Design Patterns and Real-time Object-oriented Modeling

Ross McKegney
Dept. of Computing & Information Science
Queen's University,
Kingston Ontario Canada
mckegney@cs.queensu.ca

Dr. Terry Shepard
Dept. of Electrical & Computer Engineering
Royal Military College of Canada
Kingston Ontario Canada
shepard@rmc.ca

Real-time object-oriented modeling tools (e.g. Rational Rose-RT, i-Logix Rhapsody) allow developers to focus on software architecture by abstracting away low-level implementation details. We believe that design patterns can be very beneficial in this context, and present the rationale and concepts behind a proposal for the extension of such a toolset to support them explicitly.

1. Introduction

The design and development of real-time software (i.e. software that must ensure timeliness while interacting with an external environment) is more difficult than most other software. Modeling tools (e.g. [6,15]) help deal with this complexity, allowing developers to view the system at various levels of abstraction, animate the models in a simulation environment, and even generate the code for a variety of target hardware/ RTOS configurations. A natural extension to these tools is to provide support for design patterns (a method of documenting experience in the form of problem/context/solution triples for recurring problems). Such an extension provides yet another layer of abstraction to the models, and makes explicit the application of design patterns.

Rational Rose-Realtime (Rose-RT) will be used as the basis for a concrete example of how such an abstraction layer might be implemented.

2. Rose-RT Toolset

Rose-RT extends the Rational Rose visual modeling tool with model execution and code generation capabilities from ObjecTime Limited [13]. Models are built using active objects (called capsules) that interact with each other through signal-based boundary objects called 'ports'. Each capsule has an associated hierarchical state machine (that specifies its behaviour); complex capsules can also contain sub-capsules. Complete C++ based executables can be generated directly from the models – for a variety of hardware/RTOS targets.

The Rose-RT toolset is an excellent candidate for the extension proposed for three reasons: first, the tool allows visual modeling of component structure and behaviour; second, the tool supports explicit modeling of issues like concurrency, distribution and timeliness; third, the code generation capability means that the design can be followed all the way to the target.

OOPSLA 2000 Companion Minneapolis, Minnesota

3. RTOO Design Patterns

The patterns concept has proved useful among developers of real-time software (i.e. software that must ensure timeliness while interacting with an external environment). A survey of design patterns applicable to real-time object-oriented (RTOO) software is included in [9], and summarized below.

The existing RTOO patterns can be classified as either general purpose or domain specific. The general purpose patterns include: Douglass' real-time design patterns [5] addressing a variety of issues such as distribution, state behaviour, safety and reliability; Schmidt et. al.'s patterns for concurrent, parallel, and distributed systems [16]; and other patterns (e.g. [2,7]). The domain specific patterns can be sub-categorized as process control, tele-communications, or embedded computer systems. The process control patterns include patterns for assisting the design of avionics control systems [8], and fire alarm control systems [11]. The telecommunications domain has a rich set of patterns (e.g. [3,10,14]) for dealing with the performance and robustness required by such systems. Finally, the embedded computer systems patterns (e.g. [1,4,12]) discuss issues such as resource management and optimization, porting issues, and scheduling.

4. Proposal

We present a proposal for extension to Rational Rose-RT to support design patterns. The following are the features that we feel are desirable in such a tool:

- **Multiple pattern repositories:** Provide support for multiple pattern repositories, with the possibility to share repositories over a network.
- **Improved classification mechanism:** An improved mechanism for structuring the potentially large sets of patterns to facilitate finding applicable patterns.
- **Ease of Expansion:** Relatively easy to add or extend patterns.
- **Explicit support for pattern languages:** Cross-referencing between patterns in languages, and the ability to specify relationships, including integration issues, between patterns.
- **Multiple pattern implementations:** Associate multiple concrete implementations with each abstract pattern specification.
- **Multiple implementation types:** Allow pattern implementations to consist of Rose-RT behaviour models and/or structure models, or files of another type.

- **Highlight patterns in models:** Keep track of where patterns have been used in a model, and make it easy for a developer to see where and how these patterns have been applied.
- **Pattern verification:** Automatically verify whether changes to the model have 'broken' any of the patterns.
- **Pattern mining:** Automatically detect occurrences of patterns in a model.
- **Three methods of pattern implementation:** Support top-down, bottom-up, or mixed pattern implementation (i.e. create Capsules/ Protocols/ Data from one of the stored implementations for the pattern; associate Capsules/ Protocols/ Data from an existing model with those of a stored implementation for the pattern; or use some combination).
- **HTML publishing:** Publish the set of patterns to an Internet or Intranet website so they can be shared with others who do not have access to the tool.
- **Allow patterns and modeling tool to exist independently:** Use the patterns tool without Rose-RT, or use Rose-RT without the patterns extension.

