Integrated System Analysis Environment for the Continuous Consistency and Completeness Checking

Erki EESSAAR
Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn, ESTONIA
eessaar@staff.ttu.ee

Abstract. Modelers create models of a system during information system development. Models should be correct, complete and consistent in order to be most useful. Modeler must have possibility to use software that helps to automate these kinds of checks in order to raise the speed of work and quality of its results. This article describes the software system that allows to perform strategic- and detailed analysis of a database-centric information system without using complicated visual notations. System supports methodological framework for the Enterprise Information System (EIS) strategic analysis (like the one that is described by Roost et al. [1]). User of the proposed system records specification in a database as a single integrated model. We propose database queries that help to find consistency and completeness (CC) problems of the recorded models.

Keywords. Model multiplicity, metamodeling, consistency, completeness, CASE

Introduction

Successful development of an information system takes a lot of effort. Important part of this work is modeling of the system. Common approach is to create different types of models that describe different aspects (dimensions) of the system with the different level of abstraction. An example is the Zachman framework for the information systems architecture [2]. Each model type is used in order to describe one aspect of a system. For example, system is described in terms of different views in case of visual modeling language UML. Each view has one or more corresponding diagram types. UML version 1.5 specifies nine types of diagrams [3] and UML 2.0 specifies thirteen types of diagrams [4]. In the different projects only subsets of these diagram types are used, depending on the goals. But more than one type of diagrams is needed in order to describe the static structure as well as the behaviour of the system. For example, Larman [5] presents a possible structure of a system analysis specification. The specification must contain UML diagrams (visual models) as well as textual models:

- Use case model (diagrams and textual specification of use-cases).
- Domain model (diagrams and textual specification of conceptual classes and attributes).
- System sequence diagram.
- Contracts of the system operations.
These models together also constitute a model. Final versions of the models that describe a system have to be syntactically and semantically correct, complete and consistent within itself and with each other in order to be most useful. If model is the combination of diagrams and textual description, then there can be inconsistencies between these components.

Examples of the inconsistencies within one model:
- Use cases/actors have different names in a diagram and in a text.
- There are different amount of use cases/actors in a diagram and in a text.
- A use case is associated with the different actors in a diagram and in a text.

Examples of the inconsistencies across different models:
- Names of the actors are different in a use case model and in the sequence diagrams that describe system operations.
- Names of the elements of a domain model differ from the names that are used in the pre- or post conditions of the contracts of the system operations.

Examples of the completeness problems:
- A domain model is missing.
- A use case diagram is not accompanied with a textual specification.
- An operation contract doesn't have the post-conditions.

Checking of the models in order to find such problems can be at least partially automated by the software system. Models could be inconsistent and incomplete during the development process. But it should be possible at any time to get information about these problems. It helps to gradually improve the quality of the models. Pedagogical pattern Built in Failure [6] suggests that teacher should remove the fear of failure as a barrier to learning by making failure a part of the learning process. A hypothesis that must be controlled in the future is that the checking functionality would also change a modeling tool to a valuable learning tool. It is because continuous feedback from the system instead of a teacher reduces fear to make mistakes that hampers the learning process. It also eases the work of a teacher who doesn't have to check consistency and completeness (CC) problems manually.

But CASE systems today do not provide enough support for checking consistency between different types of models [7], [8] and between evolving versions of models [9]. Instead it is mostly the manual work and takes quite a long time and a lot of effort. And even if CASE tools provide some support it is still a "daunting tasks beyond anyone's cognitive ability" [10] because of big amount of different types of models.

An author of this article teaches database design in a university. A part of the course work is a term project. Students have to create a strategic- and detailed analysis of an information system and a prototype of its software. Current structure of the project documentation has been used during the last four years. For example, 75 projects where presented in the spring of 2005. Teacher reviewed projects together with their authors and pointed to the mistakes. Average length of a review of one project was about 20 minutes. Average interval between first checking of a project by the teacher and acceptance of the project was 4.2 days. Students improved their projects during this period and sometimes they did it repeatedly. Students used word processor in order to write textual models and CASE-tools or diagram editors in order to draw visual models. These systems don't support automatic consistency and completeness checks. Presented projects reflected this situation and contained many such deficiencies. Another problem is that students have difficulties to understand how all
these different models are connected with each other. Therefore the system is needed that gives fast and precise feedback to the students. 

