A Method for Data Minimization in Personal Information Sharing

Prima Gustiene and Remigijus Gustas

Department of Information Systems, Karlstad University, Sweden
{Prima.Gustiene, Remigijus.Gustas}@kau.se

Abstract. A fundamental privacy principle, which is enforced in many privacy-enhancing technologies, is data minimization, i.e. the amount of personal data that are revealed to others and extend to which they are processed should be minimized. Privacy-enhancing identity management is important for processing personal data, the purpose of which is to protect personal data. This is especially relevant for communication via Internet where users are leaving much personal data. Privacy issues should be embedded into a system's core functionality. Minimization of data should be maintained and controlled throughout the systems lifecycle, from the early stages of system analysis and design to implementation. The primary goal of this paper is to present a conceptual modelling method, including the framework, modelling process and the basic modelling constructs, which enables minimization of data. Data cannot be analysed separately without taken into account the processes that cause the changes of data as well as goals. Analysis of relevant data contributes to the problem of data minimization in privacy-enhancing technologies.

Keywords: conceptual modelling, data analysis and minimization, privacy enhancement, service-oriented modelling method.

1 Introduction

Privacy is an essential and fundamental human right (Schütz & Friedewald, 2011). Growing technological possibilities enable transformation of our society towards a computerised social community highly dependent on information sharing. Information sharing increases privacy problems e.g., unauthorised access to personal data and using it for unexpected purposes. Privacy involves the protection of personal information. Information privacy is one of the aspects of the concept of privacy, which is related to the person's right to determine what, when and how personal information can be communicated to various recipients (Westin, 1967). The best protection of personal information is when the information is not revealed at all, but this is not possible in this computerised society.

Nowadays business more and more takes place on the Internet. This situation increases problems to secure personal data. The processing of the personal data is usually not transparent for the users, but it can have painful consequences to privacy and security of the users. Sophisticated technologies provide possibilities to trace, store and use non-protected data for different purposes. The question is *what* data and *how much data* should be collected, stored and shared doing business online. The answer could be found in data analysis and design methods, which could help to analyse and minimize the amount of personal data that are revealed to others. The problem today is that the methods are not pragmatic-driven and they have not enough semantic power for analysis of data.

Privacy-enhancing identity management (IDM) systems are designed to enforce legal privacy requirements to guarantee the processing of privacy compliant data (Fischer-Hübner & Hedbom, 2008). IDM provides technical means that enables protection of legal privacy principles. Identity management systems manage different identities of a person, helping in processing their personal data and making data life cycle management more transparent. Difficult problems in the area of personal data processing such as lack of data minimization and data life cycle management require the new research solutions for reducing problems of personal data processing. There is a lack of a method that provides systematic guiding principles for aligning the overall systems design with respect to privacy-enhancement mechanisms. These mechanisms must be analysed in the context of a larger inter-organisational system between service requester and service provider. As every system is unique, privacy issues should be integrated with other types of system requirements. To integrate privacy issues into the whole enterprise development system, it is necessary to consider these issues, as a part of a new system development life cycle, from the early stages of business goals analysis, requirements analysis and design to delivery of the system.

Data is an important asset concerning information privacy. Privacy-enhancing identity management systems require accurate data analysis for processing personal data. Data is created, processed and consumed in various transactions for different operational and analytical purposes. As data is the main element for the management of information privacy, it is critical to identify a minimal amount of data, which are relevant in the specific context or scenario. Interdependencies among models that represent different aspects of the system cannot be analysed in isolation. Business data and business processes should be analysed together. Just having an integrated and systematic modelling method provides possibility to analyse structural, interactive and behavioural aspects of a system together. Such method can be applied for the analysis of different scenarios including such non-functional requirements as information privacy issues. The main goal of this paper is to present a framework of a service-oriented modelling method and a modelling process that could be applied for data analysis and design to solve data minimisation problems.

