# Concern-Oriented Software Design with TouchRAM

Matthias Schöttle, Omar Alam, Abir Ayed, Jörg Kienzle

School of Computer Science, McGill University, Montreal, Canada
`mschoettle@cs.mcgill.ca, Omar.Alam@mail.mcgill.ca, Abir.Ayed@mail.mcgill.ca, Joerg.Kienzle@mcgill.ca`

**Abstract** TouchRAM is a multitouch-enabled tool for agile software design modelling aimed at developing scalable and reusable software design models. This paper briefly summarizes the main features of the Reusable Aspect Models modelling approach, highlights the new features of TouchRAM that have been added in the last 6 months, and then describes how the tool is used to incrementally elaborate a software design model. A video that demonstrates the use of TouchRAM can be found here: `http://www.youtube.com/watch?v=l8LMqwwRPg4`

## 1 Introduction

TouchRAM [1] is a multitouch-enabled tool for agile software design modelling aimed at developing scalable and reusable software design models. The tool gives the designer access to a vast library of reusable design models encoding essential recurring design concerns. It exploits model interfaces and aspect-oriented model weaving to enable the designer to rapidly apply reusable design concerns within the design model of the software under development. The user interface features of the tool are specifically designed for ease of use, reuse and agility.

This paper briefly summarizes the main features of the Reusable Aspect Models (RAM) [5] approach that our tool exploits to make design model reuse possible, and highlights the new features that were added to TouchRAM since the last demonstration at AOSD 2013 [4] in section 2. Section 3 describes the software design process using TouchRAM and the last section draws some conclusions.

## 2 Essential Features of RAM

TouchRAM is based on Reusable Aspect Models (RAM) [5], an aspect-oriented multi-view modelling approach that integrates class diagram, sequence diagram and state diagrams. As a result, a RAM model can describe the structure and the behaviour of a software design concern. Currently, however, TouchRAM only provides full support for structural modelling with class and sequence diagrams, and only partial support for protocol modelling with state diagrams.

The most important concepts of RAM that make incremental software design modelling possible are *model interfaces*, *model hierarchies* and *model libraries*.

### 2.1 Design Model Interfaces

Every RAM model has a well-defined *model interface* [2], which makes the concern that is modelled easy to use within other models.

The interface has two parts: the *customization interface* and the *usage interface*. The *customization interface* specifies how a generic design model needs to be adapted to be used within a specific application. To increase reusability of models, a RAM modeller is encouraged to develop models that are as general as possible. As a result, many classes and methods of a RAM model are only partially defined. The idea of the customization interface is to clearly highlight those model elements of the design that need to be completed/composed with application-specific model elements before a generic design can be used for a specific purpose. These model elements are called *mandatory instantiation parameters*.

The *usage interface* is similar to a "classic interface" found in programming languages. It is comprised of all the *public* model elements, i.e., the structural and behavioural properties that the classes within the design model expose to the outside. In other words, the usage interface presents an abstraction of the functionality encapsulated within the model to the user of such a model. It describes how the rest of the application can trigger the functionality provided by the model by instantiating classes and invoking operations on them. At the same time, the usage interface hides the internal details of the design model from the rest of the application, which does not need to know how the functionality is decomposed into classes/methods and how objects interact at run-time to achieve the functionality.

## 2.2 Design Model Hierarchies

RAM allows a modeller to build complex models of any size by putting together many interdependent, simple models. This is achieved through *model dependencies*. To use the functionality provided by a (base) RAM model $B$ within the design of a model $A$, the designer must specify instantiation directives that map the model elements of $B$ that provide the desired functionality to the model elements of $A$ that need the functionality. Using these directives, the TouchRAM tool can, upon request, compose both models to yield a "woven" model that combines the model elements from both designs.

RAM supports two kinds of model dependencies, *model extensions* and *model customizations*. When $A$ *extends* $B$, the modeller's intent is to add additional structural and/or behavioural model elements to $B$ that provide *additional*, *alternative* or *complementary* properties to what already exists in $B$. *The extension model $A$ augments the interface of the base model $B$* with additional structure and behaviour.

