A Formal Specification Automation Method Based on Focus Framework

Xiuna Zhu

***Abstract*—An open issue in the area of formal methods is the automation methods and their support tools which need to be applied in the specification process. The automated methodology not only saves time, but also excludes human factors that lead to failure, so it is becoming a trend of model-driven development technology in requirements engineering. For this purpose, we proposed and implemented a formal specification automation process which aims to generate formal specification documents from software engineering models. Our work focuses on the automatic generation of requirements specification in the Focus modeling framework from AutoFocus models. The proposed method not only generates a readable specification of the requirements, but also provides an effective way to combine the CASE tool AutoFocus and the Focus framework.**

***Index Terms*—Focus framework, formal methods, formal specification.**

I. Introduction

System and software requirements documents play a crucial role in software engineering. They must both communicate requirements to clients in an understandable manner and define requirements in precise detail for system developers [1]. The formal specifications are useful for precisely documenting program behavior and guiding implementation. Deriving these specifications is more complicated and takes more time compared with using natural language description. Thus, the formal specification is still limited in its use in industry. One of the common problems is that the formal specification documents are often out-of-date or do not match with the requirements after several development phases. The outdated system documentation problem is usually considered a secondary issue and overlooked because of timing or cost constraints. What is more, in model-driven development, the formal specification is hard to keep up-to-date when the system model is frequently changing [2]. For instance, after step-by-step refinement, more and more detail information of the system model is added to clarify requirements and remove ambiguity, inconsistency, and incompleteness; then the formal specification of the new system model is needed. In addition, software implementations are often not consistent with the documents, since developers do not update system specification documents when they change the design models. Therefore, a method and its tool allowing reverse engineering would be useful if the formal requirements specification were generated based on the updated design.

Additionally, a formal specification can contain mistakes or disagree with requirements because of human factors of formal methods [3]. Experience in formal specification reveals that different developers with the same background may end up with fairly different specifications for the same initial problem formulation [4]. In order to unify of the formal specification and decrease the need for manual inspection between system models and formal mathematical specifications, a set of methods and tools is needed to obtain a development process that supports the automation. Modern approaches of model-based development try to use generator technologies to provide automatic support to implement formal specification automation. This implies that automation of some system development steps is considered as a proposal for reducing human fault risk. We can validate and verify the generated specifications to be sure that they conform to their requirements.

The overall objective of this work is to carry out a comprehensive review and synthesis of previous and current research to find methods dealing with the generation, translation, integration, or synchronization of system or software models and textual requirements. We focus on the combination of software engineering models and textual requirements in the requirements specification process, in particular those approaches that generate textual formal specification with which to document the models. Our intention is to study a formal specification automation method and its support tools. The reverse problem, going from textual requirements to models, is discussed in [5], which is not within the scope of this paper. Based on synchronous semantics for the behavior of systems, we sketch how formal Focus specifications can be generated from Auto Focus (AF) models. The feasibility of this approach was proven in a number of case studies, including adaptive cruise control system, which will be illustrated in this paper.

The rest of this paper is organized as follows. In Section II, we take a closer look at the main features of the Focus framework and the AF modeling environment used in this paper. Section III gives an overview of our approach, while Section IV introduces the model transformation from AF executable system models to formal specifications on Focus framework. In Section V, an example is given to illustrate this approach. Finally, we conclude the paper in Section VI.

II. Focus and Its CASE Tool Auto Focus

A. Focus Framework

The Focus framework [6] has been developed for the formal specification and stepwise development and

Manuscript received December 10, 2013; revised February 7, 2014. This work was supported by the Chair of Software & Systems Engineering at Technische Universität München.

Xiuna Zhu is with China Scholarship Council (CSC) scholarships as a Ph.D. student working at the Institute of Software & Systems Engineering at Technische Universität München, München, Germany (e-mail: zhux@in.tum.de).

management of distributed and interactive systems. The development process of Focus is based on modeling systems as networks of components communicating asynchronously via unbounded, directed channels [7]. An interface specification is characterized through the relation between communication histories for the external input and output channels [8].