Convergence of design patterns tools with RTOO modeling tools has great potential; however, there are some significant issues to be addressed.

- **Compression:** A single class may participate in multiple patterns; this issue will complicate the pattern detection, verification, and highlighting functionalities.
- **Rose-RT Classes:** Rose-RT models consist of capsule, protocol and data classes; not all patterns will have structure/ behaviour that can reconcile with these class types. Developers may have other patterns that they want included in the repository, but that don't map well to Rose-RT models (e.g. process/ organizational patterns, or design patterns/ idioms for components of systems not implemented in a language supported by Rose-RT).
- **Abstract nature of patterns:** The mining and verification functions will be very difficult if flexibility of implementation is allowed.

Based on the previous two sections, what appears to be a feasible and appropriate extension to Rose-RT is proposed. It is presented in terms of the major design decisions that must be made.

- Where should the patterns functionality reside?
- What is the recommended initial set of features?
- What other features should be considered?
- What pattern template should be used?
- How should patterns be classified?
- How should pattern implementations be represented?
- How will pattern language members be linked together?
- How will adding patterns to a model in top-down, bottom-up and mixed fashions be implemented?
- How can the tool facilitate keeping track of where patterns are used in a model?
- What will be published to HTML?

5. Conclusions

RTOO modeling tools abstract away the low-level details of systems, allowing developers to focus on the design model – precisely the area where design patterns can be useful. The extension to Rose-RT proposed is meant as a basis for a prototype; the complexity of the problem is such that more than one iteration of prototypes will be needed. This proposal is a starting point for that experimentation.

Initially, the goal of the proposed tool was to extend pattern repository functionality to support Rose-RT models (that can be used to capture both the structure and the underlying source code for pattern implementations). However, as the research progressed it became increasingly apparent that tool support can also be beneficial for describing the inter-relationships between patterns. The proposal thus included repository functionality, and pattern grouping, organization and relationship functionality.

The result is a proposal for a tool that not only makes the application of individual patterns easier and more explicit, but also encourages the systematic application of multiple related patterns. The refinement and implementation of this proposal is a logical next step.

References

[1] Bottomley, M. "A Pattern Language for Simple Embedded Systems". In *Proceedings of PLoP '99*. 1999

[2] Buschmann, F. "Real-time Constraints as Strategies," In *Proceedings of EuroPLoP '98*. 1998.

[3] DeBruler, D.L. "Telecommunications distributed processing patterns." http://www.belllabs.com/people/cope/Patterns/DistributedProcessing/DeBruler/index.html. Avail. 2000.

[4] de Champlain, M. "Patterns to ease the port of micro-kernels in embedded systems." In *Proceedings of PLoP '96*. 1996.

[5] Douglass, B.P. *Doing Hard Time: Developing Real-time Systems with UML, Objects, Frameworks, and Patterns*. Addison-Wesley, Reading, Mass. 1999.

[6] i-Logix Rhapsody Toolset. http://www.i-Logix.com. Avail. 2000.

[7] Jiménez-Peris, R. M. Patiño-Martínez, and S. Arévalo. "Multithreaded Rendezvous: A Design Pattern for Distributed Rendezvous." In *Proceedings of SAC '99*. 1999.

[8] Lea, D. "Design Patterns for Avionics Control Systems." http://g.oswego.edu/dl/acs/acs/acs.html. Avail. 2000.

[9] McKegney, Ross. "Application of Patterns to Real-time Object-oriented Software Design", *MSc. Thesis*. Department of Computing & Information Sciences, Queen's University. July 2000.

[10] Meszaros, G. "A Pattern Language for Improving the Capacity of Reactive Systems" in Vlissides, J.M., J.O. Coplien, and N.L. Kerth. (eds.) *Pattern Languages of Program Design 2*. Addison-Wesley, USA, 1996.

[11] Molin, P. and L. Ohlsson. "Points & Deviations - A pattern language for fire alarm systems." In *Proceedings of PLoP '96*. 1996.

[12] Noble, J., & C. Weir. "Proceedings of the Memory Preservation Society." In *Proceedings of EuroPLoP '98*. 1998.

[13] ObjecTime Corp. http://www.ObjecTime.com. Avail. 2000.

[14] Petriu, D. & G. Somadder. "A Pattern Language for Improving the Capacity of Layered Client/Server Systems with Multi-Threaded Servers." In *Proceedings of EuroPLoP '97*. 1997.

[15] Rational Rose-Realtime Toolset. http://www.rational.com. Avail. 2000.

[16] Schmidt, D.C. "Patterns for Concurrent, Parallel, and Distributed Systems Home Page". http://www.cs.wustl.edu/~schmidt/patterns-ace.html. Avail. 2000.