# Concern-Driven Software Development with jUCMNav and TouchRAM

Nishanth Thimmegowda[1], Omar Alam[1], Matthias Schöttle[1], Wisam Al Abed[1], Thomas Di'Meco[2], Laura Martellotto[2], Gunter Mussbacher[3], Jörg Kienzle[1]

[1]School of Computer Science, McGill University, Montreal, Canada
[2]Polytech Nice-Sophia, Sophia Antipolis, France
[3]Dep. of Electrical and Computer Engineering, McGill University, Montreal, Canada
{Nishanth.Thimmegowda,Omar.Alam,Matthias.Schoettle,Wisam.Alabed}@mail.mcgill.ca, {Thomas.DiMeco,Laura.Martellotto}@gmail.com, {Gunter.Mussbacher,Joerg.Kienzle}@mcgill.ca

**Abstract** A concern is a unit of reuse that groups together software artifacts describing properties and behaviour related to any domain of interest to a software engineer at different levels of abstraction. This demonstration illustrates how to specify, design, and reuse concerns with two integrated tools: *jUCMNav* for feature modelling, goal modelling, and scenario modelling, and *TouchRAM* for design modelling with class, sequence, and state diagrams, and for code generation. For a demo video see: http://www.youtube.com/watch?v=KWZ7wLsRFFA.

## 1 Introduction

In contrast to the focus of classic Model-Driven Engineering (MDE) on models, the main unit of abstraction, construction, and reasoning in Concern-Driven Software Development (CDD) is the *concern* [2]. CDD seeks to address the challenge of how to enable broad-scale, model-based reuse. A concern is a unit of reuse that groups together software artifacts (models and code, henceforth called simply models) describing properties and behaviour related to any domain of interest to a software engineer at different levels of abstraction.

A concern provides a three-part interface. The *variation interface* describes required design decisions and their impact on high-level system qualities, both explicitly expressed using feature models and goal models in the concern specification. The goal models used in CDD are called impact models. The *customization interface* allows the chosen variation to be adapted to a specific reuse context, while the *usage interface* defines how the functionality encapsulated by a concern may eventually be used.

Building a concern is a non-trivial, time consuming task, typically done by or in consultation with a domain expert (subsequently called the *concern designer*). On the other hand, reusing an existing concern is extremely simple, and essentially involves 3 steps for the *concern user*:

1. Selecting the feature(s) of the concern with the best impact on relevant goals and system qualities from the variation interface of the concern,
2. Adapting the general models of features of the concern that were selected to the specific application context based on the customization interface, and

3. Using the functionality provided by the selected concern features as defined in the usage interface within the application.

In general, MDE approaches rely heavily on tool support. Tool support is even more important in the context of CDD, in particular for the concern user:

- When selecting the set of features of a concern that best meets the requirements of the application under development, a concern user needs to be able to perform trade-off analysis between different variations/implementations of the needed functionality. To do that efficiently, a tool is needed that performs real-time impact analysis of feature selections.
- Once a selection is made, a tool is needed that composes the models that realize the selected features to yield new models of the concern corresponding to the desired configuration.
- When adapting the generated concern models to the application context, the concern user must map customization interface elements from the concern to application-specific model elements in the application. Tool support is helpful to ensure that the mapping is specified correctly.
- Once the concern model is customized, a tool can help to ensure that the functionality provided by the concern is correctly used.

This demo illustrates CDD in practice by demonstrating Concern-Driven Development with two integrated tools: jUCMNav and TouchRAM. Section 2 of this paper briefly describes how the two tools were modified to support concerns. Section 3 outlines concern development and concern reuse with the two tools.

## 2 Integration of jUCMNav and TouchRAM

*jUCMNav* is a requirements engineering tool created in 2005 for the elicitation, analysis, specification, and validation of requirements with the User Requirements Notation (URN). jUCMNav combines two complementary views: one for goals provided by the Goal-oriented Requirement Language (GRL) and one for scenarios provided by the Use Case Map (UCM) notation. Recently, jUCMNav was extended to support combined goal and feature modelling and their integrated analysis based on GRL semantics [5]. Over the last two years, jUCMNav was demoed at RE [3,4], the 2013 SDL Forum, and the 2013 iStar workshop.

*TouchRAM* is a multitouch-enabled tool for agile software design modelling aimed at developing scalable and reusable software design models using UML class, sequence, and state diagrams. It exploits model interfaces and aspect-oriented model weaving to enable the concern user 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. TouchRAM was introduced initially at SLE 2012 [1], and later demonstrated at Modularity:aosd 2013 and 2014 [7] and MODELS 2013 [6].