A *customization* dependency is useful when a modeller's intent is to adapt the structure and behaviour provided by a base model $B$ to be useful in a specific context. Within a customization model $A$, a modeller *alters* or *augments* existing base model properties to render them useful for a new purpose or to complete partially defined elements. When using customization, all mandatory instantiation parameters of base $B$ need to be mapped to model elements in $A$. *The interface of the customizing model $A$ hides the details of $B$ from the outside world; if any, $A$ exposes only model elements of $B$ produced as a result of customization.*

In RAM, a model $C$ can depend on several models $B_i$, which in turn can depend on models $A_{ij}$, etc., thus creating a model hierarchy. Using the instantiation directives, TouchRAM can incrementally and recursively combine all models of the hierarchy to yield a design model that shows the complete design.

### 2.3 Library of Reusable Design Concern Models

Class libraries offer programmers thousands of classes that provide solutions for common implementation concerns, e.g., common data structures, such as lists, trees, and maps. That way, a programmer does not need to code these classes herself, but simply reuses their behaviour by instantiating them and calling the appropriate methods. The idea of the reusable design concern model library (RDCML) that ships with TouchRAM is similar, but is applied to modelling. Its purpose is to increase modelling productivity by providing models for common design concerns that a modeller can use within an application model with minimal effort when appropriate. Current design models shipped with TouchRAM include workflow execution middleware, network support, and design patterns.

### 2.4 Extensions of TouchRAM since Demo at AOSD 2013

TouchRAM has been significantly extended since the last demonstration at AOSD 2013 [4]. The user interface architecture was completely rewritten to now supports true multitouch. Gesture processors can now be associated with every GUI element, and as a result can handle multiple concurrent gestures. This is especially important in the context of big touch-enabled devices, which allow multiple users to interact with the models simultaneously. It is now, for instance, possible for one user to edit properties of a class, while another user creates associations or moves classes around.

The second major extension relates to the model weaver, which now supports weaving of sequence diagrams. A message view describes the interactions that take place between objects to achieve a specific functionality at the level of abstraction of the concern that is being modelled, i.e., the view only shows the executed method calls that are defined in the current model. With message view weaving, the user can now "expand" the behavioural scenarios to include lower level details, i.e., the subinteractions that take place when methods are called that are defined in lower-level models.

Finally, TouchRAM now also supports the creation and visualization of state views. To this aim, the RAM metamodel had to be extended, and a new view capable of displaying states and transitions was added to the user interface. State diagram weaving, however, is not supported yet.

## 3 Software Design with TouchRAM

Software design modelling with TouchRAM integrates well with modern software design processes, e.g., prototyping or iterative methodologies, as the design and implementation of the application is conducted in phases. First, a simple version of the application is developed that only provides core functionality and services. Detailed and additional functionalities are added in subsequent iterations.

## 3.1 Design Modularization Strategies

**Completeness**: The most important criteria for designing with model hierarchies is *coherent modularization.* Each model specifies a logical design step towards the final design model, and therefore needs to contain *all* the structural and/or behavioural elements pertaining to that logical step. This is important for *internal consistency* of the model: it simplifies reasoning about the design concern as well as making coherent changes to the modelled structure and/or behaviour, if needed. An additional advantage of completeness of individual models is that, by construction, any composed model is therefore also complete.

**Size**: Each individual RAM model should be small, as psychological studies show that the active *working memory* of a human is limited. Examining or building a model of a system induces a certain mental effort on the modeller. This effort is correlated with the model size, and influences the amount of working memory the modelling activity utilizes [6]. When an individual undertakes a mental task (e.g., attempting to analyse a model or answer questions about a model) that exceeds their working memory capacity, errors are likely to occur [8].

**Vertical Design Decomposition**: One way of modularizing a complex software design is to follow a top-down and/or bottom-up strategy, depending on whether the focus is to first elaborate high-level abstractions and functionality, or rather to initially flesh out certain important low-level details of parts of the design. For instance, if detailed functional requirements for the software under development have been elaborated, the initial design phase might begin with deciding on a high-level architecture for the system, and how the required functionality is to be decomposed into subfunctionalities and allocated to different components. On the other hand, if a certain subfunctionality is crucial to the functioning of the software under development, or if reusing an existing software artifact such as a middleware is mandatory or highly cost-effective, then low-level details of a specific required functionality might be designed first in order to determine if the design is actually feasible.