2 Model Driven Architecture and Modelling Levels

Model Driven Architecture (MDA) (OMG, 2010) is an approach for model-driven engineering of software systems. It provides a set of guidelines for the structuring of specifications, which are expressed as three models: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model. The important feature of MDA is the mapping rules and techniques used to modify one model into another. Still, a question mapping rule automation between CIM and PIM levels is feasible and remains a major research effort. Pragmatic and semantic levels are supporting computation-neutral modelling (CNM) and are similar to CIM at MDA framework (see figure 1). Technology-oriented design is done at a syntactic level. It is implementation specific modelling (ISM), which can be compared with PIM at MDA framework.

Pragmatic specifications aim to provide motivation for conceptual representations of enterprise components at the semantic level that defines business processes across organisational and technical system boundaries. Pragmatic knowledge, expressed in terms of pragmatic entities such as goals, problems and opportunities, provides motivation for conceptual representations of enterprise components, which take part in various business processes of an enterprise. To structure the pragmatic knowledge about business processes (as services) is important, because such knowledge provides motivation for various configurations of service architectures and defines the *'why'* aspect of the problem domain. Pragmatic specifications are necessary for several reasons. They motivate service events and show the guidelines over how pragmatic aspects are mapped to conceptual representations, which define the semantics of business design, including the structural, behavioural and interactive aspects of business processes.

Pragmatic dependencies can be viewed as modelling basis to reason about the intentions of designers related to new solutions. Pragmatic entities such as goals, problems, and opportunities can be related by pragmatic dependencies and analysed in different situations. Any business process functionality can be defined as a service. A service, in different contexts, from pragmatic point of view can be regarded as different pragmatic entities such as a problem, opportunity or a goal. Business activities can be defined in terms of a set of interaction loops between service requester and a service provider, and linked to design goals (Gustas & Gustiene, 2008). Behind every business process is a clear motivation or goal, which can be analysed together with a final process state. The achievement of this state should bring value to customer. Goal hierarchies can help to identify missing processes and data. Goals also provide a basis for reasoning about the semantic incompleteness of system specifications.

Pragmatic specifications aim to provide justification for conceptual representations of data and processes. To understand *how* and *why* technical system components are useful and how they fit into the overall organizational system at least three modelling levels of information system specifications are necessary: pragmatic level, semantic level and syntactic level (Gustas & Gustiene, 2009). According to FRISCO report (Falkenberg et al., 1996) these three levels are of great interest in the context of information systems design as they deal with the usage, meaning, and structures of system representations. Architectural framework for service-oriented modelling is presented in figure 1 and is explained below.

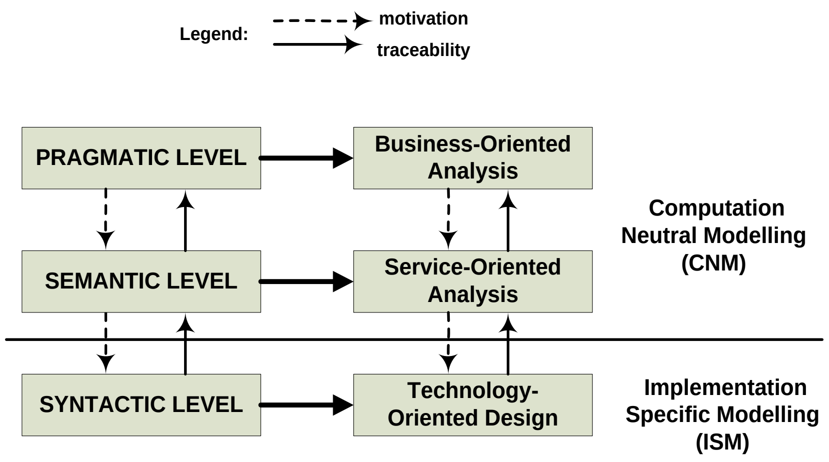
Legend:
motivation
traceability
PRAGMATIC LEVEL
Business-Oriented Analysis
SEMANTIC LEVEL
Service-Oriented Analysis
SYNTACTIC LEVEL
Technology-Oriented Design
Computation Neutral Modelling (CNM)
Implementation Specific Modelling (ISM)

Fig.1. Architectural framework for service-oriented modelling

Pragmatic level is the level where business-oriented analysis is done. Analysis at this level using pragmatic dependencies (Gustiene, 2010) is supposed to drive a system engineering process from business goals to service interactions. Here the decision of which information is necessary and why takes place. *Semantic level* is important for integration of interactive 'where' and 'who', structural 'what' and behavioural 'how' and 'when' aspects (Zachman, 1987) of conceptual representations. Semantic descriptions constrain the implementation specific representations. *Syntactic level* defines the details, which explain the data processing needs for specific application or software component.

