From goal models to three-layer web-based systems: an exploratory study

Sotirios Liaskos, Marina Daoud Jungblut
York University

John Mylopoulos
University of Toronto
Outline

• Motivation
  – Goal Models
  – From Goal Models to Software

• The Study
  – Process and Product
  – Preliminary Findings, Lessons

• Discussion
Goal Models

Problem

Solution
Goal Models
Goal Models
From Goal Models to Software

• Goal models:
  – Connecting problem statements with solution statements
  – Analysis and automated reasoning opportunities

• But how can they help us build better software?
  – What does the goal model tell us about how to design software?
  – How can the wealth of reasoning and analysis techniques for goal models also be used for analyzing and reasoning about the software system itself?
    • e.g. goal alternatives analysis \(\rightarrow\) software configuration analysis
  – How do elements of the goal model relate to elements of the design/code?
From Goal Models to Software: Methodologies

• The Tropos Project\[^1\]
  – Multi-agent goal models represent problems that need to be solved.
  – A new “System” actor assumes responsibility of leaf level activity that fulfills the goals.
  – Has been refined into a complete development framework.\[^2\]

• In KAOS \[^3\]
  – Sketch of a component-based design.

• Y. Yu et al.\[^4\]
  – Near-isomorphism control and static structures.

What is missing?

• Methodologies:
  – Impose constraints to the development process (structural/architectural, process).
  – Are these constraints justified/natural?
  – Do they fit well with established development cultures?

• Empirical Investigation:
  – If we don’t impose any custom (goal-oriented) methodological constraints.
  – How will the goal model relate to the design?
  – Knowing this would help us design better methodologies.
Exploring the Goal vs. Code Relationship

• Conducted an Exploratory Case Study.
• Methodology:
  – Model requirements using goal models.
  – Develop software based on those requirements.
    • But following a conventional development approach.
  – Explore the result.
• Picked for this study:
  – A web-based, three-layer object-oriented, PHP/HTML-based system.
  – An on-line cart system.
Development Process

• First author
  – Developed the initial goal model.
  – Maintained it throughout the process.
  – Attended the development process ensuring a 3-layer architecture was followed
    • did not make decisions though.

• Two senior undergraduate students:
  – Developed the system that meets the requirements in the goal model.
  – Were told to:
    • Ignore goal structure
    • Focus on leaf level tasks and treat them as a set of acceptance tests.
  – Followed a standard “textbook”\[1,2\] 3-layer object oriented architectural style.
  – Had no prior experience in goal modeling.

Product

View

Application Logic

Data Access

PHP Objects

HTML/JavaScript rendering free-standing PHP code

PHP Objects & Free-standing code

DBMS
Product

View

<table>
<thead>
<tr>
<th>LOC</th>
<th>~1.6K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>0</td>
</tr>
<tr>
<td>Files</td>
<td>18</td>
</tr>
</tbody>
</table>

Application Logic

<table>
<thead>
<tr>
<th>LOC</th>
<th>2,377</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>22</td>
</tr>
<tr>
<td>Files</td>
<td>25</td>
</tr>
</tbody>
</table>

Data Access

<table>
<thead>
<tr>
<th>LOC</th>
<th>1,103</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>13</td>
</tr>
<tr>
<td>Files</td>
<td>15</td>
</tr>
</tbody>
</table>

DBMS

HTML/JavaScript - rendering free-standing PHP code

PHP Objects & Free-standing code

PHP Objects
Analysis

• Goals:
  – Understand relationship between goal decomposition structure and design structure.
  – *Locus* and *type* of code that implements leaf level tasks.

• Amongst Findings and Lessons:
  – Leaf-level tasks relate to code fragments that were written exclusively for those tasks.
  – Goal Variability vs. Software Feature Variability
  – Beyond the User Goal Model
Tasks vs. Code

• Types of Tasks:
  – Those that are performed by humans alone.
    • E.g. “Consult Printed Catalogue”
  – Those that are performed by the machine together with the humans.
    • Let us call them *machine tasks*.
    • e.g. “Post a Comment”
      – Machine offers a textbox.
      – Human enters the comment.
    • e.g. “View Prices”
      – Machine prints the price on the screen.
      – Human reads the price.

• Machine Tasks and Code
  – There are one or more code fragments that were written exclusively for the task.
  – The fragments are located at the view layer.
Task-Exclusive Code
Task-Exclusive Code
Task-Exclusive Code

View Layer

```php
// Code for cart manipulation
```

```php
...<a href='?php echo $clientRoot ? cartControl/viewCart.php'> View Cart </a>
```

```php
viewDetailProductInfo.php

...echo "<TD>Qty: <input type='text' name='qty' size='5'></input> <input type='submit' name='addToCart' size='7' value='Add to cart'></td>";
```

```php
viewProductItemList.php

...echo "<form method='POST'
action=''.$clientRoot."cartControl/CartControl.php'">
...
```

Data Access

```php
...orderItemDA.php
```

Application Logic

```php
...cartControl.php
```

```php
...orderItem.php
```

```php
...sessionControl.php
```
Task-Exclusive Code

• Exists at the View layer.
• May be spread in one or across many modules.
• Renders User Interface elements that allow users perform the task.
• Can disable the task by conditioning the code.
  – The users have no facility to complete the tasks.
  – Possible implementation technique for goal variability.
Goal Variability vs. Software Variability

• **Goal Models:**
  – Express possible ways by which humans can behave.
  – E.g. I may or may not choose to “Read Comments” or “Post a Comment”
    • There is variability in that regard.

• **Software Variability:**
  – E.g. modeled using Feature Models
  – Express variability in software configuration
  – E.g. the widget for “Reading Comments” and “Posting a Comment” is always there.
    • There is *no* variability in the software per-se.
    • If there is variability (e.g. the shop owner disables commenting) it affects but does not relate to the *user* goal model (but the goal model of some other stakeholder).
User Goal Model Enough?
User Goal Model Enough?
User Goal Model Enough?

• The user goal decomposition model relates to a small fraction of the code (in an exclusive way).
  – Yet every line of code must written to satisfy some goal.

• Currently exploring goal models of other stakeholders and how they relate to code:
  – Designer
  – Usability Analyst
  – Database Analyst
  – Store Owner (Customization Agent)
  – ...

Research Methodology?

• How to study the relationship between goal representations and code?

• Biggest Concern: *external validity.*
  – What are classes of systems to be studied separately?

• E.g. to what systems can we generalize our task-to-code association hypothesis?
  – All software systems?
  – All *web-based* systems?
  – All *3-layer* web based systems?
  – All *PHP-based* 3-layer systems?
  – All *small* PHP-based 3-layer systems?
Conclusion

• An exploratory study on developing software based on goal models
• Goal: understand how goals are reflected into the design if no goal-oriented methodology is adopted.
• Picked a web-based system designed following a 3-layer pattern.
• Experiences, Findings and Lessons:
  – Leaf level tasks associate with code that is written exclusively for allowing users fulfill the tasks.
    • Exclusive code is concentrated at the View layer of the 3-layer design.
    • Yet, it may be scattered in one or across modules.
    • Code in other layers that relates to the task may be shared (not exclusive).
    • Modular decomposition does not necessarily follow goal decomposition.
  – Goal Variability vs. Feature Variability
  – The role of other stakeholders → cannot restrict ourselves to the user.
Thank You!
(questions?)