University of Zurich UZH

THE CODE IS THE MODEL

LUZIUS MEISSER AT THE WORLD CONGRESS OF MICROSIMULATION

Background

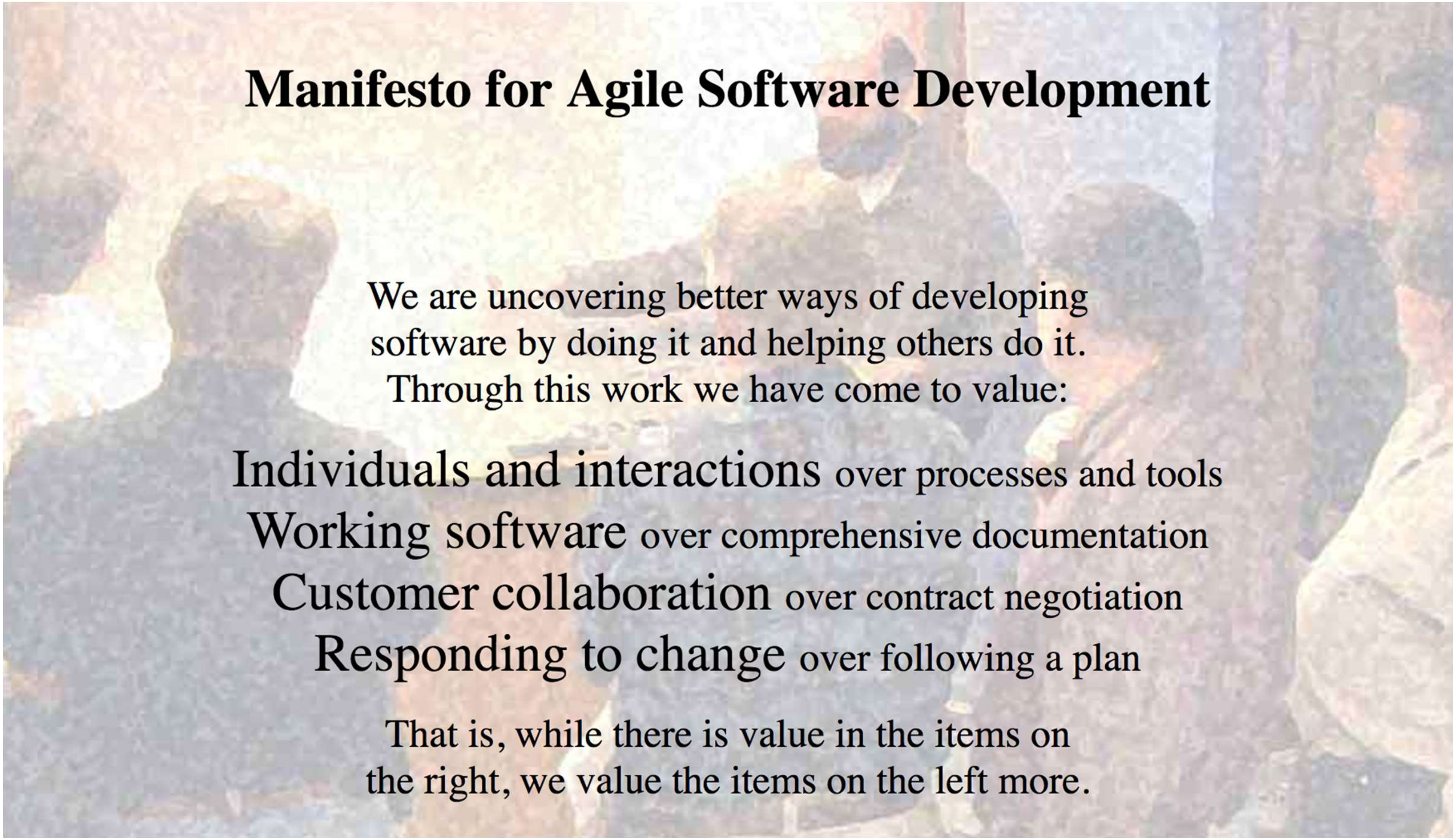
- MSc Computer Science
- Years of Java programming experience as the CTO of startup
- MA in Economics
- Currently pursuing a PhD on “Agent-based financial economics” with Prof. Thorsten Hens (evolutionary finance, heuristics)
- Will be teaching a course on “Agent-based financial economics” at University in Zurich next semester