The fitness of system specifications between these levels is critical for the success of the final result. Consistency between levels much depends on having appropriate modelling techniques for the refinement of pragmatic entities that justify and represent their structural and dynamic aspects at the semantic level. An integrated modelling way provides possibility to check consistency between levels. It also underlines the possibilities for requirements traceability, which is crucial for verification and validation of system requirements (Maciaszek, 2005). Such three-level architectural framework is the foundation of modelling that helps to provide interplay between business-oriented analysis, service-oriented analysis and technology-oriented design, which are critical for relevant data and process analysis.

3 Perspectives of Service-Oriented Modelling Method

Service-oriented modelling method for information system design (Gustiene, 2010), (Gustas & Gustiene, 2012) is a method based on new principles of service-oriented analysis and design. This method puts into foreground the modelling of interactions (Dietz, 2001) among various enterprise actors. From ontological point of view (Dietz, 2006), every enterprise system could be seen as a composition of different actors that could be viewed as organizational and technical components. The interaction among them is motivated according to the strategic goals of some specific scenario that could be seen as part of some problem domain. The uniqueness of the method is that it is based on the principles of service orientation. Business processes can be analysed as a composition of service interactions. Service-oriented representations are built by conceptualizing interactions among organizational and technical components, which are viewed as various types of enterprise actors. Continuity of interaction loops is the main principle of service orientation. Modelling of interactions between different types of enterprise actors is critical from system analysis and design point of view for several reasons. Explicit modelling of interactions helps to develop an integrated graphical representation of business data and processes. Interaction dependencies among different enterprise actors are important for motivating data transition events and effects (Gustas, 2011). Interaction dependencies provide the possibility to preserve the modularity of crosscutting concerns between different components and to integrate behavioural effects with structural changes in various classes of objects, which represent different data.

The definition of service presented in this method explains the necessary elements and provides with the guidelines how these elements are related to form service architecture. There are two ontological perspectives of communication action (Dietz, 2006), (Gustas & Gustiene, 2009) that lie in the foundation of the main construct for service-oriented modelling: *intersubjective* and *objective.*

The intersubjective perspective defines how actors (service requesters and service providers) are related to each other. This perspective is important as it presents the actors, as independent loosely coupled components, who add value by performing some activities. It signifies certain commitment and responsibilities between enterprise actors (Ferrario & Guarino, 2008).

The *objective* perspective defines how different objects change when the actions during interaction process take place. The objective perspective can be applied to represent the internal behaviour of the objects. It represents data, which is analysed in the context of interaction between organizational and technical system components. Interactions among different actors can be used to manifest object property changes that are results of different actions. Property changes are important for eliciting of semantic meaning of the problem domain. The cohesion of these two perspectives results into a single modelling notation (construct), which allows the integration of static and dynamic aspects of the system, which are important to maintain a holistic representation where external and internal views of service conceptualizations are visualized together.

A starting point of the ontological definition of an enterprise system in the presented service-oriented foundation is quite similar to ontological understanding of system and enterprise as a system. Enterprise system is a composition of the organizational and technical components, which are viewed as various types of enterprise *actors* and which interact as service requesters and service providers. Actors are subsystems that are represented by individuals, organizations and their divisions or roles, which denote groups of people. Technical actors are subsystems such as machines, software and hardware components, etc. Any two actors can be linked by inheritance, composition, classification or interaction dependencies, which are represented graphically in figure 2.

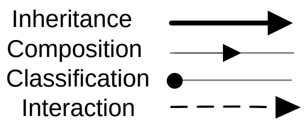

Fig.2. Actor dependencies

Inheritance dependency between actors is used for sharing the static and dynamic similarities. More specific actors inherit the composition and interaction dependencies from more general actors. Dependencies represent additional intrinsic actor interoperation features and structural properties that are prescribed by an enterprise system. *Composition* is a conceptual dependency used to relate a whole to other concepts that are viewed as parts. It is a stricter semantic relation as compared to an aggregation and a composition that is defined in the object-oriented approaches.

