Supporting An Architecture-Based Approach to Mission Modeling

John C. Georgas and Richard N. Taylor
Institute for Software Research
University of California, Irvine
Irvine, CA 92697, U.S.A.
+1 949 824 5160
{jgeorgas, taylor}@ics.uci.edu

ABSTRACT

Designing a space exploration mission involves many varied yet interconnected domains which are often found to be in conflict with each other. A conceptually coherent architectural model with a solid syntactic base provides for the identification and resolution of these relationships and conflicts with automated facilities, as well as enabling advanced analysis and simulation capabilities. This presentation presents the results of leveraging the domain independent XML-based framework associated with the xADL project to provide the mission design field with the technical infrastructure needed to make the most of architectural models.

1. INTRODUCTION

The design of a space exploration mission is a complex process involving a large number of different groups. Each of these groups is charged with the design of heterogeneous artifacts; the work of the control software team, for example, is very different from the work of the cost analysis group. Further differentiating the domains within which these groups operate are differences in tools, process, and, perhaps most importantly, in the underlying models that represent the artifacts these groups deal with.

As different as these domains dealing with the different aspects of a mission are, they may still have significant relationships between them. The result of these relationships is that the work of one group may have far-reaching effects on the work of another. Significant problems arise during mission design because these effects are only implicitly represented in the various artifacts produced by domain experts. Conflicts and cause-effect relationships are mostly reconciled through interpersonal communication between designers.

The Architecture Working Group (AWG) within the Consultative Committee for Space Data Systems (CCSDS) is engaged in efforts to rectify some of the mentioned problems by establishing an overall CCSDS approach to the architectural models used in mission design as well as the development process [2]. To support the conceptual underpinnings developed by this effort, we present an architecture-based approach to the modeling of missions using concepts and technologies rooted in the domain of software architecture.

Specifically, we leveraged the XML-based framework described in [4] to support the specification of mission models and their syntactic representation, provide mission-specific tools to support the mission design process, and offer the basis for mission simulation using tools from the Ptolemy project.

2. ARCHITECTURE-BASED APPROACH

High-level software architecture descriptions provide a way to reason about the constituent components of an overall system and the interconnections between them [6]; system behavior is expressed as the collaborations between components. The development framework associated with xADL 2.0 [3] provides a domain independent framework to define an architecture description language (ADL) as well as a collection of tools which provide substantial support for the representation and manipulation of systems described in this ADL [4]. Though originating in the field of software engineering, these technologies are essentially domain independent. It is this domain independence that enables the use of this framework in the field of space mission design.

The initial efforts in this architecture-based approach revolve around five main tasks: model specification, syntactic representation, tool use, space mission specific support, and simulation.

2.1 Model Specification

Eliciting and specifying a model that can be used to describe space missions is a challenging undertaking. The various domain experts involved in all the facets of a mission have to reach an agreement about a great number of concepts including the types of components present in a system, their interactions, and the optimal way to model these elements. Though the specification of this concrete model lies at the very core of any attempt to create a meaningful development framework, it was not the main concern of our effort. After all, it is the domain experts that are best-suited to define the models that are used in their field.

Instead, our main contribution to the specification of these models centered around enabling the transparency of relationships between models representing the many varied domains of a space mission. For example, there is a very strong relationship between the models governing the physical artifacts comprising a space probe with the models representing the communication attributes of the overall mission. In current practice, these relationships are embedded into each domain's model. It takes a significant amount of cross-domain knowledge and experience in mission design for an individual to recognize these complex relationships, the effects they have on the overall space mission design, and the best way by which to resolve any arising conflicts.

