Empirical Studies into Modelling in Software Development in the Age of Big Data and AI

Michel R.V. Chaudron
Professor in Software Engineering
Joint Computer Science and Engineering dept.
Chalmers and Gotenburg university, Sweden
chaudron@chalmers.se

Outline of talk

Introduction

• Modeling in Software Development
  - Motivation
• Description of Practice of Modeling

Empirical Research in Software Design and Modeling

• Topics include:
  - illustrations of various of modeling-related ‘big’ data
  - illustrations of machine learning in this field

Summary & Conclusions

Questions are welcome
Joint Dept of Computer Science
Chalmers and Gotenborg Univ`s: Divisions

Computing Science
Formal Methods
Functional Programming
Computer Engineering

Information Security
Interaction Design
Software Engineering
Networks and systems

Software Engineering Division Staff
SOFTWARE ENGINEERING
CONFERENCES IN GOTHENBURG

Open Source 2016
REFSQ 2016
ICSA 2017
Mensura 2017
IFIPTM 2017 Int. Conf. on Trust Management

+ yearly Lindholmen Software Development day (600+)

Software Engineering in Africa (SEiA)

• Software Engineering in Africa workshop at ICSE
• SIDA BRIGHT project (2015-2010)
  – Supervise 10 lecturers in Uganda towards their PhD degree.
  – Yearly summerschools in East-Africa

• Tentative plans:
  African Summerschool on Software Engineering 2020
Introduction: Research Interests

• What are the pay-offs of investing in architecture/design/modeling?
  Fewer defects?
  Cheaper maintenance? ...

• Extract design knowledge from software repositories (models, text, ...)

• Analysis and Reasoning about Quality Properties of System Architectures
  – Automating Architecture Design Optimization by Genetic Algorithms
  – Maintainability, Technical Debt

Modeling in Software Engineering

Long history of modeling in software engineering:


Models of software capture structure and behaviour
Modelling History

Example 4+1 Views model

**Structure view:**
class/component-diagram

A → B

C   D

**Behaviour view:**
Sequence diagram

A → B → C → D

BC/WC e2e-response times, freq.

**Development view**
file ownership
Config. Mngnt view
versioning policies
...

**Use cases**

**Deployment view:**
physical model + mapping

TCP/IP over Ethernet

bandwidth, availability

Juha-Pekka Tolvanen, CEO Meta-Case, Keynote at Code Generation 2014:
The business cases of modeling and generators
Fake UML News – Urban Myths

UML is a notation, not a method –

UML can only be used with object orientation

UML is not a process
  i.e. UML is not tied to waterfall or RUP or SCRUM or …

Only when used for code generation, is modeling effective

Michel Chaudron is a UML advocate

---

The Importance of Architecture

Cost of SW fault repair costs

Architecting/design determines 90% of the costs and risks

≥ 100x

Analyse before you build.
Increasing Complexity of Software

Context factors
- Complex system with many modules and dependencies

‘Management Design Complexity’

Modeling is inevitable

Motivation
Why should we care about modeling software design?

Pro-Modeling-camp: “I can not imagine we could develop our systems without any modeling.”

Anti-Modeling-camp: “Models are useless. Code is the only truth.”
Modeling & Design in Civil Engineering

Building Information Modeling Opportunities

Context factors
- Complex endeavor
- Large project team
- Multidisciplinary expertise
- Large scope over time

‘Knowledge Management’, Planning & Coordination

Summary of Arguments in favour of Modelling in Software Development

+ Communicating / Coordination
+ Analysing / Predicting
+ Understanding / Structuring
+ Guiding Construction
+ Blueprint for Production
Some arguments against modeling

- Creating models takes too much time
  - we can do the same without models
- Maintaining models takes too much time
- Notation is too complex
- Notation is not expressive enough
- Our customer does not ask for it
Not always rational ...

We're going to try something called Agile Programming.

That means no more planning and no more documentation. Just start writing code and complaining.

I'm glad it has a name.

That was your training.

Modeling & Documentation in Agile Development

• Agile principles: working software over comprehensive documentation

Survey under 75+ agile developers

Tending towards: we need a bit more

Modeling is compatible with agile development

Christoph J. Stettina and Werner Heijstek, Necessary and Neglected? An Empirical Study of Internal Documentation in Agile Software Development Teams 29th ACM Int. Conf. on Design of Communication, Pisa, Italy
Informal (whiteboard) drawing

Easy to make.
But not so easy to:
- Change / update
- Search
- Link / Trace
- Analyse / Check

To Model or Not to Model?