Classification link between two actors is used to define their instances. In conceptual modelling, an instance can be viewed as an element of a set that is defined by a concept it belongs to. In the same way as an object can be manipulated by operations, an actor has interaction privileges and responsibilities that are defined by the interaction dependencies.

Interaction dependencies are used to conceptualize services between various enterprise system actors. Since actors are implemented as organizational and technical system components, they can use each other according to prescribed patterns to achieve their goals. Two interaction dependencies into opposite directions between a service requester and service provider define a typical action workflow loop. Interaction flows of a conference management system are represented in figure 3.

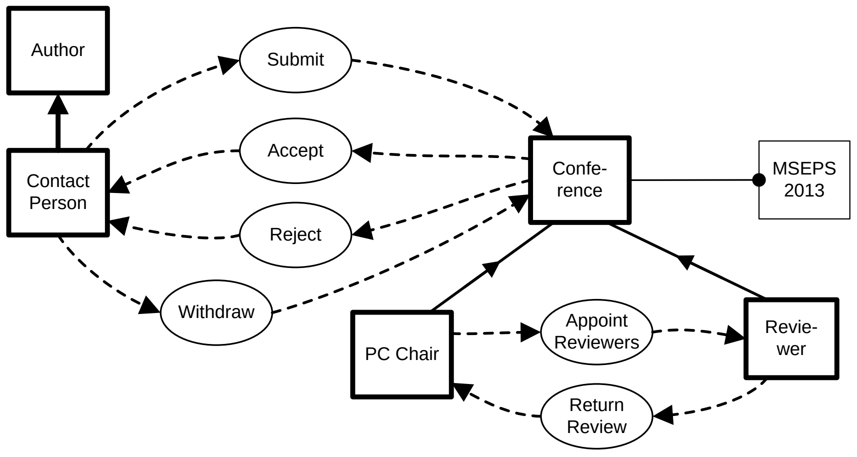

Fig. 3. Example of semantic dependencies between actors

Identification of interaction flows and static dependencies among actors is the first step in the modelling process. An interaction link between two actors indicates that one actor depends on another actor by a specific action. It represents an intersubjective perspective of interaction. For instance, a contact person submits a paper to the conference. He wants this paper to be published. It can be done by accept action, which is initiated by the PC chair. The conference has quite different goals. For instance, conference goals would be to make the submission and reviewing processes as smooth as possible and to be the best conference, i.e. to accept just the best papers.

There are two actors involved in this business process, a contact person who will submit a paper and a conference. A conference is composed of PC Chairs and Reviewers. A conference management system has delegated all major communication through a PC chair, which is a part of a Conference (see composition dependency). PC Chair has a goal to handle reviewing process as good as possible that is to do everything in time. He appoints reviewers and sends the review results to a contact person. The task of reviewers is to reviews the papers. A contact person who is also an author (see inheritance dependency) submits the paper to the conference, by triggering the action Submit. When the person submits the paper, the conference has two possibilities the paper will be accepted, or rejected (see actions Accept and Reject). A Contact Person has also possibility to withdraw the paper (see action Withdraw). When the conference receives the submitted paper, a PC Chair chooses reviewers and sends the paper for review. After reviewing process, a Reviewer returns review to PC Chair (see action Return Review). The results of the review will be sent to contact person (acceptance information or rejection).

4 Modeling Process

The main contribution of this paper is to present the modelling process that consists of five fundamental steps, which support the incremental and systematic service-oriented analysis and design process. An integrated modelling process provides the guidelines for the transition between levels (see figure 1). The requirements traceability is critical for change management, verification and validation processes. A starting point of service-oriented analysis is identification of actor goals and interaction dependencies among service requesters and service providers. The structural aspects of a system are used to represent business data. The behavioural aspects are clarified by defining object transition effects. Without ability to represent noteworthy structural changes, it would be difficult to understand the deep semantics of interactions. Having possibility to reiterate these modelling steps, helps to keep data minimal. The steps of this process are as follows:

1. Identification of the main scenario.

Suppose the conference needs to accept just the best papers and a contact person hopes that his paper to be among those accepted papers. This scenario represents a normal flow of events (Cockburn, 2001). It can be expressed by using two interaction loops, which are represented in figure 3. The first service loop related to paper submission and acceptance. It can be represented as follows:

if Submit(Contact Person ····▶ PC Chair)

then Accept(PC Chair ····▶ Contact Person).

