An Approach to Information Systems Modelling Based on Systematic Complexity Reduction

A.H. Seltveit
Faculty of Electrical Engineering and Computer Science
The Norwegian Institute of Technology
N-7034 Trondheim

Abstract

An approach to information systems development based on systematic complexity reduction is presented. It facilitates integrated specifications and at the same time it provides facilities that allow the developers to split a complicated task into a set of more comprehensible subtasks. Through the introduction of viewspecs the suggested approach recognises the need to adopt multiple perspectives when developing large and complex information systems. The possibility of using predefined filters, user-defined filters and combinations of these, provides the developer using the complexity reduction approach with a great deal of freedom in focusing on different aspects of a system. The viewspecs are used 'actively' during the development in the sense that they do not only serve as passive components or projections of a set of specifications (e.g., for presentation purposes) but also serve as a basis for entering new information into specifications.

1 Introduction

The rapidly changing business environment now found nationally and internationally and the hard competition to survive, have made the information systems crucial to the success of most companies. The right information, in the right form, and at the right time is crucial to staying in business. The increasing complexity and evolving nature of organisations mean that systems development approaches must cope with a highly complex and changing reality. Although substantial progress has been made with respects to technology, systems development still suffers from low productivity, high development and maintenance costs and delays in delivering on time, e.g., [5].

Contemporary approaches provide a wide range of support to tame the 'wickedness' of information systems development, each emphasising different problem areas or development phases. Some areas of particular relevance are:

- CASE tools [8, 17]: the method and tool aspects are emphasised.
- Requirement Engineering/Conceptual modelling approaches [15, 21]: the modelling aspect is emphasised.
- Business Process Reengineering approaches [7]: organisational aspects are emphasised.

The trend has been to automate or support more development activities and address the problems at earlier phases. The later a mistake or omission is discovered the more severe are its effects. Thus, much effort is being put into developing more comprehensive CASE tools covering organisational, modelling, method and tool aspects. The TEMPORA projects which forms the basis for the work reported here belong to such efforts.

A key point in providing support for development of wicked systems is to tame the problem as much as possible by gaining insight about the problem and possible solutions. Thus, to deal with the 'wickedness' requires that the actors (e.g., developers and end-users) involved in the development process, must have sufficient understanding of those parts of the world that may have an impact on the target information system as well as on its environment. Two issues arise:

- How is this understanding achieved?
- How can the actors' understanding of the world be reflected in a computer system?

A hypothesis is that people understand the world by building mental models, e.g., [22]. These models are simplifications or abstractions of the real world. Whether the human brain actually deals with information in terms of mental models is not important for
our work. The important issues are that it is impossible for the human brain to perceive all the details of the world, and that the concepts of model and abstraction have been shown to be useful in order to gain understanding about the world. The actors involved in the development process use models as a basis for communication to gain insight into the problem domain. The models are elaborated until they contain sufficient details to provide the basis for coding the software. We will later refer to this as the dilemma of the duality of specifications.

In Information Systems Engineering, the concept of abstraction is crucial. It constitutes the major complexity reducing technique ever known in computing. In the widest sense, abstraction is underlying all modelling. We cannot put the reality as such into the computer, it must be simplified — we only model the major features. Moreover, we cannot consider all details at one time but concentrate on subsets that includes the relevant details that are necessary to carry out a task. In choosing relevant subsets, we apply abstraction more or less consciously. The importance of abstraction in information systems modelling has been pivotal in the development of the complexity reduction approach.

The paper is structured as follows: Section 2 presents the idea of complexity reduction. The complexity reduction approach is described in Sections 3. Sections 4 outlines related work and Section 5 gives some concluding remarks.

2 Complexity Reduction

2.1 The Duality of Specifications

Before we elaborate on the concepts of complexity and complexity reduction, we need to elaborate on the dilemma of the duality of the specifications. Specifications of information systems play dual roles in the systems development process: A specification serves as a basis for communication and understanding, and, for coding of software. This duality of specifications is illustrated in Figure 1.

