
Problems and Opportunities for Model-Centric vs. Code-Centric Development: A Survey of Software Practitioners

Presented June 2010

**Professor: Dr. Timothy C. Lethbridge
Students: Andrew Forward, Omar Badreddin**

University of Ottawa

Some Key Objectives of the CRuiSE group:

Better understand the roots of software system complexity

- From the perspective of all stakeholders
 - Software engineers, users, configurers, etc.
- What actions and activities drive complexity?

Better understand the successes and failures of existing tools, when it comes to complexity reduction

- E.g. Modeling in general, UML

Empirical Study: Attitudes about Modelling

Objective: Understand how software engineers

- think about modelling
- ... and **do** modelling
- ... so we can develop better tools

Types of questions

- What is a model?
- How do you create / modify software models?
- How do you learn about the design of software?
- What modelling notations do you use?
- What problems do you experience when
 - Modelling
 - Not modelling

Empirical Study: Overview

113 Participants
18 questions

Demographics

- Average 14 years of SE experience
- 56% from Canada and United States
- 24% Other regions include United Kingdom, Other parts of Europe, India, Pakistan, Australia, Mexico and Singapore
- Remainder did not indicate location

Empirical Study: Overview

Roles in software engineering

- Modelers: 41%
- Generate code from models: 14%
- Real-time developers: 17%

Data: Q1 Which of These is a Model?

Entity that might be a model	N	% Str. Disagree (1)	% Disagree (1 + 2)	% Agree (4 + 5)	% Str. Agree (5)
Class Diagram	112	0.9	2.7	88.4	48.2
UML Deployment Diagram	111	1.8	5.4	77.5	36.0
Use Case Diagram	112	1.8	9.8	82.1	33.9
Picture By Drawing Tool	111	1.8	7.2	85.6	25.2
Textual Use Case	113	2.7	10.6	78.8	30.1
Picture By Hand	113	4.4	8.8	78.8	29.2
Whiteboard Drawing	112	3.6	9.8	57.1	22.3
Source Code	111	13.5	38.7	46.8	23.4
Source Code Comment	112	11.6	41.1	33.9	9.8

Values range: Strongly Disagree (1), Disagree (2), Neutral (3), Agree (4), to Strongly Agree (5).

Implication: Integrate source and model

Data: Q2 How Participants Model

Medium or method used to model	N	mean	s.d.	% Never (1)	% (1 + 2)	% (4 + 5)	% Always (5)
Whiteboard drawing	111	3.2	1.1	5.4	33.3	45.0	9.9
Diagramming tool (e.g. Visio)	111	2.9	1.2	15.3	42.3	36.9	9.9
Word processor / text	112	2.8	1.1	7.1	45.5	26.8	8.9
Word of mouth	111	2.8	1.1	12.6	42.3	27.0	8.1
Handwritten material	111	2.6	1.1	13.5	51.4	22.5	4.5
Comments in source code	111	2.5	1.2	27.0	51.4	21.6	5.4
Modeling tool/CASE	112	2.4	1.4	38.4	58.9	29.5	10.7
Drawing software	111	2.1	1.0	29.7	72.1	12.6	2.7

Note. Values range from Never (1), Sometimes (2), Moderately often (3), Very often(4), to Always (5).

Implication

- Improve modeling tools to increase adoption
- Provide a way to brainstorm electronically

Data: Q3 How Participants Learn About Software

Refer to material created by/as	N	mean	s.d.	% Never (1)	% (1 + 2)	% (4 + 5)	% Always (5)
Word of mouth	112	3.4	1.1	4.5	22.3	54.5	17.0
Word processor / text	110	3.3	1.1	2.7	30.0	48.2	10.0
Diagramming tool (e.g. Visio)	111	3.1	1.1	9.9	32.4	42.3	9.0
Whiteboard drawing	110	3.0	1.1	9.1	34.5	41.8	5.5
Comments in source code	112	2.9	1.2	11.6	42.0	30.4	10.7
Drawing software	109	2.6	1.0	14.7	57.8	13.8	3.7
Modeling tool/CASE	111	2.5	1.4	33.3	55.9	31.5	8.1
Handwritten material	109	2.4	1.1	23.9	56.0	20.2	3.7

Note. Values range from Never (1) to Always (5).