The second interaction loop deals with the actions of appointing reviewers and retuning reviews. It is as follows:

if Appoint Reviewers(PC Chair ····▶ Reviewer)

then Return Review(Reviewer ····▶ PC Chair).

The interaction flows among actors are graphically illustrated in figure 4.

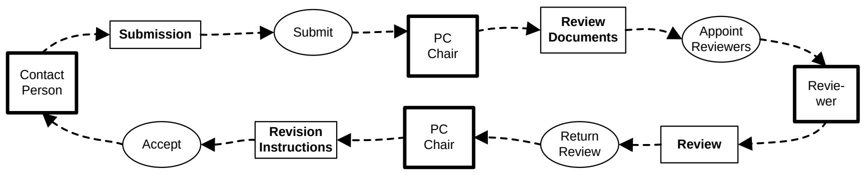

Fig. 4. Main interactions in a conference management system

Interaction flows are the special types of concepts that represent moving flows. In service-oriented modelling method, solid rectangles are used for the denotation of material flows and light boxes show information flows. An action with a missing data or material flow is understood as a decision or control flow. Actions are performed by actors and are represented by ellipses. They are necessary for transferring flows between subsystems, which are represented by various organizational components. Actors are denoted by square rectangles.

2. Definition of actions in terms of transition dependencies.

The internal effects of objects can be expressed by using transition links (——▶) between various classes of objects. There are three fundamental ways for representing object behaviour by using reclassification, creation and termination actions (Gustas and Gustiene, 2012). If termination and creation action is performed at the same time, then it is called a reclassification action. The graphical examples of creation, termination and reclassification are presented in figure 5.

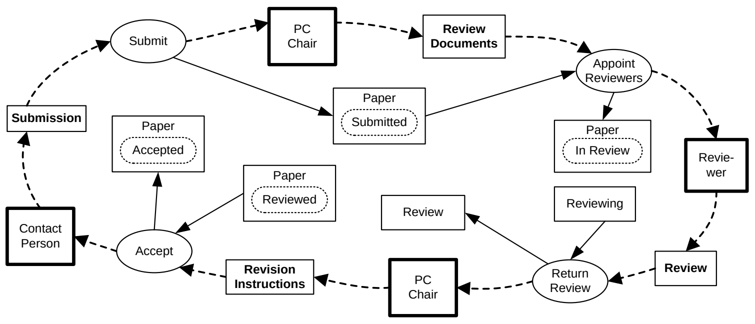

Fig. 5. Main reclassifications in a conference management system

A contact person has a possibility to *submit* a paper. The submission is performed when the Paper[Submitted] object is created. When it is accepted, the responsibility of the conference PC chair is to trigger the *appoint reviewers* action. It is used to send review documents to the reviewers and reclassify Paper[Submitted] to Paper[In review]. Reviewer is obliged to deliver review to PC chair by triggering the *return review* action, which terminates the Reviewing process and created a finalized Review. The PC Chair is authorized to *accept* a reviewed paper by informing a contact person with revision instructions. A Paper[Reviewed] is reclassified to Paper[Accepted] by the Accept action.

3. Identification of a noteworthy semantic difference in every action.

This step is important for identification of attributes, which are affected during object transitions from the pre-condition to post-condition classes. The semantic difference must be defined for every transition dependency by using mandatory attribute links. Various types of attribute dependencies in a conference management system are represented in figure 6. For example, the Accept action changes the state of a Paper object from Reviewed to Accepted. Note that each Paper[Accepted] must be characterized by five properties (Presentation Time, Review, List of Authors, Submission number, Contact Person) and Paper[Reviewed] - by four properties. The noteworthy semantic difference is represented by the complementary attribute Presentation Time.