Figure 1: Duality of systems specifications.

Specifications must contain enough details and they must be understandable for the actors in the development process. For simple (or tame) [16] problems, this duality does not cause any severe problems because existing modelling approaches provide sufficient mechanisms to deal with the limited amount of detail included in such specifications. This is in contrast to large and complex systems where the amount of detail is overwhelming and it is impossible for one person to have a complete understanding of the entire system. Hence, the dilemma of the duality of specifications contributes substantially to the wickedness of information systems development. Furthermore, very little research is done with respect to how to fulfill the dual roles of specifications in the modelling process [18].

2.2 The Idea of Complexity Reduction

The concepts of abstraction and model (specification) are central for dealing with the 'wickedness' of information systems development. The increasing complexity and evolving nature of organisations and thus, information development will not reduce this wickedness, rather the opposite. The trend is towards larger systems and therefore, it is essential that systems development environments address how the dual role of specifications can be balanced under such conditions. We want to allow all relevant details to be included in a specification and at the same time make it comprehensible for the actors that participate in the modelling process. A major challenge is to overcome the problems of

- Fragmentation of specifications. Many modelling approaches and several specification languages are used in various development phases, from requirements development to coding. The state-of-the-art is that models and languages are not integrated over phase-boundaries. There is a need for consolidation of the various approaches.

- Complexity explosion. As more details are added during the development process, a complexity 'explosion' may occur. Contemporary modelling approaches suffer from the lack of support to deal with this information overload. There is a need for a systematical way to deal with large number of details in specifications, as well as during the modelling process.
2.3 Specification Complexity

Before we elaborate on the idea of complexity reduction, we need to define what we mean by *specification complexity*. According to Webster’s II Dictionary, ‘complex’ can be defined as [9]: 1) Consisting of composite parts, 2) Intricate, and 3) A whole composed of interconnected or intricate parts. Thus, to say that something is complex means that it consists of composite parts that may be interconnected in some way. If the number of parts and interconnections are low, a person usually perceives a specification as having low complexity (Figure 2a). As the number of parts and interconnections increases, the complexity of the specification also increases (Figures 2b and 2c). At some point the number of parts and interconnection reaches a level where it becomes difficult to grasp the meaning of the specification (Figure 2c).

How specification complexity can be measured by means of metrics, e.g., [14]) is not a topic of this paper.

2.4 Complexity Reduction in Modelling

Complexity reduction can be defined as:

A process of producing *comprehensible* views of a specification in such a way that they include the necessary details in a suitable form and in such a way that they can be used *actively* during the development process.

Here ‘details’ refer to parts and interconnections contained in a specification. A ‘view’ means an abstraction of a specification. Thus, the concept of abstraction forms the basis of the complexity reduction approach. ‘Comprehensible’ means that the view should be appropriate as a basis for communication and understanding among the actors involved in the systems development process. By using views ‘actively’ during the development, we mean that they should not only serve as passive components or projections of a specification or set of specifications (e.g., for presentation purposes) but also serve as basis for entering new information into specifications. An *approach* based on complexity reduction provides support for dealing with complexity in specifications along with an appropriate modelling process. Thus, such an approach comprises a set of languages, methods, and tools. Figure 3 illustrates the idea of complexity reduction in information systems modelling.

3 The Complexity Reduction Approach

The approach based on systematic complexity reduction is divided into four major activities as shown in Figure 3 (the activities are not necessarily sequential):

- **Filtering** is the process of generating views (simplifications) from a specification, i.e., creating appropriate filters.
- **Presentation** is the process of using the views that result from the filtering in a way that facilitates communication and understanding.