The Code is the Model

Inspired by agile software engineering.
Consequences:

1. The code is the specification
2. Improved replicability
3. Programming skills gain importance
4. Models can be built like software
5. Better modularity, management of complexity
6. Peer-reviews should include replication and code inspections
7. Better quality models

The Agile Manifesto



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

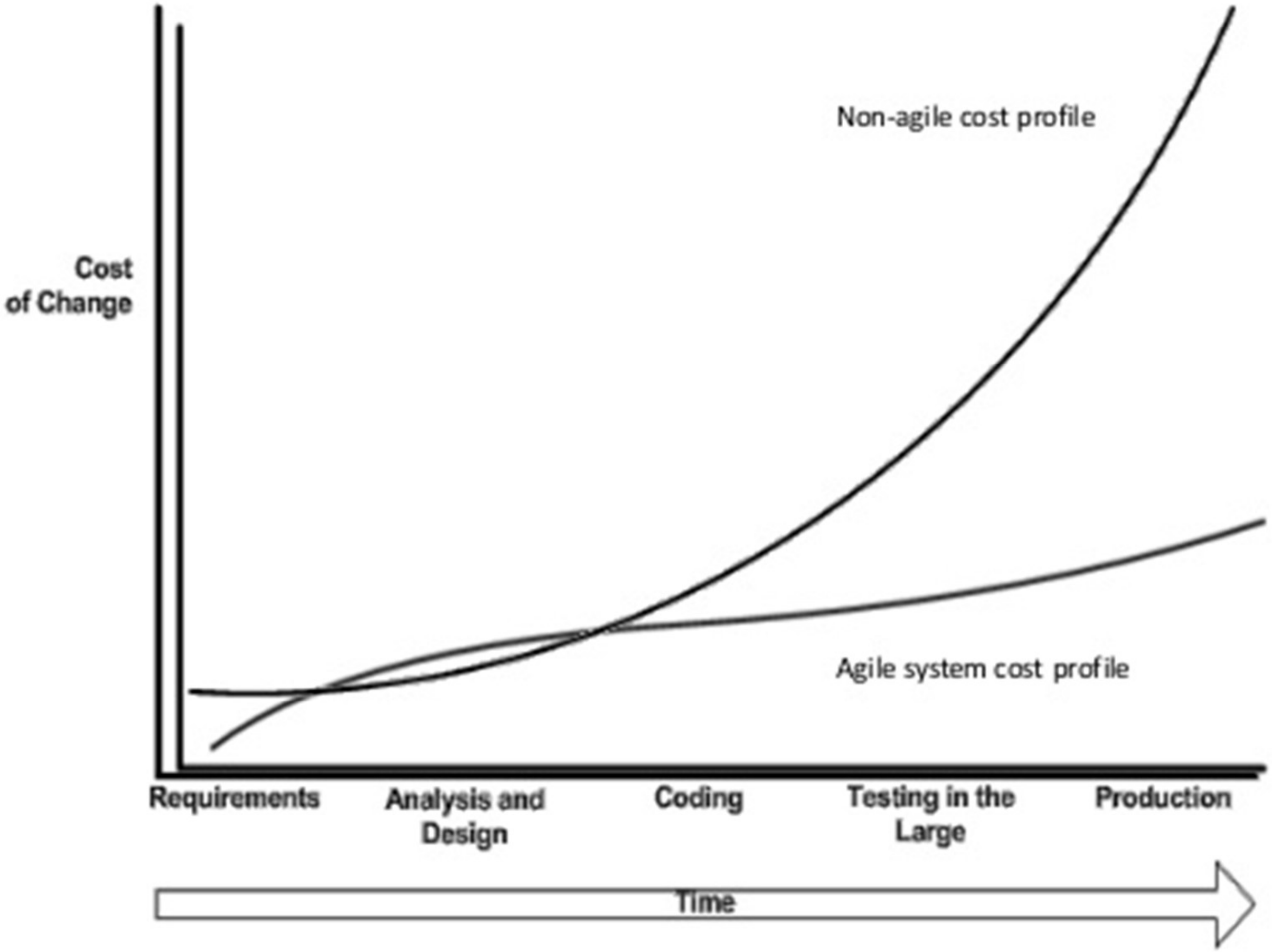
That is, while there is value in the items on
the right, we value the items on the left more.