That is NOT the question
Better Questions

- Which people in my development organization benefit from design models? Through which task/activity?
- Which information should I include in my models?
  - Who ‘owns’ this information?
- How can we integrate modeling into our 
  - development process?
  - who should be responsible for creating and maintaining models?
  - development toolchain?

Understanding the practice of modeling in software projects
The Model-based SE Spectrum

Model driven architecture: Principles and practice, Brown, A.W., Software and Systems Modeling, 2004

Styles of UML modeling

More effort ⇒ More expensive

Recipe for construction
Evolution of Models during a project

<table>
<thead>
<tr>
<th></th>
<th>Ideation</th>
<th>Externalization</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Different stages have different needs on formality
Yet tasks are connected:
‘smooth’ handover over design representation is needed
Target audience of the models?

<table>
<thead>
<tr>
<th>Ideation</th>
<th>Externalization</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main audience: people</td>
<td>cannons</td>
<td>computers</td>
</tr>
</tbody>
</table>

Ok with informal spec (has context, domain & background knowledge)  
Needs formal and complete spec

Empirical Research in Modeling in Software Design
Empirical Research
... is a way of gaining knowledge by means of observation or experience.

<table>
<thead>
<tr>
<th>Theory</th>
<th>Practice</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE body of knowledge</td>
<td>People/Skills, Processes, Methods, Technology, Artefacts</td>
</tr>
</tbody>
</table>

Observation
Prediction/Test
Validation

Topics: CMM, OO, Agile, MDA, ...  
Methods: Experiment, Case study, Survey, Project Repositories ...

How to study this?

- Communicating / Understanding
- Coordination
- Analysing / Predicting
- Structuring
- Guiding
- Production

- Understandability of designs?
- Misunderstandings??
- Software Quality
  - Modularity
  - Defects
- Productivity / Efficiency

Software Quality
The challenges with Industrial (‘Real’) Data

1. Difficult to get access to
2. Difficult to publish / share (replicate)

DIY - Let’s Build Our Own Dataset

Können wir das schaffen?!

2 main pillars for Big Data

Collecting data

Analysing data

That looks doable
Collecting Big Data

Idea: This is a ‘Search’-problem.
Probably Google can do this for us.

Attempt 0.1 & 0.2:
1. Search for (filetype) .xmi using Google
   Unsuccessful: incredibly much that is not about ‘XMI’ for UML
   Manual filtering is very time-consuming
2. Search for ‘class diagram’ via Google Image Search
   + : many images can be found, not all are class diagrams
   - : very high variation in quality
      tutorial/lecture-slides (on notation), student-projects, ...
Many many false positives.

Still many false positives.
After much manual filtering: collection of about 1000 diagrams.
Classifier for recognizing UML Class diagrams

- **Input:** arbitrary image (.jpeg)
- **Output:** Yes/No (=UML Class Diagram)
- Initial filtering done by rules
  - Size (no icons), colours
- 1300 Diagrams, of which 650 UML CD and 650 other (incl. Seq.D, ER, charts, ...)

With my students:
- Ingimar Samúelsson
- Jóel Hjaltason

<table>
<thead>
<tr>
<th>Features include:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contours &amp; Shapes</td>
</tr>
<tr>
<td>Horizontal and Vertical Lines</td>
</tr>
<tr>
<td>Rectangle-H-W-ratio</td>
</tr>
<tr>
<td>Connected rectangles</td>
</tr>
<tr>
<td>Rectangles with ‘divider-line’</td>
</tr>
<tr>
<td>NON-Hor/Ver lines</td>
</tr>
<tr>
<td>% of area used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forrest</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Automated Extraction of Models from Images

Input: .jpeg, .png

Output: XMI for the class model

Milestone: IMG2UML Tool
Is Reverse Engineering a solution?

• Automatically generated from source code, hence always up-to-date!
• But, ...

Reverse Engineering of a small system

Clearly different from forward designed UML designs (o.a. in size, layout, detail, naming, ....)
Classifier for UML Class Diagram Styles using Machine Learning

- Forward vs Reverse Engineered Diagram Classifier
- 16 Features
  - mostly related to ‘parameters’ of methods in diagram
- Sample of 999 class diagrams
  - Reverse: 806
  - Forward: 193

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forrest</td>
<td>90.74</td>
<td>0.95</td>
<td>0.93</td>
<td>0.94</td>
</tr>
</tbody>
</table>

