m), the next task in the process is triggered and the remaining m-n instances are cancelled.
- *Dynamic Partial Join for Multiple Instances* (WCP-36): within a given process instance, multiple concurrent instances of a task can be created. The required number of instances may depend on a number of runtime factors, including state data, resource availability and inter-process communications and is not known until the final instance has completed. At any time, whilst instances are running, it is possible for additional instances to be initiated providing the ability to do so had not been disabled. A completion condition is specified which is evaluated each time an instance of the task completes. Once the completion condition evaluates to true, the next task in the process is triggered. Subsequent completions of the remaining task instances are inconsequential and no new instances can be created.

- *State-Based Patterns* reflects situations for which the execution depends upon the notion of state.

  - *Deferred Choice* (WCP-16): a point in a process where one of several branches is chosen based on interaction with the operating environment. Prior to the decision, all branches represent possible future courses of execution. The decision is made by initiating the first task in one of the branches i.e. there is no explicit choice but rather a race between different branches. After the decision is made, execution alternatives in branches other than the one selected are withdrawn.
  - *Interleaved Partial Order Routing* (WCP-17): a set of tasks has a partial ordering defining the order in which they must be executed. Each task in the set must be executed once and they can by completed in any order according with the partial order. However, as an additional requirement, no two tasks can be executed at the same time (i.e. no two tasks can be active for the same process instance at the same time).
  - *Milestone* (WCP-18): a task is only enabled when the process instance (of which it is part) is in a specific state (typically a parallel branch). The state is assumed to be a specific execution point (also known as a milestone) in the process model. When this execution point is reached the nominated task can be enabled. If the process instance has progressed beyond this state, then the task cannot be enabled now or at any future time (i.e. the deadline has expired). Note that the execution does not influence the state itself, i.e. unlike normal control-flow dependencies it is a test rather than a trigger.
  - *Critical Section* (WCP-39): two or more connected subgraphs of a process model are identified as "critical sections". At runtime for a given process instance, only tasks in one of these "critical sections" can be active at any given time. Once execution of the tasks in one "critical section" commences, it must complete before another "critical section" can commence.
  - *Interleaved Routing* (WCP-40): each member of a set of tasks must be executed once. They can be executed in any order but no two tasks can be executed at the same time (i.e. no two tasks can be active for the same process instance at the same time). Once all of the tasks have completed, the next task in the process can be initiated.
- *Cancelation and Force Completion Patterns* categorize the various cancelation scenarios that may arise in a business process.
  - *Cancel Activity* (WCP-19): an enabled activity is withdrawn prior to its commencing execution. If the activity has started, it is disabled and, where possible, the currently running instance is halted and removed.
  - *Cancel Case* (WCP-20): a complete process instance is removed. This includes currently executing activities, those which may execute at some future time and all sub-processes. The process instance is recorded as having completed unsuccessfully.
  - *Cancel Region* (WCP-25): the ability to disable a set of activities in a process instance. If any of the activities are already executing, then they are withdrawn. The activities need not ba a connected subset of the overall process model.
  - *Cancel Multiple Instance Activity* (WCP-26): within a given process instance, multiple instances of a task can be created. The required number of instances is known at design time. These instances are independent of each

other and run concurrently. At any time, the multiple instance task can be cancelled and any instances which have not completed are withdrawn. Task instances that have already completed are unaffected.
  - *Complete Multiple Instance Activity* (WCP-27): within a given process instance, multiple instances of a task can be created. The required number of instances is known at design time. These instances are independent of each other and run concurrently. It is necessary to synchronize the instances at completion before any subsequent tasks can be triggered. During the course of execution, it is possible that the task needs to be forcibly completed such that any remaining instances are withdrawn and the thread of control is passed to subsequent tasks.
- *Iteration Patterns* describe various ways in which repetitive tasks or subprocesses can be specified in a process.
  - *Arbitrary Cycles* (WCP-10): the ability to represent cycles in a process model that have more than one entry or exit point. It must be possible for individual entry and exit points to be associated with distinct branches.
  - *Structured Loop* (WCP-21): the ability to execute a task or sub-process repeatedly. The loop has either a pre-test or post-test condition associated with it that is either evaluated at the beginning or end of the loop to determine whether it should continue. The looping structure has a single entry and exit point.
  - *Recursion* (WCP-22): the ability of a task to invoke itself during its execution or an ancestor in terms of the overall decomposition structure with which it is associated.
- *Termination Patterns* address the issue of when the execution of a process can be considered to be finished.
  - *Implicit Termination* (WCP-11): a given process (or sub-process) instance should terminate when there are no remaining work items that are able to be done either now or at any time in the future and the process instance is not in deadlock.
  - *Explicit Termination* (WCP-43): a given process (or sub-process) instance should terminate when it reaches a nominated state. Typically this is denoted by a specific end node. When this end node is reached, any remaining work in the process instance is cancelled and the overall process instance is recorded as having completed successfully, regardless of whether there are any tasks in progress or remaining to be executed.
- *Trigger Patterns* identify constructs that allow to continue/start the execution of a process only when a signal from the operating environment has been reached.
  - *Transient Trigger* (WCP-23): the ability for a task instance to be triggered by a signal from another part of the process or from the external environment. These triggers are transient in nature and are lost if not acted on immediately by the receiving task. A trigger can only be utilized if there is a task instance waiting for it at the time it is received.
  - *Persistent Trigger* (WCP-24): the ability for a task to be triggered by a signal from another part of the process or from the external environment.

