Chapter 1

Introduction

In this thesis we present Form-Oriented Analysis, a new analysis technique for a distinct and ubiquitous class of interactive software systems. This class covers well-known form-based applications ranging from typical Internet shops over supply chain management to flight reservation systems. We give a precise definition of the considered class of software systems and have coined the term *submit/response* style applications for this system class. This definition is the necessary basis for our formal treatment of Form-Oriented Analysis. In this thesis we explain how to apply Form-Oriented Analysis and we give formal semantics for its artifacts.

Submit/response style applications are such applications that present to the user a page that offers information as well as a number of interaction options, typically forms. If the user has filled out a form and *submits* the form the system processes the data and generates a *response* page. This response page again offers different interaction options to the user. We model such a submit/response style application in a way that will turn out to be well suited for such systems, namely as a bipartite state machine, which alternates between presenting a page to the user and processing the data submitted by the user. This bipartite state machine is depicted in the key artifact of Form-Oriented Analysis, the form chart. Form-Oriented Analysis describes then, how to annotate this bipartite state machine with constraints, which specify the behavior of the system. The definition submit/response style is not intended to cover all kinds of software systems, but to single out a well-defined and important class of systems. There are of course other interactive software systems that do not follow this metaphor. In many software systems such as text editors or drawing tools the interaction with the system does not proceed by submission of forms that lead to a new page. Instead, the current screen is constantly updated as the interaction proceeds. However, submit/response style applications form a ubiquitous class of important systems, which justifies the development of an analysis method specifically designed for this type of systems.

Technically, submit/response style applications can appear as modern web applications or as client/server applications or of course as classic mainframe

applications. However, we deal in this thesis with the *analysis* of such systems, and the particular technical representation shall be transparent for the functional specification as a result of the analysis phase [126], hence we want to achieve a *specification* independent from *implementation*. It is therefore the key goal of this thesis to establish a high level view on this type of systems, in which we abstract from the underlying technology and focus on the inherent properties of submit/response style systems. Not every problem is amenable to a solution by a form-based system. But if a system is intuitively thought of as being accessible by a submit/response style interface this gives an important starting point for the problem analysis. In the analysis technique proposed here, called Form-Oriented Analysis, we will give a powerful approach to system modeling by understanding the system along its usage through a submit/response style interface. This interface model in Form-Oriented Analysis is an abstract interface; it is a conceptual tool for the understanding of the system. But it can be thought of as a working prototype of the actual application interface. Hence Form-Oriented Analysis is a technique for modeling a system along a prototypical submit/response style interface.

Our analysis technique is firmly based on existing well understood analysis notions and modeling techniques and consequently extends the state of the art in an important application domain: our analysis method is tailored to the class of submit/response style applications, but not designed as an analysis technique for all kinds of software systems.

This strategic decision allows Form-Oriented Analysis to fit optimally to submit/response style applications and to provide crucial added value for the analysis of such systems. In this way Form-Oriented Analysis goes beyond other one-size-fits-all techniques, as we will see in an elaborated comparison with other analysis techniques. We will compare Form-Oriented Analysis with the leading edge Object-Oriented Analysis technique [83] [50] [74] [81] and the well established and still extensively used Structured Analysis [63] [66]. We also set our approach in relation to the specification approach of the Z language [149], which also conceives a state based system as a state transition machine. Form-Oriented Analysis provides clear added value to other analysis models in that it provides an interface model while the other techniques do not. We also therefore review some interface modeling techniques, which however address chiefly design, not analysis, but could be used in the context of submit/response style interfaces [76] [72].