The Quest for UML in Open Source Projects
-- Mining GitHub --

Some statistics

July 2015

1. GHTorrent
2. GitHub

Data collection

Potential UML file list

Filter UML files

UML Image Filter

Textual Filter

Validation

Analyse result

CSVAnalyzer MySQL

Query

Extract Meta-data

12,850,000 non-forked repos

Identified 93,000+ UML files
24,000+ repositories

Collecting data

---

Surprise 1

UML models appear in many formats

<table>
<thead>
<tr>
<th>File type</th>
<th>.uml</th>
<th>xmi</th>
<th>svg</th>
<th>png</th>
<th>jpeg</th>
<th>bmp</th>
<th>jpg</th>
<th>gif</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share</td>
<td>34%</td>
<td>4%</td>
<td>1%</td>
<td>37%</td>
<td>&lt;1%</td>
<td>&lt;1%</td>
<td>10%</td>
<td>13%</td>
</tr>
</tbody>
</table>

NB. We did not yet look into: pdf, word, ppt, specific UML-CASE tooling-formats (.uml, .umple, ...)

Collecting data

image formats

60%
Projects grouped by number of UML files

<table>
<thead>
<tr>
<th>#UML files</th>
<th>1</th>
<th>[2-9]</th>
<th>[10-99]</th>
<th>[100,∞)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Repositories</td>
<td>14 749</td>
<td>8 567</td>
<td>1 333</td>
<td>68</td>
</tr>
</tbody>
</table>

![Bar chart showing projects grouped by number of UML files](chart.png)

Example analysis: size of class diagrams

![Distribution of diagrams by number of classes](distribution.png)
Countries of Projects that use UML

Distribution of participant by continents

- Europe: 54%
- North America: 12%
- South America: 11%
- Asia: 3%
- unknown: 1%
- Oceania: 1%
- Africa: 1%

Opportunity

This dataset enables many types of empirical studies

Are you looking for models?

We have collected

35 000+ class diagrams

from 24 000+ open source projects at GitHub

These diagrams can be traced back to the projects, hence it is possible to find associated project data such as source code, commit messages, commit-dates, and much more.

We would love to hear from you:
- Would you like to use the dataset in your research?
- Which research questions do you recommend us to look at?

Dataset

http://oss.models-db.com/

Research Group

Contact person: Truong Ho Quang
truongh@chalmers.se
Research Challenge

Value/Utility → use → discovery → tools

Patterns?
Which ones to look for?

Enrich via labelling:
- Reverse vs Fwd design?
- Student project?
- Domain model vs design model?

+/- 100,000

Knowledge Discovery Challenge

• Huge variety of data:
  – Graphs: UML diagrams
  – Source code
  – Text: SAD, bug-reports, ….

Currently looking beyond models and code, also into Software Architecture/Design Documents
Understanding the Modeling Process

Series 1: Code commits
Series 2: Documentation commits
Series 3: UML/model commits

Work in Progress – Some insights so far:
- very little updating of models
- no quality assurance applied to models
Approach to Modeling is contextual

Project has Context

Stakeholder has Goal

SE Approach

SE-Process drivesSE-Practices

SE-Tools drives SE-Practices

Approach to UML

Modeling Process drives Modeling Practices

Modeling Tools

Approach to Knowledge Management

Approach to Documentation

Approach to Implementation

Is there a DNA for Software Design?

“Sequence motifs are short recurring patterns in DNA that are presumed to have biological function.”


Image taken from bio.miami.edu
Challenge: Uncovering the hidden structure of designs

Rebecca Wirfs-Brock: Software designs are built from building blocks that have stereotypical responsibility-roles

Controller
Coordinator
Interfacer
Service Provider
Information Holder

Definition of Role Stereotypes

There are many ways to understand the nature of a class, but I start by looking at its name. Antonyms are names that match a person’s occupation—Joe Strong is a weight lifter, Sue Snow is a ski instructor. Repeatedscenarios or a class’s name can suggest its role in a design. I respect class names to be antonyms. For example, in Java, a StringTokenizer picks apart segments of a string, and the ClassLoader loads classes. But names aren’t always illuminating, so I also scan a class for invariant-revealing method names that suggest the class’s behavior. Of course, the definitive source is always the code, but I shouldn’t have to pore over details just to get the gist of a class.

In this column, I introduce several characteristics I associate to classes when trying to understand their nature and purpose. I hope you find these useful guidelines and not mere quibbles. (I’m both delighted by and dismayed of a word that has definitions with opposite meanings. The technical term for such a beast is antonym.)