Cost of Change

Traditional view: ”An error in the design stage costs ten times more to correct in the coding stage and a hundred times more to fix after the program is in use.” – Page and Miller (2007)

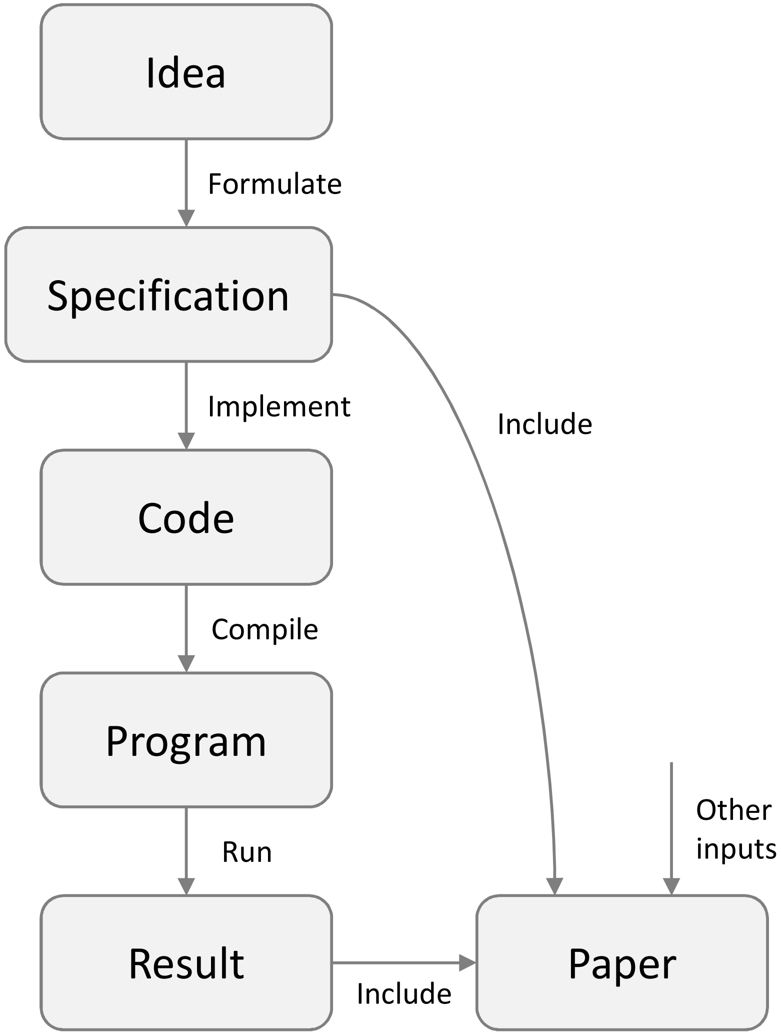
How does a decrease in the cost of change affect the returns on upfront design?

Cost of Change



The agile view:
"The only way we validate a software design is by building it and testing it. There is no silver bullet, and no 'right way' to do design. Sometimes an hour, a day, or even a week spent thinking about a problem can make a big difference when the coding actually starts. Other times, 5 minutes of testing will reveal something you never would have thought about no matter how long you tried. We do the best we can under the circumstances, and then refine it." – Reeves (2005)