---

F. P. Brooks [2] refers to this as the essentials and accidents of systems development.
3.2 Filtering

To define a filter implies that the user must determine the right abstraction level for the information in a specification which is necessary to carry out a task. Thus, the filtering process consists of two subactivities [19]:

1. Determine which specifications to focus on.
2. Determine which components of the selected specifications to focus on.

The first step means that the user must locate which specifications should form the basis for the filtering, i.e., one or more full specifications or one or more viewspecs. An example of a PID specification is shown in Figure 4. In the second step, the selected specifications from the first step are used as input to a filter. Figure 5 shows the viewspec that results when a component filter is performed on the PID specification in Figure 4. A viewspec contains the relevant details whilst the irrelevant details are suppressed. To what extent this process can be supported depends on the nature of the filter. In the TEMPORA project [24] we have defined a set of filters for rule based specifications, ERT and PID specifications. The user is however not restricted to use only predefined filters, the approach also offers the user the possibility of using user-defined filters and combinations of pre-defined and user-defined filters.

Information may be expressed at different levels of detail and we will refer to a specific level of detail as an abstraction level. We may classify abstraction levels at least in two dimensions according to [19]: specification level and amount of detail contained in a specification within each specification level. Thus, the notion of abstraction level say something about what we want to focus on in the domain/system and at what level of detail we want to examine the relevant aspects of the domain. In [18] a formal framework that provides means to describe various abstraction levels, viewspecs and filters is described.

3.1 Basic Concepts

A viewspec is a subset of a model, which has been selected by the user. Each viewspec holds a particular perspective of a model and thus, the notion of viewspecs gives us a means to view the model from different perspectives.

A viewspec is represented by a language and is a particular type of specification. It will always be associated with another specification. The specification from which the viewspec is generated will be referred to as the originating specification or just specification if it is obvious from the context. The means to generate a particular viewspec of a specification is called a filter. A specification which is not generated by a filter will be referred to as a full specification. A filter defines an abstraction of a specification. The filter specifies the criteria which should be used to suppress details from the originating specification to produce the desired viewspec.

Information may be expressed at different levels of detail and we will refer to a specific level of detail as an abstraction level. We may classify abstraction levels at least in two dimensions according to [19]: specification level and amount of detail contained in a specification within each specification level. Thus, the notion of abstraction level say something about what we want to focus on in the domain/system and at what level of detail we want to examine the relevant aspects of the domain. In [18] a formal framework that provides means to describe various abstraction levels, viewspecs and filters is described.

Figure 5: Process P4.6 and associated information (PID symbols left out).

2The Process Interaction Diagram (PID) [6] is an extension of standard DFD and describes the dynamic aspects of a system.
The user of a filter may be a human being or another system, e.g., an explanation facility. An explanation generation facility invokes a filter in order to use the resulting viewspec as a part of the generated explanation. Thus, graphical viewspecs are used together with textual explanations to enhance validation. The combination of execution techniques [26], complexity reducing techniques and presentation techniques are elaborated in [27]. We refer the interested reader to [18] for a detailed description of the various predefined filters, and how they can be used together with structuring mechanisms to generate viewspecs across specification boundaries.

3.3 Presentation

Presentation is the process of using the viewspecs in a way that facilitates communication and understanding between developers and end-users as well as among developers and eases the modelling process for a single developer. Although filtering and presentation are intertwined processes we have chosen to deal with them separately. Filtering deals with what details to be included (contents) in a viewspec, whereas presentation deals with how these details are to be presented (layout) and used in different facets of modelling.

Filtering away certain details from a diagrammatic specification simplifies the diagram, and often restructuring of the diagram is desirable, e.g., locations of symbols make the diagram too sparse. Thus, filtering and layout modification are combined because of relaxed space constraints when information is filtered away from a specification. The layout of a specification can be changed by redrawing the components of a diagrammatic specification or by rewriting the components of a non-diagrammatic specification. The purpose is to produce a simpler and more comprehensible arrangement of the details of a specification without changing the semantics. Although we expect the restructuring of diagrams to be done manually, there exists work that may be utilised to provide some support. When components are removed, modification of layout can be supported by utilising algorithms for graph drawing, such as minimising the number of crossing lines, e.g., [23] and eliminating lines, e.g., by adding a new node and collapsing arcs which have the same starting and ending point [13]. There are also other ways of presenting a viewspec such as by highlighting relevant components by using, e.g., colours and different size of symbols, by shading of compo-
Figure 4: PID specification of the acquisition activity of a library system.
nents, by using black boxes and distinguish visibility of layers.

