Large Scale Restructuring

Nicolas Anquetil
Motivation

• Large scale “refactoring”

  – Once in a while, systems need to be completely redesigned
  – e.g. Apparition of internet, cloud, tablets
  – New business opportunity
Programming in the large vs Programming in the small

PROGRAMMING-IN-THE-LARGE
VERSUS
PROGRAMMING-IN-THE-SMALL

Frank DeRemer
Hans Kron

University of California, Santa Cruz

Key words and phrases
Module interconnection, language, visibility, accessibility, scope of definition, external name, linking, system hierarchy, protection, information hiding, virtual machine, project management tool.

long and is easily comprehensible by a single person who understands the intended environment and function of the module.

We argue that structuring a large collection of

We need languages for **programming-in-the-small**, i.e. languages not unlike the common programming languages of today, for writing modules. We also need a "module interconnection language" for knitting those modules together into an integrated whole and for providing an overview that formally records the intent of the programmer(s) and that can be checked for consistency by a compiler.
Refactoring in the large vs in the small

- Current IDEs offer refactorings in the small, at the level of single entities (class, method, variable)
  - Rename
  - Move
  - Extract
  - ...

We want “refactoring” tools in the large

- At the level of system, or multi-entities
- Rename in batch
- Move several entities to create a new module
- Extract a (small) class from a big one
- Complex “refactorings” involving various steps
- ...
Large scale “refactorings”


- Help programmers perform systematic code transformations
“Refactoring” with macros

- package addPlatformRequirement: #'pharo'.
- package addProvision: #'Grease-Core-Platform'

- package addPlatformRequirement: #'pharo2.x'.
- package addProvision: #'Seaside-Canvas-Platform'

• PackageManager 0.58 → 0.59

Applied 19 times

- platform
- package addPlatformRequirement: #'pharo'.
- package addProvision: #'Grease-Core-Platform'

+ platformRequirements
  + ^ #( #'pharo' )
+ provisions
  + ^ #( #'Grease-Core-Platform' )

+ platformRequirements
  + ^ #( #'pharo2.x' )
+ provisions
  + ^ #( #'Seaside-Canvas-Platform' )
“Refactoring” with macros

• Problems:
  – Complex
  – Tedious
  – Error prone

• Proposition:
  – Manually perform the changes once + record
  – Generalize the recorded changes
  – Replay the changes in other locations
Large scale “refactorings”

- Refactoring + intelligent follow-up
  - Refactoring in-the-small
  - + additional checks on the system:
    - Did you notice that … ?
    - Would you like to also … ?
System restructuring (1)

B. Govin (Thales)

- Help programmers re-define the architecture of a system
- Extract component software architecture from a real time, embedded system
System restructuring (1)

• Help programmers re-define the architecture of a system
  - Process and tools to help engineers
  - Work at all levels:
    from packages to individual function calls
System restructuring (1)

- Traditional automatic tools don't work
  - Cohesion/Coupling metrics are useless
System restructuring (1)

- Traditional automatic tools don't work
  - Cohesion/Coupling metrics are useless
- Try using engineers knowledge
  - Identify “core” elements of components
  - + agglomerate elements around cores

• Help programmers restructure the architecture of a system
System restructuring (2)

- Horseshoe approach:
  - From source code to model
  - Work on the model
  - Propagate changes back to the code
System restructuring (2)

- Orion planner
  - Define wished architecture
  - Try things
  - Check validity
System restructuring (2)

- Orion planner
  - Define wished architecture
  - Try things
System restructuring (2)

- Orion planner
  - Try things

Versions of model

![Diagram of Evolution Planning Browser with model versions and classes]

- classes
  - implementations
  - hibernate

- 1 dependency rule(s)
  - hibernate cannot inherit ihibernate
System restructuring (2)

- Orion planner
  - Check validity

Validation (architectural rules)
System restructuring (2)

• Still missing: “Do-it” button