To enable such top-down or bottom-up design, *abstraction* and *information hiding* are key to tame the inherent complexity of a system [7]. Information hiding is the activity of consciously deciding what parts of a software module should be exposed to the outside, i.e., the "rest" of the software under development, and what parts should be hidden from external use. In RAM this is done using customization. Only the structural and behavioural properties that are relevant to use the model are exposed in the model's interface. The design details pertaining to *how* this functionality is provided are not relevant to the user. When using customization, the TouchRAM weaver automatically hides the customized model interfaces of the lower-level models from the user.

**Horizontal Design Decomposition**: When transitioning from one iteration of the software design to the next, it is typical to consider additional functionality. As a result, the core parts of the existing design are complemented with additional functionality, or new components are introduced that take care of providing the additional functionality and the existing design is adapted to integrate the new components. This form of incremental design is supported in TouchRAM using model extensions, which can *adapt and extend* existing *model* *interfaces*. When composing a model that extends another model, the interface of the composed model produced by TouchRAM is formed by the union of the two source model interfaces.

**Concern-Oriented Software Design**: [3] introduces the *concern* as a new, broader unit of model reuse that encompasses all design solutions targeted at solving a specific design problem. To make reuse straightforward, a concern provides an interface that clearly describes the different *variations* of the designs it encapsulates, as well as their impact on non-functional application properties.

Unfortunately, the current version of TouchRAM does not support the creation and editing of such a variation interface by the user. However, the models that are part of the reusable concern library that comes with TouchRAM have already been designed following a concern-oriented philosophy: models that represent variations of design solutions relating to the same design concern have been stored within a common folder. They have been designed using model extensions to share common structure and behaviour when appropriate. The modeller can therefore simply choose the set of RAM models (from within a concern folder) that provide the desired variant of a design concern and customize them to her needs.

## 4 Conclusion

This demo paper summarises how TouchRAM, a multitouch-enabled tool for agile software design modelling, can be used to elaborate a software design model. For more information on the multitouch user interface and the model transformation technology that TouchRAM is based on the interested reader is referred to [1]. The current version of TouchRAM and the reusable design concern model library can be downloaded from `http://www.cs.mcgill.ca/~joerg/SEL/TouchRAM.html`. To use the multitouch features, a TUIO supported multitouch input device must be connected.

## References

1. Al Abed, W., Bonnet, V., Schöttle, M., Alam, O., Kienzle, J.: TouchRAM: A multitouch-enabled tool for aspect-oriented software design. In: SLE 2012. pp. 275 – 285. No. 7745 in LNCS, Springer (October 2012)
2. Al Abed, W., Kienzle, J.: Information Hiding and Aspect-Oriented Modeling. In: 14th Aspect-Oriented Modeling Workshop. pp. 1–6 (October 2009)
3. Alam, O., Kienzle, J., Mussbacher, G.: Concern-Oriented Software Design. In: MoDELS 2013. Lecture Notes in Computer Science, Springer (October 2013)
4. Kienzle, J.: Reusing software design models with TouchRAM. In: AOSD '13 Companion. pp. 23–26. ACM (2013)
5. Kienzle, J., Al Abed, W., Klein, J.: Aspect-Oriented Multi-View Modeling. In: AOSD 2009. pp. 87 – 98. ACM Press (March 2009)
6. Paas, F., Tuovinen, J., Tabbers, H., Van Gerven, P.: Cognitive load measurement as a means to advance cognitive load theory. Educ. Psychologist 38(1), 63–71 (2003)
7. Parnas, D.L.: A Technique for Software Module Specification with Examples. Communications of the Association of Computing Machinery 15(5), 330–336 (May 1972)
8. Sweller, J.: Cognitive load during problem solving: Effects on learning. Cognitive science 12(2), 257–285 (1988)