The solution we propose is the treatment of these cross-model relationships as first-class entities – treating them as objects themselves rather than attributes of other objects. Once identified, these relationships can then be easily monitored and maintained as the mission design evolves toward its final form. Furthermore, by making these relationships so visible, the amount of domain knowledge that is needed for their monitoring and maintenance is significantly lessened. In the models used as part of this effort, cross-domain relationships are maintained separately from the models they link and, in essence, form their own domain. Links from each relationship entity to the elements from other domains to which it pertains are maintained. Therefore, changes in these linked elements would trigger any sort of automated response deemed necessary by the mission designer, such as the addition of affected relationships to a mission designer's "to-do" list for later consideration.

Additional concepts incorporated in the model specifications include construct versioning, typing, and sub-architecture definition capabilities.

2.2 Model Representation

Essential for the manipulation of these specified models is a well-defined syntactic representation. The technological basis of the framework we used is XML, so the structure of these models is first specified using XML schemas. These schema definitions are used as an input to 'apigen', a framework-provided tool that generates Java data binding libraries allowing the manipulation of architectural documents through function calls rather than direct editing of the XML specification. This provides a higher level of abstraction and provides functions that deal with the constructs that a mission designer would expect to deal with such as components and connectors. By providing an interface to the manipulation and maintenance of model representations that uses concepts familiar to mission designers, the barrier of technology adoption is significantly lowered.

2.3 Tool Support

Architectural descriptions are not very useful without the necessary tool support to provide editing and visualization capabilities. Adoption of the set of xADL technologies offers more capabilities than just the generation of data binding libraries that 'apigen' performs.

The framework includes ArchEdit, which is a context-aware, lightweight editor. ArchEdit adjusts to handle any given set of data-binding libraries that is used, meaning that there needs to be no modification of the editor for use with new or modified architectural schemas. The editing capabilities of ArchEdit are, however, text-driven and low-level. Nevertheless, always having an available editor for any set of schemas defining a model is a significant benefit in that no customized editing solution needs to be created or customized every time new models are created or old ones modified.

2.4 Mission-Specific Support

Even though generic tools are valuable, in order to fully provide conceptual models with utility, domain-specific support is required. The specialized tools created for this mission-modeling effort center around design-time sanity checking of mission architectural descriptions, and the maintenance of first-class relationship entities.

In any complex domain such as space mission design, there is a variety of constraints placed on the manner in which components and connections between them may be established. In essence, these are the semantic constraints placed upon a design. To develop components that maintain these constraints, we leveraged the Critic framework of ArchStudio 3.0 [1]. ArchStudio 3.0 is a software development environment incorporating the XML-based framework already discussed, and the Critic framework is a combination of an API and a collection of reusable managerial components that ease the development process of design-time critic components. In addition to the maintenance of the first-class relationship objects already discussed, the initial Critic components we developed monitored basic concepts such as type-matching and required properties. The facilities of type-matching revolve around ensuring that a component or connector instance reflects the properties and characteristics of the type the instance belongs to; this enables type-based analysis of resulting mission designs. Further functionality that mission-specific Critics implement is the ability to require that designers complete certain information vital to a mission design. A simple example of this is requiring that any mission component has a name value associated with it.

2.5 Simulation Framework

Mission simulation is perhaps the most interesting functionality that a well-specified mission-design architectural model would enable. As an attempt to provide facilities for this, we attempted an initial integration with the Ptolemy simulation framework [5]. The Ptolemy project provides a collection of libraries supporting simulation in various domains. Ptolemy files are saved in the Modeling Markup Language (MoML), an XML-based format. Given the core similarity between the formats of the architectural files our approach generates and of those used by Ptolemy, an integration between the two systems is particularly appropriate.

Our translation component generates a MoML description of a mission architecture for use with the Ptolemy libraries. This description is limited by the fact that the modeling concepts we used, in their current form, have little semantic information about the behavior of the elements they contain. This is simply due to the limited time available for this integration effort and is not indicative of any technological limitations of the framework itself. Therefore, what our translation generates is a framework outlining the general structure of a Ptolemy model mirroring the simulated mission architecture; in order for the simulation to be complete, a mission designer would have to use the Ptolemy editing tools to fully specify the semantics of element behavior. A more robust mission model could support a more detailed level of semantic translation to the Ptolemy framework.