In Focus framework, *stream* is one of the main concepts. Channels connect two subsystems or a system and its environment. Streams represent the communication histories of directed channels. They are formally represented by functions mapping the indices in their time domains to their messages. Composed systems are defined by recursive equations over streams (see [9]).

As other model-based formal specification languages, such as the Z language for Z method [10], the B language for B method [11], and VDM-SL language for VDM, the syntax of Focus specification is well-defined. The Focus framework is preferred here over other specification frameworks since it has an integrated notion of time and modeling techniques for distributed systems, and concepts of refinement. Furthermore, a variety of specification styles are provided in the Focus framework, such as logical formulas, diagrams, and tables, and *Assumption / Guarantee* style is the most common specification technology.

B. Auto Focus Modeling Environment

AF CASE Tool is a scientific prototype implementing a modeling language based on a graphical notation. In Auto Focus modeling environment, we use three different views to build the model of the system under development: interface and structure view, behavior view, and data view. We define the data type and basic functions using the data view. The system architecture is specified in the interface and structure view. And the system's behavior is defined by I/O automata in the state transition behavior view [12]. The AF modeling method allows multi-format modeling of the behavior by means of specification extensions, such as state automaton specification, code specification, and tabular specification. Therefore, it has more domain-based specific ways to specify the behavior of components and supplies more user-friendly model specification in model-based development process. The semantics of the AF models are defined in [9].

III. Integration in Model-based Development

A. Former Modeling Method

Previous researches [13], [14] outline a formal development methodology for modeling systems with the AF tool chain. We start from an informal textual specification of the requirements. Through multiple transformation steps, a verified formal specification, a verified executable AF model, and a verified C/C0 code implementation can be obtained. The C/C0 code can be automatically generated from the AF models; nevertheless some transformation steps are manually and cumbersome, e.g., bidirectional transformation between Focus specification and executable AF models. Particularly acquiring the formal Focus specification from AF models is difficult and time-consuming.

The Auto Focus is designed to work in seamless integrated manner and easy to extend with advanced features. A number of add-ons, to be designed and developed, could be integrated into the tools to extend the formal model check and theorem proof by add-ons. Furthermore, the formats of the behavior specification in the AF modeling environment are manifold, so that the specification method that fits best for the problem at hand can be chosen. Unless a stakeholder is familiar with a model, it can be difficult to determine where to start reading a model in a modeling tool. Similar with other visual modeling languages or tools, not all stakeholders of AF are able to understand the syntax and semantics of the models. In order to access the information embedded in the model, it may require the users to master a modeling tool. Meanwhile, manually operations may increase the human fault risk. Thus, a formal specification automatic generation method that can obtain the specifications with unambiguous and easy-to-understand notations by the modeling tools will help improve the former modeling method and change the current situation of limited practical use.

B. Strategy Overview

The proposed paper is an extension of previous work [13]. The context of this work is research cooperation between the Focus framework and the AF modeling method. Our approach focuses on automating formal methods used within the formal specification phase of the system development process. The main point in our method is an alignment with the future proofs to make them simpler and appropriate for an application, not only in theory but also in practice.

For this purpose, we propose to extend the AF modeling method and to implement an automatically formal specification generator for Focus framework. In our case, correctness, clearness, and readability of a formal specification need to be considered. We propose to apply all of the validation and verification methods in the former modeling environment. For instance, the TVARC model checker is applied to check the C/C0 code generated form the AF3 model and SMV checker is used directly to the designed model, e.g., Z3 or NuSMV.

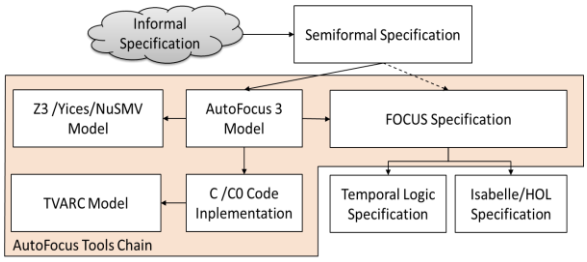


Fig. 1. Overview of proposed strategy.

