McGill UNIVERSITY

Model-Based Reuse of APIs using Concern-Orientation

Get TouchCORE

touchcore.cs.mcgill.ca

Matthias Schöttle | McGill University, Montréal, Canada

Reuse Challenge

Reuse at the Implementation Level

- Essential part of development and reusable code artifacts (e.g., frameworks) are widespread
- Mostly well-maintained and bundled with extensive documentation of various forms
- Can be outdated and difficult to find out which artifact to choose and what the impacts are

Reuse in MDE

- Not very common, possible reasons could be:
  - Difficult to make models reusable
  - Model import and export between tools non-trivial
  - Not many model repositories with reusable models
- Unlikely that existing functionality will become available as models in the near future

Vision

Raise reusable code artifacts to the modelling level and show the artifact from the user's perspective to simplify their use. This builds a bridge between models and code and allows to

- show which features/variations are provided
- show the impacts of features on user goals
- present a subset of the API to the user based on needs

Background

We use Concern-Driven Development (CDD) where the unit of reuse is a concern:

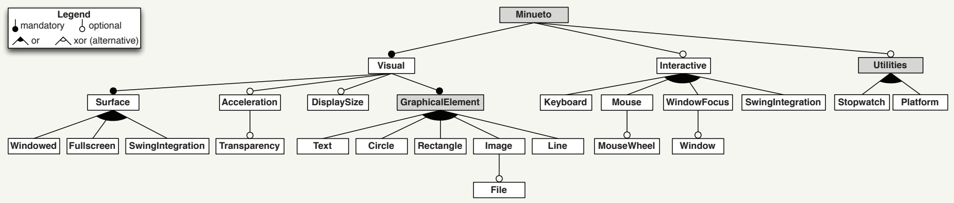
"A concern is a unit of reuse that groups together software artifacts describing properties and behaviour of a domain of interest to a software engineer at different levels of abstraction."

A concern provides three interfaces:

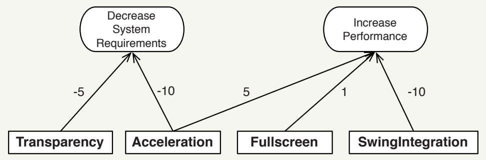
- The variation interface describes required design decisions and their impact on high-level system qualities using a feature model and impact models.
- The customization interface allows the chosen variation to be adapted to a specific reuse context.
- The usage interface defines how the functionality of a concern may be used.

Concernification

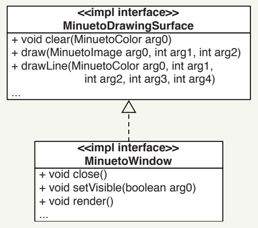
Feature Model of an example framework called *Minueto*



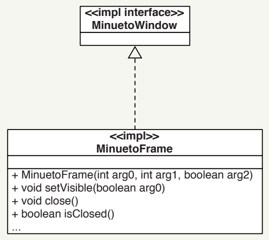
Impact Model showing goals and the feature's impacts



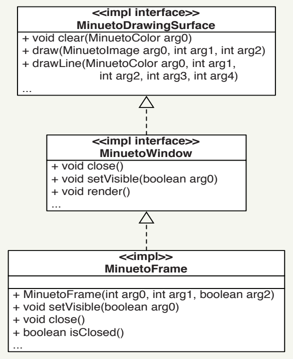
Design model of the feature *Surface*


<<impl interface>>
MinuetoDrawingSurface
+ void clear(MinuetoColor arg0)
+ draw(MinuetoImage arg0, int arg1, int arg2)
+ drawLine(MinuetoColor arg0, int arg1, int arg2, int arg3, int arg4)
...
<<impl interface>>
MinuetoWindow
+ void close()
+ void setVisible(boolean arg0)
+ void render()
...

Design model of the feature *Windowed*



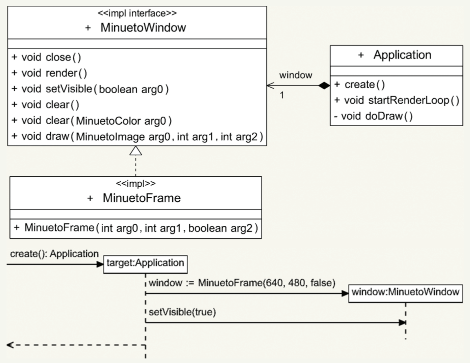
Composed model of features *Surface* and *Windowed*


<<impl interface>>
MinuetoDrawingSurface
+ void clear(MinuetoColor arg0)
+ draw(MinuetoImage arg0, int arg1, int arg2)
+ drawLine(MinuetoColor arg0, int arg1, int arg2, int arg3, int arg4)
...
<<impl interface>>
MinuetoWindow
+ void close()
+ void setVisible(boolean arg0)
+ void render()
...
<<impl>>
MinuetoFrame
+ MinuetoFrame(int arg0, int arg1, boolean arg2)
+ void setVisible(boolean arg0)
+ void close()
+ boolean isClosed()
...

Other Benefits

- Incorporate usage protocols to formally specify the protocol of different classes on how they can be used
- Integrate partial structure and behaviour to force user to provide mappings to application-specific elements
- Use traceability to show which feature elements come from in the composed model

Application showing use of API based on a selection



Tool Support

- We use TouchCORE: Try it out!
- Supports Concern-Orientation using feature and impact models as well as design models
- Import Implementation Classes from existing code (e.g., programming language, frameworks, …) showing only used operations
- Structure and behaviour can connect/use implementation classes
- Generate code for user-specific models

Future

- Concernify larger, frequently used framework(s) to ensure feasibility of approach
- Add automated concernification to extract features and allow user to adjust if necessary
- Perform (user) studies to evaluate approach and automated concernification