Role stereotypes

- Information holder: an object designed to know certain information and provide that information to other objects.
- Structure: an object that maintains relationships between objects and information about those relationships. Complex structures might pool, collect, and maintain groups of many objects; simpler structures maintain relationships between a few objects. An example of a generic structure is a Java HashMap, which relates keys to values.
- Service provider: an object that performs specific work and offers services to others on demand.
- Controller: an object designed to make decisions and control a complex task.
- Coordinator: an object that doesn’t make many decisions but, in a role or mechanical way, delegatures work to other objects. The Mediator pattern is one example.
- Interface: an object that transforms information or requests between distinct parts of a system. The edges of an application contain two-interface objects that interact with the user and external interface objects, which communicate with external systems. Interfaces also exist between subsystems. The Facade pattern is an example of a class designed to simplify interactions and limit clients’ visibility of objects within a subsystem.
Automated Classification of Role-Stereotypes by Machine Learning

Arif Nurwidyanto
Monash University
Melbourne, Australia
arif.nurwidyanto@monash.edu

Truong Ho-Quang
Chalmers | Gothenburg University
Gothenburg, Sweden
truonghq@chalmers.se

Michel R.V. Chaudron
Chalmers | Gothenburg University
Gothenburg, Sweden
chaudron@chalmers.se

Accepted for publication in EASE2019

- Build a ground truth for 3 open source projects
  - K9, BitCoinWallet, Home3D
- Features used:
  - source code (e.g. names), design metrics, network-metrics
Mining graph-patterns in Software Designs

Through labelling of roles, we find recurring patterns in the design

These patterns represent typical collaborations between responsibility-stereotypes
Distribution of Role-Stereotypes

Role-Stereotypes Differ Significantly in their Design Metrics
Role-stereotypes-based visualization of software design

Ratio of different role-stereotypes within a component gives information about its main type of responsibility

Possible uses of Role-Stereotypes

Role-Stereotypes aid in:
- Program understanding
- Design summarization
- QA for Architecture/Design
- Tailoring test-generation/coverage/...
Challenge: Mining graph-patterns in Software Designs

Challenges:
- Stereotypes are ‘idealized’.
  In practice, classes are not ‘ideal’ designs.
Documentation Challenge: Find & Navigate

- Where is the information that I need?
- Many ‘places’
  - GitHub, Project-PC, ...
- Poor searchability
- Poor navigability
- Is it up-to-date?

Knowledge Management in Software Development & Maintenance

Reduces load on creating & maintaining documentation

- Extraction
  - NLP, image processing
- Linking
  - ontologies
- Up-dating
- Navigate / Present
Modelling is part of Knowledge Management

Documentation Paradox

• Value of documentation as a function of time (=familiarity with the system)

Novices need it, but cannot make it. Experts can make it, but do not need it.
Concluding Remarks

• We have poor understanding of the consumption/use of models in practice
  • Mismatch between tools and tasks

• Engineering reality:
  • Modelling styles are manifold, driven by purpose & context
  • Lack of adoption of hygiene practices

• Next step / challenges
  • Knowledge-discovery
    • Heterogeneity of data
    • Absence of context / noise
  • Requires community effort
Does the use of modeling improve software quality?

A large number of developers indicated the use of UML improves understandability and modularity.
Summary of Evidence on Modeling

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>Confirmed</td>
</tr>
<tr>
<td>Analysis</td>
<td>No evidence</td>
</tr>
<tr>
<td>Structuring</td>
<td>No evidence</td>
</tr>
<tr>
<td>Guiding</td>
<td>Confirmed</td>
</tr>
</tbody>
</table>

Main hurdles:
- migrating existing documentation
- Integrating modeling tools in toolchain
- Integrating modeling into process
- Keeping models & documentation up to date

Project: Towards an online learning environment for software design

with Dave Stikkolorum

- Simple UML editor that runs in a browser
- Logs activities (creating, moving, renaming)
- Provides interactive feedback to students

Goal is to integrate ‘doing design’ in online learning courses (UML/Analysis & Design/Softw architecture)
Visualization of log of modeling activities

Fig. 2: LogViz visualisation tool

Summary:
AI will augment Human Intelligence in Software Engineering
Modelling style and model purpose

- **understanding**
- **communicating**
- **managing complexity**
- **blueprint for production**

Styles of using UML:
- as a sketch
- for communicating system design
- as a blueprint — guide the implementation work
- as a implementation (MDA) - code generation

Effect of Defects on Understandability of UML

**Experiment:**
Show participants 2 types of UML models
A. ‘good model’
B. model omission / inconsistency

For ‘faulty’ models, participants more often differ in their interpretation.

Both students & professionals

Challenge 1
Can we automatically assess the quality of a software design?