For CDD, jUCMNav and TouchRAM are complementary to each other. jUCMNav covers the requirements modelling side, providing support for feature and goal modelling necessary for the definition of a concern's variation interface. Furthermore, jUCMNav supports scenario modelling with the Use Case Map notation. TouchRAM on the other hand provides support for detailed design

modelling and code generation. The first step in integrating the two tools was to define the concepts of CDD in a metamodel – the CORE (Concern-Oriented REuse) metamodel. It defines:

- *Concern*, which groups together a set of models,
- *Concern Interface*, i.e., the variation interface, the customization interface, and the usage interface,
- *Concern Reuse*, i.e., a concept that is used to store the selected features and the customization whenever a concern is reused, and
- *Feature,* with associations to connect the models that realize the feature and the concern reuses that the feature specifies.

Next, the existing metamodels of jUCMNav and TouchRAM had to be made compliant with CORE. This involved declaring classes in the metamodel of jUCMNav and TouchRAM to subclass classes in CORE. In jUCMNav, the new subclasses of CORE concepts also subclassed existing URN classes. This allowed the same analyses performed on URN models to also be performed on CORE-compliant URN models. Since none of the classes and properties that already existed in the old metamodels had to be removed or modified, the tools are still backwards compatible, i.e., they can still read models created with previous versions of the tools. In addition, the two tools are now file compatible, i.e., it is possible to create a concern and define features for it in jUCMNav and then work with it in TouchRAM, and vice versa.

The current version of jUCMNav can be downloaded from `http://www.softwareengineering.ca/jucmnav`, the current version of TouchRAM from `http://cs.mcgill.ca/~joerg/SEL/TouchRAM.html`

## 3 Concern-Driven Development in Action

The demo at MODELS shows how to first build requirements and design models for the *Authentication* concern with jUCMNav and TouchRAM, and then how simple it is to reuse the *Authentication* concern within a banking application.

### 3.1 Developing a Concern

First, the concern is created in jUCMNav, and the features of the concern are specified. The left picture in Fig. 1 shows that *Authentication* has a mandatory *Authentication Means* feature that may either be *Password* or *Biometrics*. *Biometrics* must at least include *Retinal Scan* or *Voice Recognition*. An optional subfeature of *Password* is *Password Expiry*. If desired, unsuccessful authentication may lead to *Access Blocking* and long idle periods to *Auto Logoff*.

The right picture in Fig. 1 shows how goal models are used to specify the relative impact that each feature has on non-functional requirements and qualities (e.g., security). The model shows that *Retinal Scan* and *Voice Recognition* are the strongest *Security* mechanisms, even stronger than *Password* with *Password Expiry*, *Access Blocking*, and *Auto Logoff*. Once the impacts are specified, jUCMNav allows the concern designer to interact with the feature model and evaluate the impacts of different configurations (sets of feature selections) of the concern (visualized with a color scheme and evaluation values from 0 to 100).

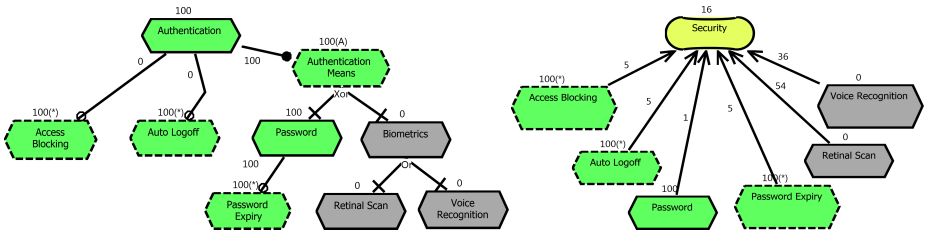


**Figure 1.** Feature and Impact Modelling and Analysis in *jUCMNav*

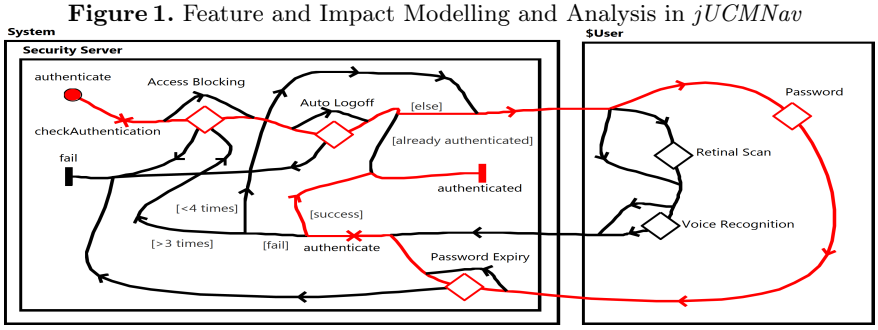


**Figure 2.** Scenario Modelling and Analysis in *jUCMNav*

In jUCMNav, it is also possible to specify scenarios that describe how a user would interact with the *Authentication* concern with the Use Case Maps notation as shown in Fig. 2. The modeller can associate features with path elements in the scenario, which makes it possible to automatically visualize the scenario traversal for a given feature configuration (by highlighting the scenario path in red).