Delen et al. [8] write: “Second, there is a need for a completely Web-based integrated modeling environment for distributed collaborative users.” Our main contribution is description of the system that allows to perform strategic- and detailed analysis of a database-centric information system without using complicated visual notations. This system should ease creation of an independent work by the students and correction of it by the teachers. It can also be used in the real-world information system development projects. System should support methodological framework for the Enterprise Information System (EIS) strategic analysis (for example, framework that is described by Roost et al. [1]). Person with the modeler role will record system specification in a database as an integrated model using one tool. Modeler doesn't have to create a collection of weakly connected models by using different tools any more. Modeler will use the modeling language that is simplified synthesis of the different system specification languages (UML [3], SMX [11] and OPM [10] among others). The structure of the database will be derived from the fixed metamodel of this language. We plan to use DBMS that allows to use SQL language. Because of the fixed metamodel this system will be more similar to a CASE than to a Meta-CASE tool. Data about the various aspects of a system will be recorded in a database using a form-based web interface. We propose queries that find consistency or completeness (CC) problems that are present in a specification. Our approach eliminates problems with the inconsistencies between diagrammatic and textual representations. They are kind of views to the information in the database that could be generated by the system at any time. We don't have yet completely implemented the system but have started to create its prototype. We see that such a system will help students/teachers in the learning/teaching process and therefore continue its development. It will also help to improve quality of the result of the real information system development projects.

The rest of the paper is organized as follows. Section 1 gives an overview of the existing work about checking consistency and completeness of the models. Section 2 presents the metamodel of the modeling language that our system will use. Section 3 lists queries that help to identify consistency and completeness problems of the model. Section 4 contains discussion and comparison of our work with the existing work in the field. Section 5 summarizes this article.

1. Related Works

UML is nowadays de facto standard for describing the information systems. Problems of consistency between UML diagrams have for example acknowledged by Engels and Groewegen [12] who describe open issues in the object-oriented development. Modeler has to learn a lot of different notations, rules and guidelines that are used in the different types of models. McLeod [13] notes that UML 1.1 contains 233 discrete concepts. Still he proposes rich visual notation for process models which could replace UML dynamic diagrams [13]. If we study one UML or other similar model, then we have to constantly look the other models in order to fully understand it. Switching between different pages/ files/ packages is inconvenient as well as mentally challenging and wearying. "Multiplicity of representational styles impedes communication between modeling professionals and their clients." [14] Visual models are usually created with the different CASE or model drawing tools (Rational Rose, ERWin, ArgoUML, Visio,
Dia etc.) and are accompanied with the textual specifications that are created using a text editor. We have to have this software in our computer if we want to thoroughly study these models or modify them. Modification of one model requires modifications of dependent models as well. Multiplicity of modeling software and files that contain models causes often creation of the model from scratch instead of reusing the existing ones. Some approaches that are used in order to achieve correct and consistent models:

1. Formulation of the guiding rules that a modeler who creates different types of models should follow. For example, Glinz [15] describes rules that help to minimize inconsistencies between a class model and a use case model.
2. Usage of the cross references between the different types of models. For example, Glinz [15] proposes to use references to a class model in the scenarios of use cases.
3. Usage of the specific models which contain cross references between other models. Example of such a model is a CRUD matrix. It shows associations between object types and processes [16] and helps to check their consistency.
4. Usage of the systems that evaluate models created by the CASE tools or assist user of a CASE tool.
5. Usage of the modeling notations and systems that use one type of model in order to specify multiple aspects of a system.

Drawback of the approaches 2 and 3 is that without a tool support, references in a model or new kinds of models may themselves have CC problems. CASE systems can have supporting tools that check the models or provide active assistance to its users (approach 4). They may transform the diagrams into some other form of representation in order to analyze them. For example, generated description logics statements [9] are analysed by using description logic query tool. Richters and Gogolla [7] describe the system which translates UML models to the statements of UML-based Specification Environment language. Models are then analyzed by simulating the case when model elements have instances. Agents based system WayPointer [17] monitors the use case model, which is created by some CASE tool, for completeness, consistency and correctness. It can point to the problems and offer recommendations.

Agarwal and Sinha [18] conclude that developers don't rate any of the UML diagrams as very high in terms of usability. It might be caused by the usage of the several model types which leads to the inconsistencies between various parts of system specification [10], [19]. Acknowledgement of the "model multiplicity problem" is not new. A single model-based approach is superior to the multiple model approaches for late requirements engineering through implementation according to Paige and Ostroff [20]. Already Jäderlund [11] describes a methodology for the holistic system development that uses so-called system matrices in order to describe the system. System matrix (SMX) incorporates multiple views of the system. It provides methods for checking correctness, completeness, and consistency (CCC check) of the system.

