Tracking of dependencies between code changes

Lucas Godoy

Agenda

1. Epicea
2. Ring
3. Change Dependency Tracking
4. Implementation
5. Usage scenarios
6. Challenges
7. Q&A

Epicea

- Change model based on events made by Martin
- Supports structural changes, refactorings, system changes (i.e: expression evaluations), undo / redo
- Writes changes automatically to Ombu files using announcements

Epicea (cont.)

- Uses core Ring classes as snapshots of involved code units (package, class, etc)
- Cannot do dependency analysis between changes

Ring

- Among other issues, in the Smalltalk metamodel instance variables are not first-class objects
- Ring is an extensible solution for this problem, providing a common API with the Smalltalk Runtime Model

Change Dependencies Tracking

- Modelled in Ring as the RingC extension
- A RGChangeDependency is a dependency between two RGChange objects
- RGChange objects are extracted from the history representation of the system (RingH extension)

However...

- RingC don’t model refactorings as Epicea does (only CUD operations)
- Adding RingH to the recipe only to use RingC would increase the complexity of Epicea
- We need a simple way to model dependencies between changes

How to do that?

An Ombu entry has two components:

- A content: can be any object
- Tags

In the particular case of Epicea, the content is an event and the tags can be the author, the event timestamp and a reference to the previous entry (called “prior” tag)

Skip pointers

- We can link the entries by adding more prior tags by the affected entity type (class, method, etc) or entity name
- So the entries are linked as in a persistent skip list

Latest Changes Cache

- In order to update fast the skip pointers on each event announcement, we can keep a cache with the latest changes related to each skip pointer
- Cache by type: few items
- Cache by name: several items

Usage scenarios

- Assisted cherry-picking: The user won’t need to collect dependencies manually
- We can implement some filters in the Log Browser more efficiently, since we won’t need to iterate all the entries in the log
- UI: visualization of dependencies in the Log Browser

Some challenges

- References to events vs. references to Ombu entries
- Efficient cache implementation, specially when caching by entity name
- Combined filters implementation

That’s all

Questions?