6 Concepts and concept location

- Initiation
- Concept Location
- Impact Analysis
- Prefactoring
- Actualization
- Postfactoring
- Conclusion
Role of concept location

- Concept location finds code snippet where a change is to be made
- Change requests are most often formulated in terms of domain concepts
  - Example: “Correct error that arises when trying to paste a text”
  - the programmer must find in the code the locations where concept “paste” is located
  - this is the start of the change
Partial comprehension of a code

- Large programs cannot be completely comprehended
  - programmers seek the minimum essential understanding for the particular software task
  - they use an as-needed strategy
  - they attempt to understand how certain specific concepts are reflected in the code
- Analogy: visiting a large city
Spelling corner (Merriam-Webster)

• **Intension** \(\text{in-’ten(t)-shən}\)
  – synonym CONNOTATION
  • the suggesting of a meaning by a word apart from the thing it explicitly names or describes b: something suggested by a word or thing — W. R. Inge> an essential property or group of properties of a thing named by a term in logic

• **Intention** \(\text{in-’ten(t)-shən}\)
  – synonyms INTENT, PURPOSE, DESIGN, AIM, END, OBJECT, OBJECTIVE, GOAL mean what one intends to accomplish or attain.
  • INTENTION implies little more than what one has in mind to do or bring about <announced his intention to marry>. . .
Dog as an example

<<intension >>

Dog / Pes / Hund

<<intension >>

Hairy animal with teeth…

<<extensions >>

Fido

Lassie

Buck (in “Call of the wild” by Jack London)
Concept location

• Concept extensions are implemented as code fragments
  • variables, classes, methods, or other
• Programmers finds these code fragments
  • easy in small programs or in the programs that the programmer knows well
  • hard in large programs or programs that the programmer does not know
  • Watchmaker anecdote
Search in the unknown parts of system

1. Understanding the problem
2. Selecting a search strategy
3. Formulating a query
4. Executing the search
5. Analysis of results
Formulating a query

• *Extract* the set of concepts used in the change request

• *Delete* the concepts intended for the communication with the programmers

• *Delete* the concepts that are unlikely to be implemented in the code
  • concepts related to the things that are outside of the scope of the program
  • concepts that are to be implemented in the future.

• *Rank* the remaining concepts by the likelihood that they can be easily located
Example

• Point of Sale system
• Change request is “Implement a credit card payment”
• Identify the concepts
  • “Implement”
  • “Credit card”
  • “Payment”
Example

• Point of Sale system
• Change request is “Implement a credit card payment”
• Identify the concepts
  • “Implement” … communication with programmer
  • “Credit card”
  • “Payment”
Example

• Point of Sale system

• Change request is “Implement a credit card payment”

• Identify the concepts
  • “Implement” … communication with programmer
  • “Credit card” … to be implemented, not in the old code
  • “Payment”
Example

- Point of Sale system
- Change request is “Implement a credit card payment”
- Identify the concepts
  - “Implement” … communication with programmer
  - “Credit card” … to be implemented, not in the old code
  - “Payment” !!! Significant concept, find it in the code
Recognize concept

• Reading code
  – Comments and identifiers
  – Characteristic algorithm (plan)

• Small modification
  – Change the code slightly, execute
  – Throw away this modification! Return the code to the original state!
Concept location methodologies

• Human knowledge
• Traceability tools
• Dynamic search (execution traces)
• Static search
  – dependency search
  – "grep" (pattern matching)
  – information retrieval techniques
GREP Search Technique

• GREP is an acronym for "global regular expression print".  
  – GREP prints out the lines that contain a match for a regular expression.  
  – Programmer iteratively formulates search query and then investigates the results.  
  – If the results are too big to review, programmer either performs further search within these results or reformulates the search query.
Example: Art of Illusion

- 3D modeling studio, written in Java
- More than 600 classes, 100,838 LOC.
- Implement a zooming control
  - currently, the only way to zoom is to enter the zooming value into the specific text box
    - a value of the zoom has to be typed in by the user
    - the default value is 100%.
  - implement zooming control that uses arrow keys
GREP example

• First search: “zoom”
  – The query produced irrelevant 6 lines
• Second search: “scale”
  – returned in 1,544 lines, too large for inspection.
• Third search: “100”
  – default scaling value is 100
  – search the results of the previous search
  – returned 4 lines from the ViewerCanvas.java file.
• Inspection
  – one of the lines is the location
Dependency Search Technique

- Uses Class Dependency Graphs (CDG)
  - extracted from the existing code
- Local functionality
  - consists of concepts that are actually implemented in the module and are not delegated to others.
- Composite functionality
  - as the complete functionality of a module combined with all its supporting modules.
- Determined by reading code and documentation
Functionalities of component X

Concept A

Concept B

Component X

Local responsibility

Composite responsibility
Concept location by dependency search

Find set of starting modules

Select one module

Is the concept implemented in the module?
[Yes] [No]

Is the concept implemented in the composite responsibility?
[Yes] Stop the search
[No]

Find set of the supplier modules

Find set of backtrack modules
Dependency search part 1

• **Start at the** `ModelingApp` **class**
  – concept not contained within its local responsibility

• **The next step: inspect** `LayoutWindow`
  – responsible for constructing the main AOI window
  – composite responsibility contains the concept, but the local responsibility does not.