IV. Model Transformation

In Focus framework, a system is represented by a set of components. A component has a syntactical interface described as a set of input and output ports. There ports are used to interconnect several components for communication. The components are connected by channels and exchange information in terms of messages of specified types. The Focus framework has primitive types for basic values; meanwhile, algebraic data types may be defined by the user. A library of predefined data types and operators is provided.

Ports and assigned data types to restrict the messages allowed to be transmitted.

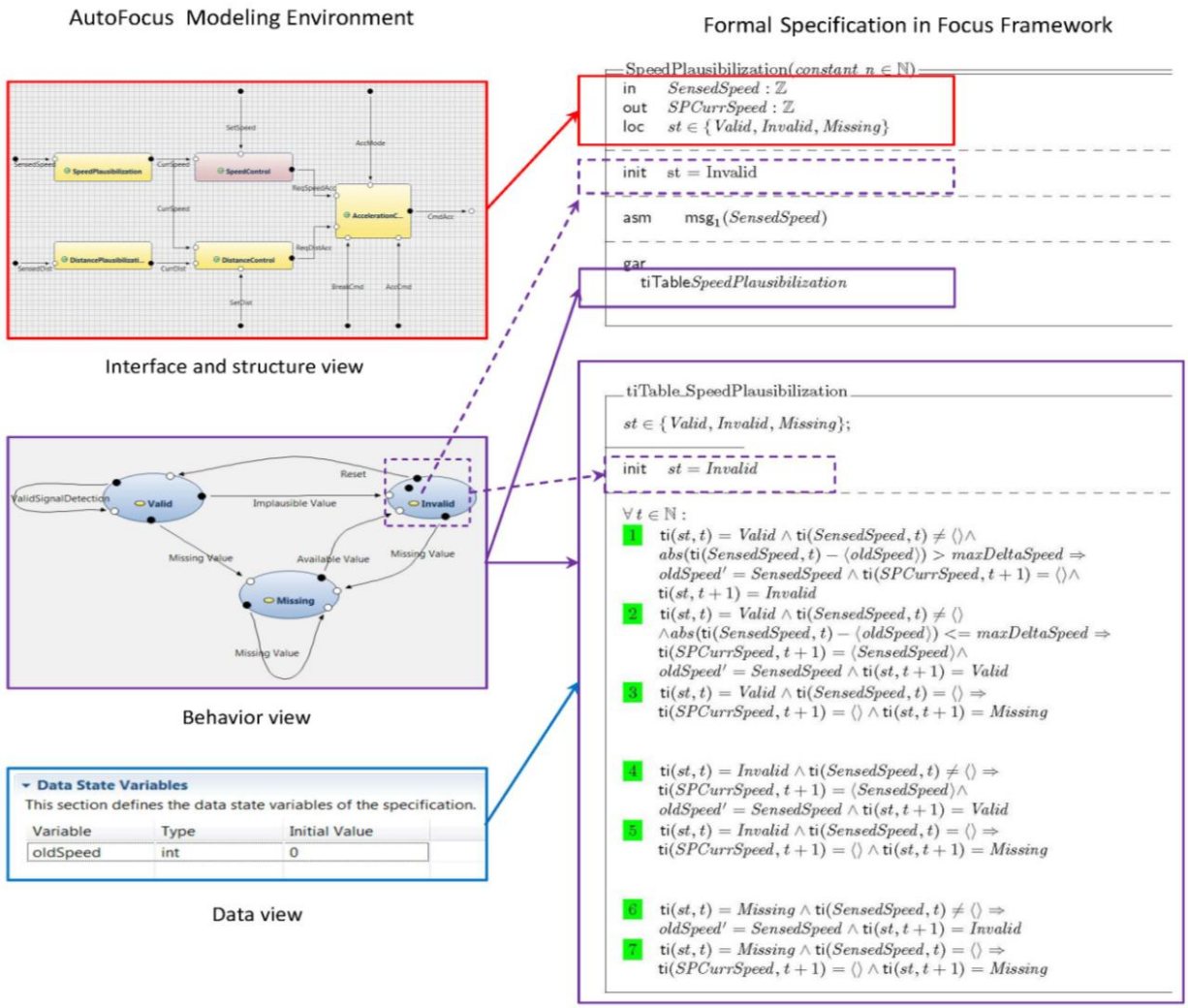


Fig. 2. Schematic translation of component speed plausibilization.