3. IMPORTANT QUESTIONS

During the course of our work, we confronted a variety of interesting questions about the nature of mission-design models. Satisfactorily addressing these issues will be an important task if an overall architectural model is to be established in this domain. Two of the most interesting of these concerns revolve around the coupling between design and implementation, and model design.

Abstract models of complex constructs are used in a variety of domains. These models allow designers to achieve clarity of vision about their work as well as to effectively communicate design ideas. Furthermore, in most of these domains, models correspond to concrete constructs. For example, the models used in our effort contain elements that correspond to sensors, maneuvering thrusters, or computer control systems. An important consideration for any of these domains is the tightness of the coupling between the model and the concrete implementation that this model abstracts, including ways to ensure this coupling remains strong throughout the design lifecycle. A technique in the software engineering domain, for example, is to automatically generate source code based on model specifications. Is a generative approach such as this appropriate in the mission design domain, and exactly what kinds of constructs would be generated? Are there any other ways to achieve a tight coupling between mission models and their eventual implementations? Architectural mission design models are very valuable during design-time, but questions such as these need to be answered in order for their utility to extend throughout the mission lifecycle.

Space exploration missions, as already mentioned, involve various interconnected domains. Yet, missions are collectively composed of this multitude of domains. The approach we have taken in our work is to represent each of these heterogeneous domains as a separate model, and designate connections between them as first-class entities. This allows for a clear separation between models for clarity, and well-separated model version management. However, this is not the only approach that could have been taken. If these various domains compose the whole of a mission, a comprehensive model can perhaps be constructed. The collection of models interconnected by various relationships is replaced by a central model containing the exact same concepts. It is not clear whether this kind of central model would be any more useful than other approaches, or indeed even possible. But, as it is commonly thought that the different aspects of complex constructs are only different views of a central entity; perhaps it is this central entity that needs to be modeled.

4. CONCLUSION

A well-defined syntactic base for mission design models is necessary to support the technical underpinnings needed for a full-fledged development environment and a tight coupling to the resulting mission implementation. We have leveraged domain independent xADL-related technologies to provide this syntactic support and show a possible structure of a mission-design development environment, as well as the kinds of analytic and simulation support it could provide. Adopting the technology framework that we used in support of an architecture-based mission modeling approach eases the development process of new mission modeling concepts and tools, as much of the infrastructure for their rapid development is already provided. This allows the various domain experts to focus on the elicitation and definition of space mission models, and experimentation with the ramifications of modeling choices rather than be concerned with the details of the technical infrastructure.

5. ACKNOWLEDGMENTS

The authors would like to thank Peter Shames and Nicolas Rouquette of the Jet Propulsion Laboratory for their support during this effort.

6. REFERENCES

[1] ArchStudio 3.0. URL: http://www.isr.uci.edu/projects/archstudio/

[2] CCSDS. URL: http://www.ccsds.org/

[3] E. Dashofy, A. van der Hoek, and R. N. Taylor. *A Highly-Extensible, XML-Based Architecture Description Language.* In Proceedings of the Working IEEE/IFIP Conference on Software Architectures (WICSA 2001), Amsterdam, Netherlands.

[4] Eric M. Dashofy, André van der Hoek, and Richard N. Taylor. *An Infrastructure for the Rapid Development of XML-based Architecture Description Languages.* In Proceedings of the 24th International Conference on Software Engineering (ICSE2002), Orlando, Florida.

[5] Edward A. Lee. *Overview of the Ptolemy Project.* Technical Memorandum UCB/ERL M01/11, University of California, Berkeley, March 6, 2001.

[6] D.E. Perry and A. L. Wolf. *Foundations for the Study of Software Architectures.* ACM SIGSOFT Software Engineering Notes, pages 40-52, October 1992.