We do not only want to present one viewspec at a time. The TEMPORA languages allow system specification from three different perspectives: data-oriented (the Entity Relationship Time Language (ERT) [12]), process-oriented (the PID [6]) and rule-oriented (the External Rule Language (ERL) [24]). Viewspecs can be generated from all the various specifications. Thus, we need to be able to work with them simultaneously, i.e., we want to allow multiple viewspecs to co-exist.

To achieve this a versioning and configuration management system is needed to keep track of specifications. Viewspecs are related to their originating specification as well as to other viewspecs and this is provided by the versioning system. To allow multiple viewspecs and multiple viewspecs to co-exist, we distinguish between three different relations in the versioning graph: filter, variant and context relations.

A filter relation depicts the relationship between a viewspec and its originating specification. Two specifications are filter related if one specification is generated from the other specification by a filter. Thus, a filter relation depicts relationship between a full specification and a viewspec or between two viewspecs. Moreover, a specification can be filter related to several viewspecs, that is, several viewspecs are generated from the same specification. Figure 6 shows a versioning graph that depicts a situation where a transaction has checked out a specification S1.1 and a set of viewspecs have been generated from it. A specification is uniquely identified by name S1.1{τ}{α},λ, where {τ} indicates it belongs to transaction τ, {α} indicates that it is a viewspec of type α and λ is the revision number local to the transaction.

3.4 Entering New Specification Details

Views pecs can be used as a basis for entering new information into specifications. Revisions of full specifications has a counterpart in updated viewspecs. Figure 7 shows a version graph for the PID specification in Figure 4 with the associated viewspec depicted in Figure 5. If a viewspec is updated this is depicted by a revision relation, as shown in Figure 7. It shows that the PID viewspec in Figure 5 has been updated, and it is the contents of the viewspec that are changed, e.g., by adding or deleting information in the PID viewspec. Each update of a viewspec results in a new revision of the viewspec.

A filter relation depicts the relationship between a viewspec and its originating specification. Two specifications are filter related if one specification is generated from the other specification by a filter. Thus, a filter relation depicts relationship between a full specification and a viewspec or between two viewspecs. Moreover, a specification can be filter related to several viewspecs, that is, several viewspecs are generated from the same specification. Figure 6 shows a versioning graph that depicts a situation where a transaction has checked out a specification S1.1 and a set of viewspecs have been generated from it. A specification is uniquely identified by name S1.1{τ}{α},λ, where {τ} indicates it belongs to transaction τ, {α} indicates that it is a viewspec of type α and λ is the revision number local to the transaction.

Figure 6: Viewspecs are filter related to their originating specifications.

3.5 Inclusion of Changes

The interested reader is referred to [1] for a detailed description of the versioning and configuration management system.
of including the changes contained in the viewspecs into the full specification, is not straightforward. During the modelling process several viewspecs may be updated and several versions of each viewspec may co-exist. Thus, an important step in a modelling process based on viewspecs is to provide support for the process of solving eventual conflicts between updated viewspecs and the originating specification. The result of the specification integration process is a consistent version of the previous full specification including the changes in the selected viewspecs.

Figure 8 shows how specifications that form the basis for a specification integration process is depicted in a versioning graph. The input to the integration process is related with merge relations between the relevant specifications. Possible specifications are: a previous version of the full specification, one or more update viewspecs or one or more viewspecs which are approximations of the corresponding originating specification. Additional information may be provided about each updated viewspec, e.g., type and what changes have been made since it was created. This information is recorded during the filtering and updating processes. The versioning facility provides the information together with the names of the specifications. The specifications are then retrieved from the specification repository.

Figure 8: Building a new revision based on updated viewspecs.