As shown in Fig. 1, we create the AF model according to semiformal requirements of the system using the AF modeling tools Chain. The AF models are under development with three views, interface and structure view, behavior view, and data view, respectively. Since the descriptions in the different view are considered, the semantics of the model in each perspective are synchronized. Our work emphasizes on synchronizing semantics to transform models from AF model to the Focus specification.

The process of model transformation could be suggested in several steps:

1) firstly, in data view mapping message types to data definitions in specification, e.g. local variables, data types and expression terms;
2) secondly, in behavior view we construct behaviorally equivalent for user-defined AF functions and specify input/output behavior for each component with the Focus syntax;
3) thirdly, in structure view we specify the component hierarchies of the system according to the internal communication;
4) lastly, complete the generated Focus specification.

System in Focus framework consists of named component models and channels with discrete global time; thus we consider the composition of behavior specification. To describe interface of the component, we translate the information of messages in AF models to data type definition in Focus framework. By mapping functionalities to logical components, and translate state automata-based behavior model to stream-based specification, we specify the behavior of the component on the Focus framework. Using streams to model communication history of channels, then composed systems are defined by recursive equations over streams. Finally, system structure is specified, and the Focus specification is completed. Each AF model can be translated into a formal specification in different refinement layers in Focus framework, and each input/output port of the produces a named channel with possibly typed. For each general step, the proof of simulation equivalence between the AF model semantics is given in [14], and the Focus semantics is given in [9].

V. Case Study

Technical report [13] presents the main methodological requirements. In this section, we use an adaptive cruise control (ACC) system as an example to illustrate and clarify

the process. Note that the modeling details do not be presented in detail in this paper. More examples of AF models are available in the website of the AF [15].

A. Translation of the Component

Based on synchronous semantics for the behavior of systems, we generate the formal specification of a component according to the representation of an AF model. As shown in Fig. 2, the behavior of component *Speed* Plausibilization is specified by stream histories of the input/output channels in the Focus framework. In the specified component *Speed Plausibilization*, assumption and guarantee are specified in terms of *asm* and *gar*. When assumption holds, the guarantee is required to fulfill. By integrating the information of the Auto Focus model, such as External Syntactic Interface in the interface and structure view (red block), State Transition Behavior of the component in the behavior view (purple block), and Data State Variables in the data view (blue block), we extended our structure hierarchy specification with its internal structure and translated the behavior representation into the Focus syntax.

B. Translation of the State Automaton

In AF modeling environment, automaton specification can represent the logical component as a timed state transition diagram. We implement the transformation from a single transition segment in an AF model to a transition rule with the formal notation of the Focus framework. The automaton specifications' data, e.g. the current control state and the data state variables, will be dynamically mapped into the state space of the Focus specification. As shown in Fig.2, the initial state in the state automaton can be mapped into the initial value of the local variable state (purple dotted block).

C. LaTeX Sugar for the Focus Framework

To increase the efficiency of the specification process, a visualization editor tool for the Focus operators and frames should be provided. It provides modern writing support, like interactive spell checking, code folding, and syntax highlighting. The proposed Focus Sugar Editor not only inherited the most of the functions from an open source plugin TeXlipse, but also was extended with additional features for the operators and frames in Focus framework. The tool support for LaTeX sugar for the Focus framework is considered as a contribution of the approach proposed in this paper.