More recently model multiplicity problem has been addressed by introducing Object-Process Methodology (OPM) [10], [19], which is a holistic system modeling, development and evolution approach. OPM uses Object-Process Diagrams (OP diagrams) for the graphic specification and Object-Process Language (OPL) for the textual specification of the system. OPM uses one integrated type of model in order to describe structural, functional and behavioural aspects of the system [10]. CASE tool Object-Process Case Tool (OPCAT) that supports OPM has been developed [19].

Yet another example of the modeling languages that corresponds to the single-model principle is Eiffel [20]. Delen et al. [8] propose system Modelmosaic that allows
to create different types of models. It records models and relationships between the elements of different types of models in a single integrated information base. These relationships, that are recorded as business rules, allow to generate new models from the existing ones.

2. Description of the Modeling Language

A model is created by using some language. Specification of the semi-formal language should contain descriptions of the abstract syntax, well-formedness rules and semantics [21]. For example, metamodel that describes the abstract syntax of UML is presented as a set of class diagrams [3], [4]. Well-formedness rules of UML are expressed using OCL constraints and its semantics are described using free-form text. In this section we present the metamodel of the language that will be used for specifying information systems in our proposed system. Diagrams that present fragments of the metamodel are accompanied with the free-form textual descriptions that explain some of the underlying concepts. Structure of the database for recording specifications of the information system will be derived from this metamodel. In the section 3 we present queries that help to check the well-formedness of the recorded models.

Interested parties can participate in the information system development projects in the different roles (see Figure 1).

An information system (IS) is described using three types of subsystems according to the methodological framework for the Enterprise Information System (EIS) strategic analysis [1]. These types are: areas of competence, functional subsystems and data centric subsystems that are also called registers (see Figure 2). A functional subsystem corresponds to one or more business processes [1]. "A register is a logical data-centric view of a business object that holds the state and transactions data of the object and provides related recording and query services." [1] Administrative subsystems help to perform administrative tasks of the organizations. Examples are subsystems for the management of data about the workers and documents. These kinds of subsystems are part of many different information systems. Business subsystems help to perform specific business tasks of the organizations. These tasks are reason why this organization is founded in the first place. For example, university IS has subsystems for the management of data about the students, curriculums and study results.

Functional subsystems use the services of one or more registers by reading and modifying data in them. Subjects who have some role in an IS use the services of one or more functional subsystems that belong to the area of competence of their role [1].

Figure 1. Metamodel of the projects and participants
We provide possibility to specify non-functional requirements of the system (see Figure 2) by using structure described in Volere Requirements Model [22]. Functional requirements of the information system can be specified as use cases. Corresponding fragment of the metamodel (see Figure 4) is created based on the guidelines of Larman [5] and Cockburn [23]. Each use case belongs to some functional subsystem (see Figure 4).

![Metamodel of the subsystems](image2)

![Metamodel of the data elements](image3)
Each use case describes scenarios that consist of actions. Most of actions are performed sequentially. Some actions can be performed at any point of the scenario. Use cases can be related by using either extension or inclusion relationships.

Actions may be performed by the actors who are either agents or instruments. Agent corresponds to the area of competence. Registers that are used by a functional subsystem are specified in terms of the data elements (see Figure 3) and contracts of the database operations (see Figure 5). Action that is part of the use case can cause execution of the database operation. Result of the operation is described in terms of the post-conditions. Use case is triggered by an event (see Figure 4). Events cause change of the state of the objects (see Figure 6).
### 3. Consistency and Completeness (CC) Checks

Structure of the database determines the elements that can be associated. It also enforces the constraints of the metamodel according to which participation and/or cardinality is 1 at some relationship end. General principle of our system is that consistency and completeness of the models will not be ensured by the database constraints. It gives more freedom to the modeler. Occurrences of each problem will be found by using a query or a set of queries. Next we present some CC checks that the system must be able to perform. A query must find the cases where the CC rule is not fulfilled. Another possibility is to create a query that simply returns the value false if a model contains a problem. It must be possible to use these queries at any moment.