Only viewspecs which contain approximations of the originating specification and updated viewspecs should be considered for integration. Other viewspecs are not relevant because they are projections of the originating specification and thus, the viewspec's information is already included in the full specification. However, not all updated viewspecs are considered for integration. It is a user decision to select what viewspecs that should be considered for integration. For example, if a viewspec is updated and only used for presentation purposes, we do not consider to propagate the changes. We may also have situations where updated viewspecs are used to explore alternative solutions/conflicting viewpoints. Only the set of viewspecs that represent the 'consensus' solution are considered for the integration process. In this case, we envisage that additional facilities are provided to support the process of arriving at a consensus solution, e.g., [11].

4 Related Work

Mechanisms to allow definition of different views are well known means to abstract away irrelevant details both in specification languages for static modelling (e.g., the concept of scenario for defining different views of a domain in the PhM [20]) as well as for database applications (e.g., SQL views [3]). It is however not until recently the viewing feature has been widely adopted by other approaches (beyond static modelling). Viewing is a complexity reducing technique which has gained more popularity in recent years. This has resulted in a number of approaches which are based on views or viewpoints. Our approach allows multiple perspectives to be adopted when developing information systems, and can thus be considered as a viewpoint-oriented approach. We will therefore compare the main characteristics of our approach with respect to the well-known viewpoint-oriented approaches Viewpoint Resolution [11], ViewPoint-Oriented Development [4] and ARIES [10]:

- ARIES [10] is an environment to experiment with support for analysts in modelling target domains and in entering and formalising system requirements. Like our approach ARIES also stresses the importance of supporting different viewpoints. ARIES however, has a single internal representation of its information and provides a set of mapping from/to its internal language to the desired external language used for presentation. Thus, a change in one viewpoint results in a change in the internal representation and consequently, other presentations that use the same information will be updated. This is in contrast to our notion of viewspecs which are only loosely coupled to other specifications. We also allow viewspecs to be updated even if an inconsistency among viewspecs or between the originating specification and the viewspec may result.
• Viewpoint Resolution [11] is developed to assist validation of rule specifications. Rules are grouped together in viewpoints and Viewpoint Resolution provides a way of detect and resolve conflicts between such viewpoints. Thus, it addresses the fourth activity in our complexity reducing approach, namely inclusion of changes which has only been briefly addressed in our work. Viewpoint Resolution is complementary to our work in that it may provide useful input for us in future development of integration of viewspecs that are expressed in ERL.

• ViewPoint-Oriented Development [4] is not tailored to any particular modelling languages but their work include an organisational frame work developed to encompass multiple viewpoints (called ViewPoints) and various kinds of support to develop systems based on such viewpoints. ViewPoint-Oriented Development is also complementary to our approach in that it addresses more the overall organisational aspects of viewpoint oriented development which we only address to a limited degree, e.g., different roles of software development participants. Similarly to Viewpoint Resolution, ViewPoint-Oriented Development also provides support for consistency checking between pairs of ViewPoints. As mentioned above, this is not addressed in our work.

5 Concluding Remarks

This paper has outlined the main activities of the complexity reduction approach: filtering, presentation, entering new specification details and inclusion of changes. The complexity reduction approach can be used for building specifications from scratch, from existing specifications or any combinations of the two. We have experiences using intermediate versions of the approach on examples and in case studies, e.g., see [18, 25]. Basic lessons learned from the experiences using the suggested facilities to complexity reducing features in case studies can be summarised as follows. Firstly, the use of a complexity reducing approach seems promising. Secondly, the generation of viewspecs, its preparation for presentation and the inclusion of changes into full specifications based on viewspecs are tedious tasks and comprehensive tool support is therefore necessary to use viewspecs efficiently and effectively in the modelling process. Although we have done some experiences building specifications based on viewspecs more comprehensive field studies are necessary to test the suggested approach.

Acknowledgements

The work reported has partly taken place in the ESPRIT II Project TEMPORA. The project is funded by the Commission of the European Communities under the ESPRIT R&D programme. The partners in the TEMPORA consortium are: BIM (Belgium), Hitec (Greece), Imperial College (UK), Logic Programming Associates (UK), SINTEF (Norway), SISU (Sweden), University of Liege (Belgium) and UMIST (UK).

References