Next, the concern designer uses TouchRAM to create detailed design models and associate them with each feature of the concern. Aspect-oriented techniques such as class merge and sequence diagram advising are used to modularize the structural and behavioural properties of each feature. For instance, if *Authentication* defines a class called `Credential`, the design of *Password* adds a `String` attribute for the password in the class, and the design of *Password Expiry* adds a `Date` attribute that stores when the password was last changed as well as additional behaviour to update this attribute whenever the password is changed.

### 3.2 Reusing a Concern

When a modeller creates a specific application for which *Authentication* is of relevance, the modeller in the role of the concern user opens the *Authentication* concern and selects the desired features from the feature model. While interacting with the feature model, the impacts resulting from the current selection are constantly updated. When a satisfactory selection has been made, TouchRAM composes all design models of the selected features together to produce a detailed design model for this specific configuration. The modeller is then

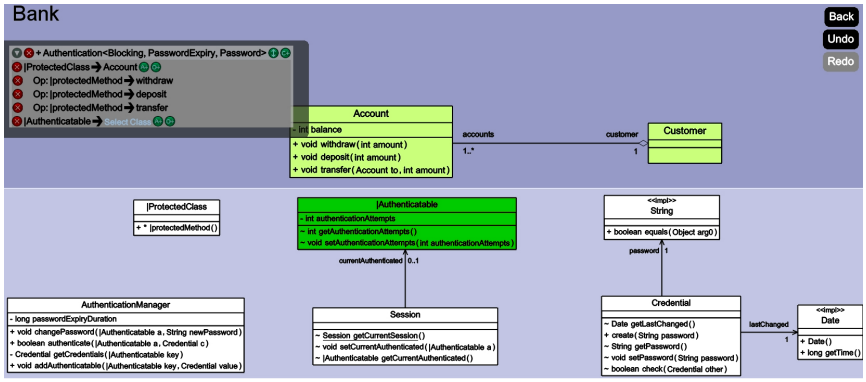


**Figure 3.** Reusing Authentication in *TouchRAM*

presented with a mapping view as shown in Fig. 3 that allows the modeller to customize the *Authentication* concern to her specific needs by establishing mappings between the model elements in the concern and the application model. In our case, the software is a simple Banking application, and the modeller wants to enforce authenticated access to accounts. Therefore, `Authenticatable` maps to the `Customer` class, `ProtectedClass` to `Account`, and `protectedMethod` to `withdraw`, `deposit`, and `transfer`.

Once the customization is completed, the designer of the bank application can instruct TouchRAM to compose the entire application model to yield the combined structure and behaviour of the system. From that, TouchRAM allows the developer to generate executable Java code.

In future work, we are planning to develop several, more complex concerns to empirically validate the integration of the jUCMNav and TouchRAM tools.

## 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. Alam, O., Kienzle, J., Mussbacher, G.: Concern-Oriented Software Design. In: MODELS 2013. LNCS, vol. 8107, pp. 604–621. Springer (October 2013)
3. Amyot, D., Leblanc, S., Kealey, J., Kienzle, J.: Concern-Driven Development with jUCMNav. In: RE 2012, Chicago, USA. pp. 319 – 320. IEEE CS (September 2012)
4. Liu, Y., Su, Y., Yin, X., Mussbacher, G.: Combined Goal and Feature Model Reasoning with the User Requirements Notation and jUCMNav. In: RE 2014, Karlskrona, Sweden. IEEE CS (August 2014)
5. Liu, Y., Su, Y., Yin, X., Mussbacher:, G.: Combined Propagation-Based Reasoning with Goal and Feature Models. In: MoDRE 2014 (August 2014)
6. Schöttle, M., Alam, O., Ayed, A., Kienzle, J.: Concern-Oriented Software Design with TouchRAM. In: Demonstration Paper at MODELS 2013. CEUR Workshop Proceedings, vol. 1115, pp. 1 – 6 (october 2013), `http://ceur-ws.org/Vol-1115/demo10.pdf`
7. Schöttle, M., Alam, O., Garcia, F.P., Mussbacher, G., Kienzle, J.: TouchRAM: A Multitouch-enabled Software Design Tool Supporting Concern-oriented Reuse. In: Companion of Modularity:2014. pp. 25–28. ACM (2014)