These triggers are persistent in form and are retained by the process until they can be acted on by the receiving task.

B.2 Data Patterns

This section briefly recalls the data patterns presented in [108].

- *Data Visibility* patterns define the binding of a data element and its region of visibility.
  - *Task Data* (WDP-01) corresponds to a data element defined in the context of a particular task. It is typically only accessible within a given task instance and has the same life-span as the task instance to which it is bound.
  - *Block Data* (WDP-02) corresponds to a data element defined in the context of a particular block or net within a process that is hierarchical in form. The data element is visible throughout the block and it has a life-span corresponding to the life-span of the block.
  - *Scope Data* (WDP-03) corresponds to a data element bound to a set of tasks in a process. If the process is hierarchical in form, the tasks are assumed to be in the same block although they need not be directly connected. A scope data element is visible only to the tasks of which the scope is comprised. It has the same lifespan as the block in which it resides.
  - *Multiple Instance Data* (WDP-04) corresponds to a data element in a specific executioninstance of a task where the task may execute multiple times, possibly concurrently.
  - *Case Data* (WDP-05) corresponds to a data element that is accessible to all tasks within a process instance. It has a life-span that is the same as the process instance.
  - *Folder Data* (WDP-06) corresponds to a data element that is defined outside the context of the process that is able to be coupled with particular process instances during their execution. Once married up with a process instance, the data element is accessible to all tasks in the process instance.
  - *Workflow Data* (WDP-07) corresponds to a data element that is accessible throughout all tasks in cases for a given process.
  - *Environment Data* (WDP-08) correspond to a data element that is defined outside the context of the process but accessible within it during execution. It is possible for an environment data element to be accessed by different case of different processes.
- *Data Interaction Patterns* are of two main types: *internal* data interaction patterns, which describe the various ways in which data elements are communicated between the main components in a process (the first four ones), and *external* data interaction patterns, which describe the various ways in which data elements are communicated between the main components in a process and the operating environment (the last four ones).
  - *Data Interaction between Tasks* (WDP-09) distinguishes the way in which data elements are passed between tasks in a process instance.

- *Data Interaction between Block Task and Sub-process Decomposition* (WDP-10, WDP-11) describes the manner in which data elements are passed to and from a composite task which has an associated sub-process decomposition.
- *Data Interaction with Multiple Instance Tasks* (WDP-12, WDP-13) describes the manner in which data is passed to and from a multiple instance task.
- *Data Interaction between Cases* (WDP-14) denotes the communication of a data element from one case to another.
- *Data Interaction Task to Environment Push-Oriented* (WDP-15): the ability of a task to initiate the passing of data elements to a resource or service in the operating environment.
- *Data Interaction Environment to Task Pull-Oriented* (WDP-16): the ability of a task to request data elements from resources or services in the operational environment.
- *Data Interaction Environment to Task Push-Oriented* (WDP-17): the ability for a task to receive and utilise data elements passed to it from services and resources in the operating environment on an unscheduled basis.
- *Data Interaction Task to Environment Pull-Oriented* (WDP-18): the ability of a task to receive and respond to requests for data elements from services and resources in the operational environment.
- *Data Interaction - Case to Environment - Push-Oriented* (WDP-19): the ability of a case to initiate the passing of data elements to a resource or service in the operational environment.
- *Data Interaction - Environment to Case - Pull-Oriented* (WDP-20): the ability of a case to request data from services or resources in the operational environment.
- *Data Interaction - Environment to Case - Push-Oriented* (WDP-21): the ability of a case to accept data elements passed to it from services or resources in the operating environment.
- *Data Interaction - Case to Environment - Pull-Oriented* (WDP-22): the ability of a case to respond to requests for data elements from a service or resource in the operating environment.
- *Data Interaction - Workflow to Environment - Push-Oriented* (WDP-23): the ability of a process environment to pass data elements to resources or services in the operational environment.
- *Data Interaction - Environment to Workflow - Pull-Oriented* (WDP-24): the ability of a process environment to pass data elements to resources or services in the operational environment.
- *Data Interaction - Environment to Workflow - Push-Oriented* (WDP-25): the ability of services or resources in the operating environment to pass global data to a process.
- *Data Interaction - Workflow to Environment - Pull-Oriented* (WDP-26): the ability of services or resources in the operating environment to pass global data to a process.

