RenPar'19 / SympA'13 / CFSE'7
Toulouse, France, du 9 au 11 septembre 2009

Multi-view specification of component-based operating systems through model refinement

Guillaume Libersat, Frédéric Loiret, Lionel Seinturier

University of Lille 1 / INRIA
LIFL UMR CNRS 8022, ADAM Project-Team
Villeuneuve d'Ascq, France
E-mail : {firstname.lastname}@inria.fr

Résumé
Ce papier présente une méthodologie outillée de haut niveau tirant profit des multi-vues, des composants et des modèles dans le but de faciliter la construction des systèmes d'exploitation complexes.

Mots-clés : système d'exploitation, modèles, raffinement, composant, multi-vues.

---

1. Context

Modern operating systems face stringent requirements which increase their size continuously. They are therefore becoming more and more complex. They need, for example, to run on numerous architectures or to embed many drivers. Therefore, it potentially favors bugs and security flaws and makes the OS less maintainable (such as simple changes triggering pernicious side effects [9]).
The Component-based software development [6] is one of the possible answers to deal with this complexity : it helps building software in a modular, reusable and cleaner way, while providing introspection and reconfiguration capabilities at runtime. Model-driven engineering, on its side, has proved over the years to be an efficient way of representing programming concepts in an unified way either at design-time or at runtime.

2. Models and Components for Operating Systems ?

While OS development could take advantage of Models and Components, they haven't really embraced them since OSes are often very constrained (such as by their memory footprint). One possible approach is to use them at design-time and discard them when producing the code. Previous works such as Think [5] have started exploring this path and proved its feasibility.
Based on this approach and pushing it further, we propose a workbench coupled with a methodology based on Components and Models. We therefore reduce design complexity and improve maintainability while sticking to the efficiency requirements of such systems and encouraging reusability.

3. A tool-assisted methodology for OS design

The methodology we propose is based on the following key ideas : (*i*) work at a very high level of abstraction ; (*ii*) only expose relevant information ; (*iii*) construct by refinement ; (*iv*) maintain the construct in a safe state and (*v*) project models to feed a compilation chain.
Working at a high level. We propose an extensible metamodel that captures the concepts needed to describe an OS that brings into play many concerns (such as memory allocation or scheduling). Using metamodel benefits, we can build tools working with any component technology and can later reason on rich metadata to produce the resulting code.
Exposing pertinent information. As designing an OS means dealing with many domains, it involves working with legion of models. We therefore need a separation of concerns [8] to cap the complexity. We use a multi-view approach to solve that, where each expert only manipulates the concepts related

to his/her domain. We also introduce an extensible Domain Specific Language (DSL) which enables an easy navigation and querying of Models using the domain's terminology ; preventing the domain expert from writing repetitive and error-prone code.

Constructing through refinement. In our approach, we build an OS by starting with a consistent set of Models and by incrementally refining them. We have defined three kinds of atomic and reversible operations on the Models : creation, transformation and deletion. We allow composition and nesting in order to build high level operations. Also, to ease the manipulation of model operations, we provide a syntactic sugar that only makes use of the domain terminologies.

Keeping the construct safe. To ensure the construct is correct, we add constraints to Models. Constraints can either be expressed for a single domain or between two domains. On top of them, we build transactional operations that can be atomically reverted if a constraint is violated. They prevent execution of wrong operations and guide the developer in the construction process.

Projecting Models. Once the OS specification is done, domain views get merged by taking advantage of the rich metadata available from the models and the code is generated. We produce the required Architecture Description Language (ADL) files (explaining how components need to be assembled) and Aspects (representing a given concern such optimization) for the compilation chain.

4. Related Works

[7] presents a way to maintain the consistency in a multiview approach using formal languages. It therefore relies on the ability of the domain experts to use such languages, what we want to avoid.

Many approaches to model-driven development for OSes have been proposed. Most of them although tends to be specific to a range of hardware [2], a particular computational model [4] or focuses on a single concern [3] . Also most of these works do not deal with components except [1] but, as a drawback, their considerations are (almost-)only architecture-centric.

Bibliographie

1. Adam Childs, Xianghua Deng, Matthew B. Dwyer, Jesse Greenwald, John Hatcliff, Prashant Kumar, Georg Jung, Venkatesh Ranganath, Robby, and Gurdip Singh. Supporting model-driven development of component-based embedded systems with cadena, 2003.
2. Mirko Conrad and Heiko Dörr. Model-based development of in-vehicle software. In *DATE '06 : Proceedings of the conference on Design, automation and test in Europe*, pages 89–90, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.
3. Matthew Eby, Jan Werner, Gabor Karsai, and Akos Ledeczi. Integrating security modeling into embedded system design. In *ECBS '07 : Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 221–228, Washington, DC, USA, 2007. IEEE Computer Society.
4. Abdoulaye Gamatié, Sébastien Le Beux, Éric Piel, Anne Etien, Rabie Ben Atitallah, Philippe Marquet, and Jean-Luc Dekeyser. A Model Driven Design Framework for High Performance Embedded Systems. Research Report RR-6614, INRIA, 2008.
5. Frédéric Loiret, Juan Navas, Jean-Philippe Babau, and Olivier Lobry. Component-Based Real-Time Operating System for Embedded Applications. In *Proceedings of the* 12^{th} *International SIGSOFT Symposium on Component-Based Software Engineering (CBSE'09)*, Lecture Notes in Computer Science, East Stroudsburg, Pennsylvania, USA, jun 2009. Springer.
6. D. Mcilroy. Mass-produced software components. In *Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany*, pages 88–98, 1968.
7. Richard F. Paige, Phillip J. Brooke, and Jonathan S. Ostroff. Metamodel-based model conformance and multiview consistency checking. *ACM Trans. Softw. Eng. Methodol.*, 16(3) :11, 2007.
8. D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12) :1053–1058, 1972.
9. Liguo Yu and Srini Ramaswamy. Change propagations in the maintenance of kernel-based software with a study on linux. In *ACM-SE 45 : Proceedings of the 45th annual southeast regional conference*, pages 76–81, New York, NY, USA, 2007. ACM.