Could design \[\rightarrow\] Good implementation

Project: Quality Assessment of Software Design - the MetricView tool

When is a design ‘good’?
Absence of ‘bad things’?

Quality Metrics capture design principles
- Coupling
- Cohesion
- ... <extensible>

\[ = \text{violation} \quad \checkmark = \text{ok} \]

with Christian Lange & Johan Muskens
MetricView Tool

http://www.youtube.com/watch?v=GHJ_QR9EG4

MetricView
The values of metrics are visualized on class diagrams using colors
Example: Coupling-Between-Objects (CBO)

Industrial Experience with MetricView as Quality Assurance Tool

• Based on 15+ industrial projects
• If there are weak spots in the design, then these are indicated as ‘suspects’ by MetricView
• About 90% of the weak spots indicated by MetricView do not require improvement according to project architects

Syntactic checks (like metrics) are not sufficient to identify important areas in the design!

• The later MetricView is applied, the fewer ‘weak spots’ are removed from the design → process issue
**Theory of Benefits of Modeling**

- **UML Modeling**

  - **Developer Benefits**
    - Better Understanding Problem Domain
    - Improved Communication
    - Better Risk management

  - **Process Benefits**
    - Improved requirements
    - Improved Design Compliance
    - Improved Design Quality

  - **Product Benefits**
    - Fewer Defects
    - Reduced rework
    - More efficient Testing
    - Reduced testing effort
    - Reduced maintenance effort

- **Better Understanding Solution Space**
- **Shared System Model**
- **More Accurate Estimating**

- **Improved Quality**
- **Improved Design Quality**
- **Improved Design Compliance**
- **Improved requirements**

- **Better Productivity**
- **Improved Maintenance**
- **Reduced rework**

- **Better Understanding Solution Space**
- **More Accurate Estimating**

- **Project Management Benefits**
- **More efficient Testing**
- **Reduced testing effort**
- **Reduced maintenance effort**

---

**Integrate Modeling into Process and Tooling**

**Naming and layout-conventions**

**Version Management**

- Many tools around (e.g. CVS, SVN, ...)

**Reviews & Inspections**

- Guidelines by e.g. Shull et.al., Biffl

**Process**

- Integrate into process:
- ‘Definition of done’


**Definitions**

**Model (noun)** = an abstract representation of a thing/system, often systematic representation

Models abstract: they focus on the essential features and leave out others.

**Modeling** = the process of creating a model (verb) i.e. choosing what to represent and how to represent it

---

**Design (verb, noun)**

Definition

**Design (v)** = the process of making decisions about something that is to be built or created:

**Design (n)** = the plans, drawings, etc., that show how something can be made

Pitfall: ‘design’ & ‘model’ can be a verb and a noun
How do ‘modeling’ and ‘design’ relate?

Modeling and designing often go hand-in-hand:

A model is used to understand, reason, analyse, break-down, which leads to adaptation and refinement of the design.

Stages of Design & Modeling

<table>
<thead>
<tr>
<th>Building</th>
<th>Ideation</th>
<th>Externalization</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animation</td>
<td>Develop idea</td>
<td>Explain to others</td>
<td>Blueprint for Production</td>
</tr>
</tbody>
</table>
Challenge: Visualisation

- Scale
- Multiple uses
- Multiple abstraction levels
  - Models & Source Code
How to layout packages of classes?

Economy of Modeling

Size of the system

% of system covered by model
Integration of Tools in Design Flow and Development Process

Are the tools the problem?

Project: OctoUML with Rodi Jolak

- Merge whiteboard (informal, free-form) and CASE-tool (formal) modeling

Digital editing for sketching: undo, resize, move, pan, zoom

Novel UI interaction models: touch, voice

Goal: Tools should support the development process (not only the modeling task)

https://www.youtube.com/watch?v=fsN3rIEAYHw&t=74s
**Challenge**

Does the quality of a software design related to the quality of the implemented software?

- Good ‘more thought through’ design
- Good implementation: Fewer defects?

---

**Project: Does Quality of Modeling Matter?**

An Industrial Case Study

Focus on **detail** in a UML Model

with Ariadi Nugroho

Low detail

High detail
Relation between UML-LoD and Code Quality

- Select ‘defects’ in defect DB
- Find classes in source code that were repaired for solving this defects
- Find corresponding classes in UML models
- Determine LoD for CD and SD

Relation between Level of Detail and Defect Density

Level of Detail for Sequence Diagrams is significantly (negatively) correlated with defect density. More detailed model => fewer defects

What about class diagrams?