4. Refactoring.

It is difficult to get initial diagram without inconsistencies and redundancies from the start. Classes and their attributes must be revisited and their semantics examined several times. The refactoring step is necessary to keep conceptual models clean from inconsistent attributes as well as minimize diagrams as much as possible. The refactoring process (Fowler, 1999) does not alter semantics of specification. Refactoring is an essential characteristic of good engineering, because it makes necessary structural changes in order to make modelling clean and understandable. The diagram, which illustrates the outcome of this step, is presented in figure 6.

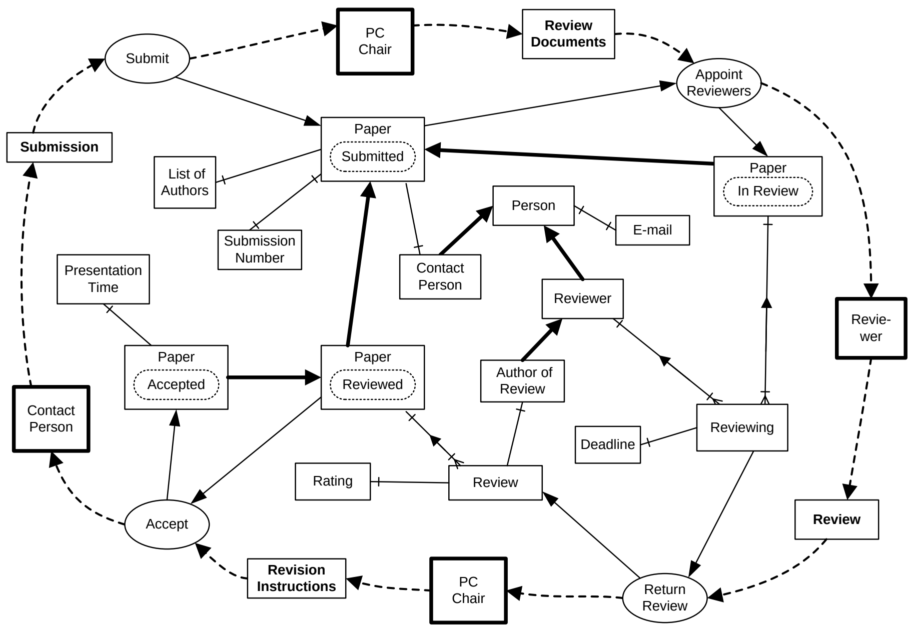

Fig. 6. Interactive, behavioural and structural aspects of a conference management system

Inheritance mechanism allows sharing attributes via generalization/specialization relations. So, inheritance hierarchies can be used to reduce the diagram. For instance, Accepted Paper class in this diagram inherits attributes from Paper in state Reviewed.

5. Describing the alternative scenario.

This step is important, because the modelling process should provide with the possibility to demonstrate available alternatives to the main course of events. Note that the Reject event is an alternative to Accept. Therefore, it should be added at this step (see figure 3).

We have introduced an incremental way of modelling enforcing only minimal data sets, which are represented for various types of actions. Five steps of analysis process are also important for integrity control between static and dynamic aspects in a system. The data necessary in this business process are limited just to most relevant, adequate and not excessive. But the process of data minimisation alone will not solve the problem. Creation of personal data records must be adequate with respect to basic activities in a transaction. It means that analysis of data should be done taking into account the context of business process as well as data handling policy that the service requester and service provider establish in an agreement. How to find out which data are relevant to specific business process depends much not only on structural analysis, but on interactions and behavioural aspects of a system.

Data life cycles vary in different scenarios. Data life cycle analysis is important to understand and to justify *why* and *when* personal data can be stored in the system and *when* it should be deleted. Creation of personal data records and keeping them in identifiable form not longer than necessary is important principle of privacy enhancement technologies. To manage data life cycle implies analysis of different aspects of data, *what* data is processed, by *whom*, *why*, *where*, and *how*. To get answers to all these questions a holistic integrated representation of a system should be analysed.

5 Conclusion

Problems in the area of personal data processing such as lack of data minimization and data life cycle management require the new research solutions. Privacy issues are always embedded into some organization and are related to business scenarios. It means that these scenarios should be analysed and designed together with functional requirements. As data is the main element concerning the management of information privacy, it is critical to have a way of data analysis for the specific context of business scenario. The advantage of this conceptual modelling method is that it facilitates reasoning about semantic integrity of data. It provides a modelling process and modelling techniques for early requirements analysis, where pragmatic and semantic aspects of different scenarios can be analysed together.