Implication

- Help create live documents from models

Data: Q11 How Modeling Tools Are Used

Activity	N	mean	s.d.	% Never (1)	% (1 + 2)	% (4 + 5)	% Always (5)
Developing a design	64	3.3	1.2	6.3	26.6	48.4	17.2
Transcribing a design into digital format	64	3.1	1.3	14.1	32.8	39.1	14.1
Prototyping a design	64	2.7	1.3	20.3	53.1	32.8	12.5
Brainstorming possible designs	64	2.6	1.2	18.8	54.7	23.4	10.9
Generating code (code editable)	63	2.2	1.2	36.5	65.1	17.5	6.3
Generating all code	64	1.8	1.2	65.6	76.6	14.1	4.7

Note. Values range from Never (1) to Always (5).

Implication

- Improve brainstorming and initial quick early design
- Code generation

Data: Q13 Most Desired Characteristics of a Modeling Tool

Attribute / Ability to	N	Rank	% top 4	% bottom 4
Communicate to others	89	1	78.7	16.9
Readability	89	2	68.5	20.2
Ease and speed to create	89	3	55.1	36.0
Ability to analyze	89	4	55.1	33.7
Collaborate amongst developers	89	5	43.8	38.2
Ability to view different aspects of a model	89	6	40.4	42.7
Generate code	89	7	23.6	70.8
Information density	88	8	17.0	72.7
Embed parts of model in documentation	89	9	13.5	82.0

The % top 4 show the % of participants that listed the attribute in their top 4. Similar for % bottom 4.

Implication

- Focus on ease of use of models
- Generation may be low because users are not convinced it works

Data: Q14 Easier with Code (top) vs. Model (Bottom)

Available activities	N	mean	s.d.	% Much easier in Models (1)	% Easier in Models (1 + 2)	% Easier in Code (4 + 5)	% Much easier in Code (5)
Fixing a bug	90	3.2	1.5	21.1	28.9	43.3	25.6
Creating efficient software	92	3.1	1.4	16.3	35.9	43.5	21.7
Creating a system as quickly as possible	92	3.0	1.5	23.9	46.7	42.4	23.9
Creating a prototype	92	2.9	1.5	26.7	43.0	32.6	22.8
Creating a usable system for end users	92	2.7	1.3	26.1	42.4	22.8	10.9
Modifying a system when requirements change	91	2.5	1.4	34.1	54.9	24.2	13.2
Creating a system that most accurately meets requirements	91	2.2	1.3	42.9	67.0	19.8	8.8
Creating a re-usable system	92	2.2	1.3	44.6	63.0	15.2	9.8
Creating a new system overall	92	2.2	1.3	43.5	68.5	20.7	7.6
Comprehending a system's behavior	89	2.0	1.3	51.7	71.9	15.7	5.6
Explaining a system to others	92	1.7	1.1	61.1	81.8	7.6	6.5

Values range: Much easier in a model-centric approach (1), to much easier in a code-centric approach (5).

Data: Q15 Problems with Model-Centric Approach

Potential problems	N	mean	s.d.	% Strongly Disagree (1)	% Disagree (1 + 2)	% Agree (4 + 5)	% Strongly Agree (5)
Models become out of date with code	92	3.8	1.2	7.6	16.3	68.5	37.0
Models cannot be easily exchanged	91	3.3	1.3	15.4	26.4	51.6	17.6
Modeling tools are 'heavyweight'	92	3.1	1.2	10.9	31.5	39.1	12.0
Code generated from a modeling tool not of the kind I would like	91	3.0	1.4	18.7	39.6	38.5	16.5
You cannot describe the kinds of details that need to be implemented	89	2.8	1.3	23.6	43.8	36.0	7.9
Creating and editing a model is slow	92	2.7	1.2	17.4	43.5	22.8	12.0
Modeling tools change, models become obsolete	92	2.7	1.2	22.8	44.6	32.6	5.4
Modeling tools lack features I need	89	2.6	1.1	19.1	44.9	21.3	5.6
Modeling tools hide too many details that would be visible in the code	92	2.6	1.1	19.6	44.6	23.9	1.1
Modeling tools are too expensive	90	2.6	1.3	26.7	46.7	26.7	6.7
Modeling tools do not allow me to analyze my design how I would want	90	2.5	1.3	28.9	51.1	25.6	6.7
Org. culture does not like modeling	92	2.5	1.2	31.5	48.9	23.9	4.3
Semantics of models different from those of programming Languages	90	2.4	1.3	31.1	56.7	23.3	8.9