- *Data transfer patterns* deal with the mechanics of how a data element is actually transported to or from a task instance in an executing process.

  - *Data Transfer by Value* (WDP-27, WDP-28) recognizes the situation where the transport of a data element is actually based on copying its value between one task and another.
  - *Data Transfer – Copy In/Copy Out* (WDP-29) corresponds to the situation where the data elements are copied from an external source (either within or outside the process environment) into its address space at the commencement of execution and to copy their final values back at completion.
  - *Data Transfer by Reference – Unlocked* (WDP-30) corresponds to the scenario where data elements are passed by reference rather than by value.
  - *Data Transfer by Reference – With Lock* (WDP-31) is an extension of the preceding pattern that also provides a measure of concurrency control by registering a lock over the transferred data element to ensure that the receiving task retains complete control over it and can rely on its integrity.
  - *Data Transformation – Input* (WDP-32) corresponds to the ability to apply a transformation function to a data element prior to it being passed to a process component.
  - *Data Transformation – Output* (WDP-33) corresponds to the ability to apply a transformation function to a data element immediately prior to it being passed out of a process component.
- *Data-based Routing* capture the various ways in which data elements can interact with other perspectives (e.g. control-flow) and influence the overall operation of a process instance.
  - *Task Precondition – Data Existence* (WDP-34) represents the ability to define a precondition for tasks based on the presence of data elements at the time of execution.
  - *Task Precondition – Data Value* (WDP-35) represents the ability to define a precondition for tasks based on the value of specific parameters at the time of execution.
  - *Task Postcondition – Data Existence* (WDP-36) represents the ability to define a precondition for tasks based on the presence of data elements at the time of completion.
  - *Task Postcondition – Data Value* (WDP-37) represents the ability to define a precondition for tasks based on the value of specific parameters at the time of completion.
  - *Event-based Task Trigger* (WDP-38) represents the ability for an external event to initiate a task and to pass data elements to it.
  - *Data-based Task Trigger* (WDP-39) reprsents the ability to trigger a specific task when an expression based on data elements in the process instance evaluates to true.
  - *Data-based Routing* (WDP-40) provides the ability to alter the control-flow within a case based on the evaluation of data-based expressions. A data-based routing expression is associated with each outgoing arc of an OR-split or XOR-split.

B.3 Resource Patterns

This section briefly recalls the resource patterns presented in [111].

- *Creation Patterns* describe the various ways in which work items can be allocated to one or several resources at run-time.
  - *Direct Distribution* (WRP-1) corresponds to the situation where a work item is directly offered or allocated to one or more specifically named resource.
  - *Role-based Distribution* (WRP-2) corresponds to the situation where a work item is directly offered or allocated to one or more specifically named roles, each of which contain one or more users.
  - *Deferred Distribution* (WRP-3) corresponds to the situation where the identification of the resource that a work item will be offered or allocated to is delayed until run-time, typically by nominating a data resource from which it can be obtained.
  - *Authorization* (WRP-4) identifies privileges that can be assigned to specific resources during the execution of a process. These privileges define the range of actions that the resource may undertake during the execution of the process.
  - *Separation of Duties* (WRP-5) corresponds to a constraint that exists between two tasks requiring that they not be executed by the same user within a given process instance.
  - *Case Handling* (WRP-6) corresponds to the ability to allocate all work items in a case to the same resource at the time of commencement.
  - *Retain Familiar* (WRP-7) corresponds to a constraint that exists between two tasks requiring that where possible they be executed by the same user within a given process instance.
  - *Capability-based Distribution* (WRP-8) corresponds to the situation where a work item is offered or allocated to one or more resources based on capabilities that they possess.
  - *Organization-based Distribution* (WRP-9) corresponds to the situation where a work item is offered or allocated to one or more resources based on their position or other responsibilities within the organization.
  - *History-based Distribution* (WRP-10) corresponds to the situation where a work item is offered or allocated to one or more resources based on their preceding execution history.
  - *Automatic Execution* (WRP-11) corresponds to the situation where a work item can be executed without needing to be distributed to a resource.
- *Push Patterns* apply in situations where work items are forwarded to resources by the system that is automating a business process.
  - *Distribution by Offer Single Resource* (WRP-12) corresponds to the situation where a work item is offered to a specific resource on a non-binding basis.
  - *Distribution by Offer Multiple Resources* (WRP-13) corresponds to the situation where a work item is offered to several resources on a non-binding basis with expectation that one of them might commit to undertaking it.
  - *Distribution by Allocation Single Resource* (WRP-14) corresponds to the situation where a work item is allocated to a specific resource on a binding basis and they are expected to complete it at some future time.
  - *Random Allocation* (WRP-15) corresponds to the allocation of a work item to a specific resource selected from a group of resources on a random basis.