• **There were clues to search** `ValueField`
  – it implements the text box.
  – concept is not present in the composite responsibility
  – backtrack to the `LayoutWindow` class
Progress of the search

- ModelingApp
  - LayoutWindow
    - SceneViewer
    - ValueField
  - ViewerCanvas
Dependency search part 2

- **SceneViewer class**
  - several functions are responsible for responding to events from the user
  - function `updateImage()` was responsible for repainting the screen
  - we determined that the composite responsibility of this function contained the concept.
  - local responsibility of `SceneViewer` still did not contain the concept

- **ViewerCanvas class**
  - contains the concept
Comparison of the Techniques

• The grep-based
  – depends on the use of naming conventions
  – independent of class structure
  – suitable for explicit concepts only

• The static dependency search technique
  – utilizes the class structure
  – needs correct understanding of composite and local functionality
  – suitable for both explicit and implicit concepts
Example Violet

• Violet
  – Open source UML editor

• Supports drawing UML Diagrams
  – Class diagram, Sequence diagram, State diagram, Object diagram, Use case diagram

• 60 classes and 10,000 lines of code
  – http://sourceforge.net/projects/violet/
GUI of Violet
Change Request

• Record the author for each figure
• This change will make Violet more versatile
  – support for cooperative work
  – the author created a figure
    • author knows the semantics of the figure
• Name of concept: “author”
  – implicit concept extension
    • the extension is not present in the current code
  – belongs to the set of the figure properties
## Status of components (marks)

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank</td>
<td>The class was never inspected and is not scheduled for an inspection.</td>
</tr>
<tr>
<td>Propagating</td>
<td>The programmers inspected the class and found that its composite responsibility contains the concept</td>
</tr>
<tr>
<td>Unchanged</td>
<td>The programmers inspected the class and found that its composite responsibility does not contain the concept</td>
</tr>
<tr>
<td>Next</td>
<td>The class is scheduled for inspection</td>
</tr>
</tbody>
</table>

© 2012 Václav Rajlich

Software Engineering: The Current Practice  Ch. 6
Classes to inspect

- UMLEditor
  - SequenceDiagram Graph
  - UseCaseDiagram Graph
  - ClassDiagram Graph
  - ObjectDiagram Graph
  - StateDiagram Graph
  - ClassRelationship Edge
  - PackageNode
  - ClassNode
  - InterfaceNode
  - NoteNode
  - MultiLineString
  - RectangularNode
  - AbstractNode
Most likely supplier
Next classes to inspect
Wrong way

- UMLEditor
  - SequenceDiagram Graph
  - UseCaseDiagram Graph
  - ClassDiagram Graph
  - ObjectDiagram Graph
  - StateDiagram Graph
  - ClassRelationship Edge
    - PackageNode
    - ClassNode
      - InterfaceNode
      - NoteNode
        - MultiLineString
        - RectangularNode
        - AbstractNode
Possible extension of the search

- UML Editor
  - Sequence Diagram Graph
  - Use Case Diagram Graph
  - Class Diagram Graph
  - Object Diagram Graph
  - State Diagram Graph
- Class Relationship Edge
- Package Node
- Class Node
  - Multi Line String
  - Rectangular Node
  - Abstract Node
- Interface Node
- Note Node
Interactive tool for concept location

Computer

Find set of starting modules

Programmer

Select one module

Is the concept implemented in the module?

[Yes] 
[No]

Is the concept implemented in the composite responsibility?

[Yes]
[No]

[Stop the search]

Find set of the supplier modules

Find set of backtrack modules

© 2012 Václav Rajlich
Software Engineering: The Current Practice  Ch. 6
Example 2: Eclipse 3.3

- Eclipse – Integrated Development Environment for Java

- 15,479 classes, 156,334 functions

- Change request: BugID 172261*: [Actions] When rename a file in one project’s navigator, the other selected file’s name is renamed

*https://bugs.eclipse.org/bugs/show_bug.cgi?id=172261
Dependency search in Eclipse
Dependency search in Eclipse
Dependency search in Eclipse
Dependency search in Eclipse

© 2012 Václav Rajlich  Software En
Dependency search in Eclipse
Results

• Main class: Launcher

• Minimum possible number of classes visited = 4

• Maximum number of classes to be visited = 1076 (1+8+1035+32)

• Developers have to use heuristics or intuition when there are too many suppliers
Advanced GREP Example

• “Anchors” bug in Mozilla*
• Change request
  – Anchors in e-mails are broken (Clicking Anchor doesn't go to target in e-mail)
• Initial knowledge

Knowledge after learning

- Programmer learns more about the domain from the repeated searches

Winning grep query: **mailbox: //**