VI. Conclusion

The use of automation techniques has proven to be a useful method for the specification and validation of the formal method, e.g., Autofocus modeling tools chain, Focus specification generator and Focus LaTeX sugar editor. We used this approach to software development to replace and extend an existing implementation of use case. We found that this method provided following benefits over previous software developments, in addition to being easy-to-use and minimizing the human error.

First, this method bridges the gap between the requirements elicitation and formal requirements specification. The design and the implementation are tightly coupled, so the documentation is always up-to-date with respect to the executable models.

Secondly, the tool can accelerate the updates and corrections activities of the design; the developer need only modify the diagram or the content of the table and regenerate the behavior specification in the Focus semantic.

Thirdly, this framework can give full play to the strong advantages of both methods. These benefits can result in reduced development and testing time and a reduction in formal method software development and maintenance costs.

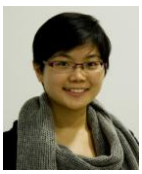
There are some additional challenges involved in adopting a model-based development approach. Optimization of the Focus language to specify some trivial case implicitly is another topic covered in this work.

Acknowledgment

This work was carried out at the chair of Software and System Engineering, TU München. We thank Prof. Manfred Broy for his continued support for this work. We also gratefully acknowledge Dongyue Mou's numerous discussions about the technical implementation of the tools.

References

[1] J. Nicolás and A. Toval, "On the generation of requirements specifications from software engineering models: A systematic literature review," *Information and Software Technology*, vol. 51, pp. 1291-1307, Sept. 2009.
[2] M. Spichkova, X. Zhu, and D. Mou, "Do we really need to write documentation for a system? - CASE tool add-ons: generator + editor for a precise documentation," in *Proc. International Conference on Model-Driven Engineering and Software Development*, vol. 52, 2013.
[3] M. Spichkova, "Human factors of formal methods," *Interfaces and Human Computer Interaction*, Lisbon, Portugal: IADIS, 2012.
[4] V. Lamsweerde, Axel, "Formal specification: a roadmap," in *Proc. Conference on the Future of Software Engineering*, New York, NY, USA, pp. 147-159, 2000
[5] S. Teufl, D. Mou, and D. Ratiu, "MIRA: A tooling-framework to experiment with model-based requirements engineering," *Rio De Janeiro*, Brasil: RE, pp. 330-331, 2013.
[6] Homepage of the Focus framework. [Online]. Available: http://www.focus.in.tum.de
[7] M. Broy, I. Krüger and M. Meisinger, "A formal model of services," *Lecture Notes in Computer Science*, pp. 385-389, 2006.
[8] C. Kühnel and M. Spichkova, "FlexRay und FTCom: formal spezifikation in focus," Technical Report TUM-I0601, Technische Univerität München, 2006.
[9] M. Broy and K. Stølen, *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*, Springer, 2001.
[10] J. P. Bowen, *Formal Specification and Documentation Using: A Case Study Approach*, International Thomson Computer Press, 1996.
[11] J. R. Abrial, *The B-Book: Assigning Programs to Meanings*, Cambridge University Press, New York, NY, USA.1996.
[12] C. Döbber and F. Hölzl, "Extending autoFocus 3 with Behavior," Institute für Informatik, Technische Univerität München, 2011.
[13] F. Hölzl, M. Spichkova, and D. Trachtenherz, "Auto Focus tool chain," Technical Report TUM-I1021, Technische Universität München, 2010.
[14] F. Hölzl, "The AutoFOCUS C0 Code Generator," Technical Report TUM-I0918, Technische Universität München, 2009.
[15] Homepage of the AutoFocus. [Online]. Available: http://www.af3.fortiss.org



Xiuna Zhu was born in Zhejiang, China on 30 December 1984. She received her M.Sc. in computer science from the China Agriculture University, in 2009. Since Oct. 2009, she is a Ph.D. student at the chair of software and system engineering at Technische Universität München, Germany. Her currently research is focused on model-based requirements engineering, model-driven development of embedded systems, and the formal specification methods for the complex systems, especially Tabular specification method.