1. An IS consists of at least one area of competence (AC), functional subsystem (FS) and registry subsystems (RS).
2. An AC has exactly one corresponding actor.
3. An actor (and therefore an AC) uses services of at least one FS.
4. Services of a FS are used by at least one actor.
5. An IS (through some of its FS) is used by at least one non-adjacent actor.
6. A FS uses services of at least one RS.
7. A RS has at least one FS that reads its data.
8. A RS has at least one FS that adds new data to it.
9. An IS has at least one administrative FS and RS and business FS and RS.
10. An IS (as a whole or through some of its FS) has at least one non-functional requirement from each of the different requirements types.
11. Functional requirements to a FS are described by using at least one use case.
12. A use case describes process what has between $n$ and $m$ actions (steps). Numbers $n$ and $m$ should be dynamically changeable.
13. A use case is associated with the description about the interest of a primary actor in the context of the use case.
14. A use case has ideally zero unsolved open issues.
15. A use case that is not an essential use case is associated with at least one database operation through some action.
16. A data element is created and read by at least one database operation (association through a post-condition).
17. A database operation has at least one post-condition.
18. A database operation is associated with at least one action that is part of the non-essential use case. Each database operation is used by at least one use case.
19. Associations between FSs and RSs are consistent with the association of the use cases (that belong to some FS) and data objects (that belong to some RS).
20. It is possible to get from the start state of an object to any other state of an object by using state transitions.
21. An object type has exactly one start state and one or more end-states.
22. Relationships have mark about aggregation/composition/generalization at most at one end. In this case other end has no such marks.

Queries can also be used in order to find suspicious parts of the specification that may or may not be the mistakes. For example, query can search registers and data elements that are not subject of the update or delete operations. Queries help to collect metrics about the IS (like proposed by [24]). For example, we can count amount of use cases in different FS-s. If some FS has a big amount of use cases compared to others then it shows that the FSs are unbalanced and some big FS needs to be further divided.
4. Discussion and Comparisons

Commercial tools Modelmosaic [8] and EA WebModeler record models in a database. EA WebModeler [25] provides form-based web interface for creating models. Our system is not unique in this regard. UML is widely used notation and therefore it is reasonable to teach it even after we start to use this system. For the teaching and presentation purposes it is sometimes useful to see system specifications in the form of UML diagrams. Possible solution is the program feature which generates XMI files that contain models, based on the data in the database. These files can be opened using a CASE tool. Similar feature exists in EA WebModeler [25].

A modelling language that follows the principle of the single model must satisfy following three criteria: conceptual integrity, consistency of views, wide spectrum applicability [20]. Our proposed solution satisfies "conceptual integrity" criteria because models are recorded in one logical database. Each model element is recorded only once. Our proposed solution satisfies "consistency of views" criteria because checking of the consistency of different views of a model is automated. Our proposed solution partly satisfies "wide-spectrum applicability" criteria. This system is used in order to perform strategic and detailed analysis of the system but not design or implementation. In the next iterations system could be extended in order to allow generation of the stored procedures based on the database operations and table specifications based on the data-elements.

Why can't we use existing free software in order to model systems by using single model type? Examples of such systems are SystemSpecifier [26] that allows to create system matrices and OPCAT [19] that allows to create OPM models. First reason is that they don't fully support the methodological framework for the Enterprise Information System (EIS) strategic analysis. They don't allow to specify different types of subsystems and their interconnections. In addition, proposed system is different from OPCAT and SystemSpecifier because models are recorded in a database but not directly in the files. It helps to avoid well known problems of the file based systems like separation-, isolation- and duplication of data. More than one modeler can work with the same model at a time. Problem of OPCAT is that it doesn't provide explicitly CCC check functionality. Our proposed system allows to use queries from the database in order to find consistency and completeness problems. Usage of the integrated model prevents repeating recording of the same information and thus helps to avoid inconsistencies. Queries can be used in order to classify elements of a model. For example, it is possible to determine whether a use case is concrete, abstract, base or addition [5, p. 388] by making query about its relationships with the other use cases. Another example is that we don't have to separately record the events that influence the registers. Instead query can find the events that trigger the use cases which steps use the services of the register. Our proposed system can also be used in order to collect information about the work amount and performance of the modelers. It would also be a useful e-learning tool, because it is planned to be a web-based system.

5. Conclusions

In this article we have described principles of the system that helps to perform strategic and detailed analysis of the information systems. This system allows to record one integrated model of the system into a database by using form-based web interface. It
provides queries for finding consistency and completeness (CC) problems of the model. These queries can be used at any moment during the modeling process. We have created a partial prototype of the system for managing subsystems (see Figure 2). This prototype also allows to manage and execute queries (see Section 3).

If the system will be ready, then we can perform the usability study in order to evaluate which way users prefer to describe the system – using visual diagrams with little support to CC check or using textual descriptions with the extensive CC checks.

References