Implications of Problems Noted

Focus on either

- Generating all code
- Unifying code with model

Focus on quick and easy modeling

Data: Q16 Problems with Code-Centric Approach

Potential problems	N	mean	s.d.	% Strongly Disagree (1)	% Disagree (1 + 2)	% Agree (4 + 5)	% Strongly Agree (5)
Hard to see overall design	94	3.8	1.1	4.3	13.8	66.0	35.1
Hard to understand behavior of system	94	3.6	1.1	4.3	19.1	60.6	21.3
Code quality degrades over time	92	3.4	1.3	9.8	28.3	55.4	25.0
Too difficult to restructure system	93	3.4	1.2	8.6	22.6	51.6	17.2
Buggy code changes	93	3.4	1.2	9.7	22.6	50.5	18.3
Changing code takes too much time	94	2.8	1.2	20.2	39.4	27.7	8.5
Our language leads to complex code	94	2.5	1.2	26.6	51.1	20.2	8.5
More skill is required than is available to develop high quality code	91	2.5	1.2	29.7	53.8	22.0	6.6
Our languages are not expressive enough	91	2.1	1.2	46.2	64.8	14.3	5.5
Organization culture does not like the code-centric approach	92	1.9	1.2	58.7	72.8	14.1	4.3
Our programming language is likely to become obsolete	93	1.9	1.1	51.6	75.3	9.7	3.2

Values range from Not a problem (1) to Terrible problem (5).

We can have the best of both worlds: Our Research Direction: Umple

Adds associations and attributes to programming languages

- Java, PHP
- Others being developed: Ruby, C++

Unifies modeling and programming

- Modelers and programmers can both work the way they are used to

Works with Eclipse and some other tools for diagram generation and code generation

- Xtext binding
- Being incorporated into Papyrus

Stand-alone code-generator and diagram is online at

- <http://cruise.site.uottawa.ca/umpleonline/>

Declaration of classes and attributes

```
class Student
{
  studentNumber; // defaults to String
  String grade;
  Integer entryAverage; // implemented as int
}
```

Associations

```
class Student { id; name; }
```

```
class Course { description; code; }
```

```
class CourseSection {  
  sectionLetter;  
  1..* -- 1 Course;  
}
```

```
association {  
  * CourseSection;  
  * Student registrant;  
}
```

Associations continued

```
class A {1 -- * B;}  
class B {}
```

Is semantically identical to

```
class A {}  
class B {}  
association {1 A -- * B;}
```

Methods can Manipulate the Above Association Using

Accessing the association end at class A

- `public B getB(int index)`
- `public List getBs() /* unmodifiable */`
- `public int numberOfBs()`
- `public boolean hasBs()`
- `public int indexOfB(B aB)`
- `public B addB() /* creates new B */`
- `public boolean addB(B aB)`
- `public boolean removeB(B aB)`

Accessing the association end at class B

- `public A getA()`
- `public boolean setA(A aA)`
- `public void delete()`

State Machines

```
class A {  
  sm {  
    S1 {  
      e1 -> S2;}  
    S2 {  
      e2 -> S1;}  
    }  
  }  
}
```

The API for methods in Java is:

- enum Sm { S1, S2 }
- setSm(Sm.s1)
- public boolean e1()
- public boolean e2()

Selected patterns

```
class University {  
    singleton;  
    String name;  
}
```

Incremental use of Uml

Incremental 'Umlification' of a legacy system

- Start with the target language
 - Hopefully with a good set of test cases
- Refactor to Uml until the model is recovered
- We did this with the Uml compiler and several other systems

Incremental development

- Start with a 'pure' UML model represented as Uml
- Generate stub/driver test and UI code
- Incrementally add algorithmic methods

Model versioning accomplished by any version control tool

References for more information

Detailed data from the survey

<http://www.site.uottawa.ca/~tcl/gradtheses/forwardphd/>

Umple home page

<http://cruise.site.uottawa.ca>