The restriction of Form-Oriented Analysis to submit/response style applications allows us to employ the clear semantics of submit/response style interfaces within the analysis phase. Hence a model obtained in form-based analysis benefits in its formal strictness and semantic clarity from the restriction to this interaction style. Form-Oriented Analysis covers the area of analysis which is typically called the functional specification. The notion of functional specification in this thesis is basically the same as the notion of functional specification from Object-Oriented Analysis and Structured Analysis. Our method therefore covers the same area as other analysis techniques, especially Structured Analysis. According to [127] a functional specification of a system should be a

document that

1. Is well understood and fully agreed to by the domain experts.
2. Sets out the logical requirements of the system without dictating a physical implementation, and
3. Expresses preferences and trade-offs.

There are other topics in the analysis phase that are not covered by these methods, including cost-benefit analysis, performance analysis, hardware selection and personal planning [63].

This thesis explains Form-Oriented Analysis by introducing the artifacts of our technique. Form-Oriented Analysis uses mainly visual artifacts for modeling. But in contrast to other visual modeling techniques we understand these artifacts mainly as a visualization of information, which also could be given in a textual representation. This flavor of visualization is important for Form-Oriented Analysis since it is a technique designed for tight integration into a suite of code based tools. One intuitive and paradigmatic tool of Form-Oriented Analysis is presented in the appendix of the thesis. This tool called Gently is able to generate a fully functional system prototype from a textual representation of a form storyboard and to perform consistency and type checks on the form storyboard description.

The main contributed artifact of Form-Oriented Analysis is the form chart as a semantically well-defined diagram. The other form-oriented diagram types, page diagram and form storyboard, are important informal predecessors of the form chart, which highlight specific aspects. Our method offers a simple yet powerful composition mechanism for artifacts, called feature composition. It has to be pointed out that a separation or juxtaposition of Form-Oriented Analysis against other techniques is artificial. On the contrary, the techniques work well together. A form chart can be used e.g. in Structured Analysis, vice versa in Form-Oriented Analysis one can use a data flow diagram if it is conceived as bringing an added value. In the same way Form-Oriented Analysis does not prescribe any process model. Of course, the different degree of formality of the different artifacts as well as the feature composition mechanism hints at a process like intuitive succession of diagrams form informal to formal, but it is important to realize that this is by no means necessary. Since the semantics of the diagrams, especially the formal semantics of the form chart, is completely independent from any process definition, the diagram is basically neutral with respect to its use in a process of whatever kind.

Form-Oriented Analysis was developed in a joint scientific project with Dirk Draheim. In this project a seamless methodology for submit/response style interfaces was developed on all abstraction layers of a typical software engineering concept i.e. the abstraction layers of requirement elicitation, analysis, design and implementation. The model obtained in Form-Oriented Analysis can be transformed without impedance mismatch into an interface design and an implementation based on well-established technologies like server pages for web interfaces.

The main contributions of this comprehensive joint scientific project are:

- Submit/response style systems follow a two-staged interaction paradigm, consisting of fine grained page interaction and coarse grained page change.
- Submit/response style systems have clear semantics and can be modeled with bipartite finite state machines.
- A complete system specification is given in four documents, a form chart, a dialogue constraint annotation, a data dictionary and a semantic data model.
- A new semantics of finite state transition diagrams as class diagrams is given.
- An extension of OCL by path expressions.
- The specification obtained by Form-Oriented Analysis is directly applicable to state-of-the-art system architectures. A tool for generating web application prototypes is fully implemented.
- A strongly typed server page mechanism as a new and secure foundation for web applications is developed.
- A powerful reverse engineering tool for web applications, JSPick has been developed.

Outline of the Thesis

In Chapter 2 we give a hands-on introduction to Form-Oriented Analysis based on a running example, a seminar registration system. Related approaches are discussed intensively throughout the thesis, but in Chapter 3 we give a dedicated discussion on selected related work. In Chapter 4 we present the central artifacts of Form-Oriented Analysis in depth. In Chapter 5 we give a formal operational semantics based on the UML notation. In the Chapter 6 we sum up the results and reflect on submit/response style systems. In the Appendix A we give the full constraint specification of our example system. In the Appendix B we explain the tool Gently.