The modelling process supports the traceability between different levels of architectural framework, which is critical for understanding how and why technical system components are useful and how they fit into overall organizational system. Analysis using this method can be applied to solve the problems of data minimization. Applying a method for analysis and design of privacy concerns would contribute with a new knowledge in designing security assurance and privacy-enhancing mechanisms. The method could be applied for diagnosing the *redundant data,* i.e. to distinguish between object properties, which are relevant or not justified with respect to scenarios in the secondary interaction loops. It can be also used to detect the *temporal data* or to distinguish between object properties that can be accessible not longer than necessary. The method is able to justify the *persistent data*, which must be to retained in relation to data transfer scenarios and policies, if various commitments are broken.

References

1. Cockburn, A.: *Writing Effective Use Cases*. Boston: Addison- Wesley (2001)
2. Dietz J. L. G.: Enterprise Ontology: Theory and Methodology, Springer, Berlin (2006)
3. Dietz J. L. G.: DEMO: Towards a Discipline of Organisational Engineering. *European Journal of Operational Research.* 128(2), 351-363 (2001)
4. Falkenberg, E. D., Hesse, W., Lingreen, P., Nilsson, B. E., Oei, J.L.H., Rolland, C., et al.: *A Framework of Information System Concepts* (Report of the IFIP WP 8.1 Task Group Frisco). Leiden; Leiden University (1996)
5. Ferrario, R.. & Guarino, N.: Towards an Ontological Foundation for Service Science. In *Future Internet – FIS2008: The First Internet Symposium, FIS 2008 Vienna, Austria. Revised Selected Papers*, pp. 152-169. Berlin: Springer (2008)

6. Fisher-Hübner, S. & Hedbom, H.: Benefits of privacy-enhancing identity management. Asia-Pacific Business Review, IV (4), 36-52 (2008)
7. Fowler, M.: *Analysis Patterns: Reusable Object Models.* Menlo Park: Addison-Wesley (1997)
8. Gustas, R.: Modeling Approach for Integration and Evolution of Information System Conceptualizations, *International Journal of Information System Modeling and Design*, Vol. 1, Issue No 1, pp.79-108 (2011)
9. Gustas, R. and Gustiene, P.: Conceptual Modeling Method for Separation of Concerns and Integration of Structure and Behavior, *International Journal of Information System Modeling and Design*, Vol. 3, Issue No 1, pp.48-77 (2012)
10. Gustas, R. & Gustiene, P.: A New Method for Conceptual Modelling of Information Systems. *Information Systems Development: Towards a Service Provision Society. Proceedings of the 17th International Conference on Information System Development*, pp.157-165. New York: Springer (2009)
11. Gustas, R. & Gustiene P.: "Pragmatic – Driven Approach for Service-Oriented Analysis and Design", *Information Systems Engineering - from Data Analysis to Process Networks*, IGI Global, USA (2008)
12. Gustiene, P.: Development of a New Service-Oriented Modelling Method for Information Systems Analysis and Design. Doctoral Thesis, Karlstad: Karlstad University Studies, 2010:19 (2010)
13. Maciaszek, L. A.: *Requirements Analysis and System Design*, Addison Wesley (2005).
14. Schütz, P. & Friedewald, M.: Privacy: What Are We Actually Talking About? A Multidisciplinary Approach. In S. Fischer-Hübner, P. Duquenoy, M. Hansen, R. Leenes & Ge Zhang (Eds.) *Privacy and Identity Management for Life.* IFIP AICT 352, pp. 1-14, New York: Springer (2011)
15. OMG.: Model Driven Architecture [Electronic Version]. Retrieved October, 2012, from http://www.omg.org/mda (2010)
16. Zachman, J. A.: A Framework for Information Systems Architecture. *IBM System Journal, 26(3)*, 276-292 (1987)
17. Westin, A.F.: *Privacy and Freedom.* Atheneum, New York, NY, USA (1967)