  - *Round Robin Allocation* (WRP-16) corresponds to the allocation of a work item to a specific resource selected from a group of resources on a round robin basis basis.
  - *Shortest Queue* (WRP-17) corresponds to the allocation of a work item to a specific resource selected from a group of resources based on who has the least work pending (i.e. the shortest work queue).
  - *Early Distribution* (WRP-18) corresponds to the situation where a work item can be offered or allocated to a resource ahead of the time that it is actually enabled and can be completed.
  - *Distribution on Enablement* (WRP-19) corresponds to the situation where a work item is offered or allocated to a resource at the same time that it is enabled and can be completed.
  - *Late Distribution* (WRP-20) corresponds to the situation where a work item is offered or allocated to a resource at some time after the time at which it is enabled.
- *Pull Patterns* correspond to work distribution actions that are initiated by the actual resources undertaking them.
  - *Resource-Initiated Allocation* (WRP-21) corresponds to the situation where a resource commits to undertaking a work item that has been offered to them.
  - *Resource-Initiated Execution Allocated Work Item* (WRP-22) corresponds to the situation where a resource starts a work item that has been offered to them.
  - *Resource-Initiated Execution Offered Work Item* (WRP-23) corresponds to the situation where a resource commits to undertake and immediately starts a work item that has been offered to them.
  - *System-Determined Work Queue Content* (WRP-24) corresponds to the ability for the system to impose an ordering on the sequence in which resources see and/or can undertake their work items.
  - *Resource-Determined Work Queue Content* (WRP-25) corresponds to the ability for the resource to impose an ordering/reordering on the sequence in which they see and/or can undertake their work items.
  - *Selection Autonomy* (WRP-26) corresponds to the ability for a resource to choose which work item they undertake next.
- *Detour Patterns* identify approaches for deviating from the work distribution strategy implied by the process model and provide various means for resources, as well as the enabling system itself, to deal with unexpected or undesirable workload situations that may arise.
  - *Delegation* (WRP-27) corresponds to the situation where a resource allocates a work item that is currently allocated to them to another resource.
  - *Escalation* (WRP-28) corresponds to the situation where the system changes an existing work item offer or allocation and redistributes the work item to another user with the goal of expediting a work item.
  - *Deallocation* (WRP-29) corresponds to the situation where a resource (or group of resources) chooses to make a work item previously offered or allocated to them available for redistribution to other resources.

  - *Stateful Reallocation* (WRP-30) corresponds to the situation where a resource chooses to allocate a work item that they have already started (but not completed) executing to another resource and to retain any associated state information.
  - *Stateless Reallocation* (WRP-31) corresponds to the situation where a resource chooses to allocate a work item that they have already started (but not completed) executing to another resource without retaining any associated state information.
  - *Suspension/Resumption* (WRP-32) corresponds to the ability of a resource to suspend and resume execution of a work item.
  - *Skip* (WRP-33) corresponds to the ability of a resource to skip a work item allocated to it and mark the work item as complete.
  - *Redo* (WRP-34) corresponds to the ability of a resource to redo a work item that has previously been completed in a case. This may necessitate that work items subsequent to it (and hence dependent on its results) also be repeated.
  - *Pre-Do* (WRP-35) corresponds to the ability of a resource to execute a work item in the current case ahead of the time that it has been offered or allocated to any resources.
- *Auto-Start Patterns* identify various means of expediting process execution by automating various aspects of work item handling.
  - *Commencement on Creation* (WRP-36) corresponds to the ability of a resource to commence execution on a work item as soon as it is created.
  - *Commencement on Allocation* (WRP-37) corresponds to the ability of a resource to commence execution on a work item as soon as it is allocated to them.
  - *Chained Execution* (WRP-39) corresponds the ability to automatically start the next work item in a case once the previous one has completed.
  - *Piled Execution* (WRP-38) corresponds to the ability to initiate the next instance of a work item corresponding to a given task (perhaps in a different case) once the previous one has completed such that all work items are allocated to the same resource.
- *Visibility Patterns* delineate the ability to configure the extent of disclosure about the state of progress on particular work items.
  - *Configurable Unallocated Work Item Visibility* (WRP-40) corresponds to the ability to configure the visibility of unallocated work items by process participants.
  - *Configurable Allocated Work Item Visibility* (WRP-41) corresponds to the ability to configure the visibility of allocated or executing work items by process participants.
- *Multiple resource patterns* identify work situations where the correspondence between work items and resources is not one-to-one.
  - *Simultaneous Execution* (WRP-42) corresponds to the ability for a resource to execute more than one work item simultaneously.
  - *Additional Resources* (WCP-43) corresponds to the ability for a given resource to request additional resources to assist in the execution of a work item that it is currently undertaking.