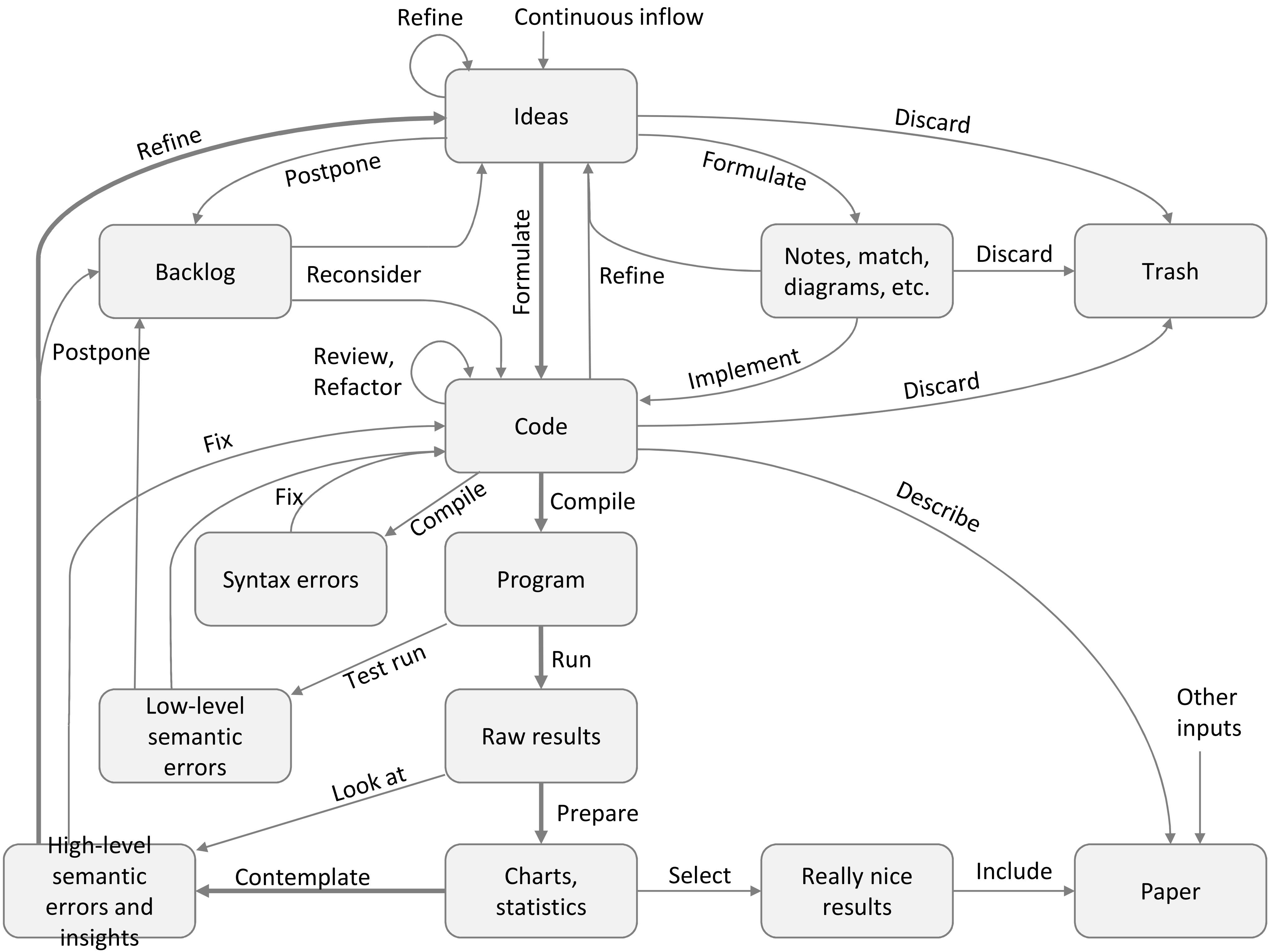
Modeling Process: traditional



The ”waterfall” model of software development.

Despite not being very realistic, this is how agent-based modeling is often implicitly assumed to work.

There is no indication on how to handle errors or how to refine the original idea except by starting from scratch.


Refine
Continuous inflow
Ideas
Discard
Refine
Postpone
Formulate
Backlog
Reconsider
Formulate
Refine
Notes, match, diagrams, etc.
Discard
Trash
Postpone
Review, Refactor
Implement
Discard
Code
Fix
Fix
Compile
Compile
Describe
Syntax errors
Program
Test run
Run
Low-level semantic errors
Raw results
Other inputs
Look at
Prepare
High-level semantic errors and insights
Contemplate
Charts, statistics
Select
Really nice results
Include
Paper

Model Specification: Text + Math

- Most common approach
- Well-suited for small, simple models, e.g. Schelling’s segregation model
- Problematic with larger models

Model Specification: The ODD Protocol

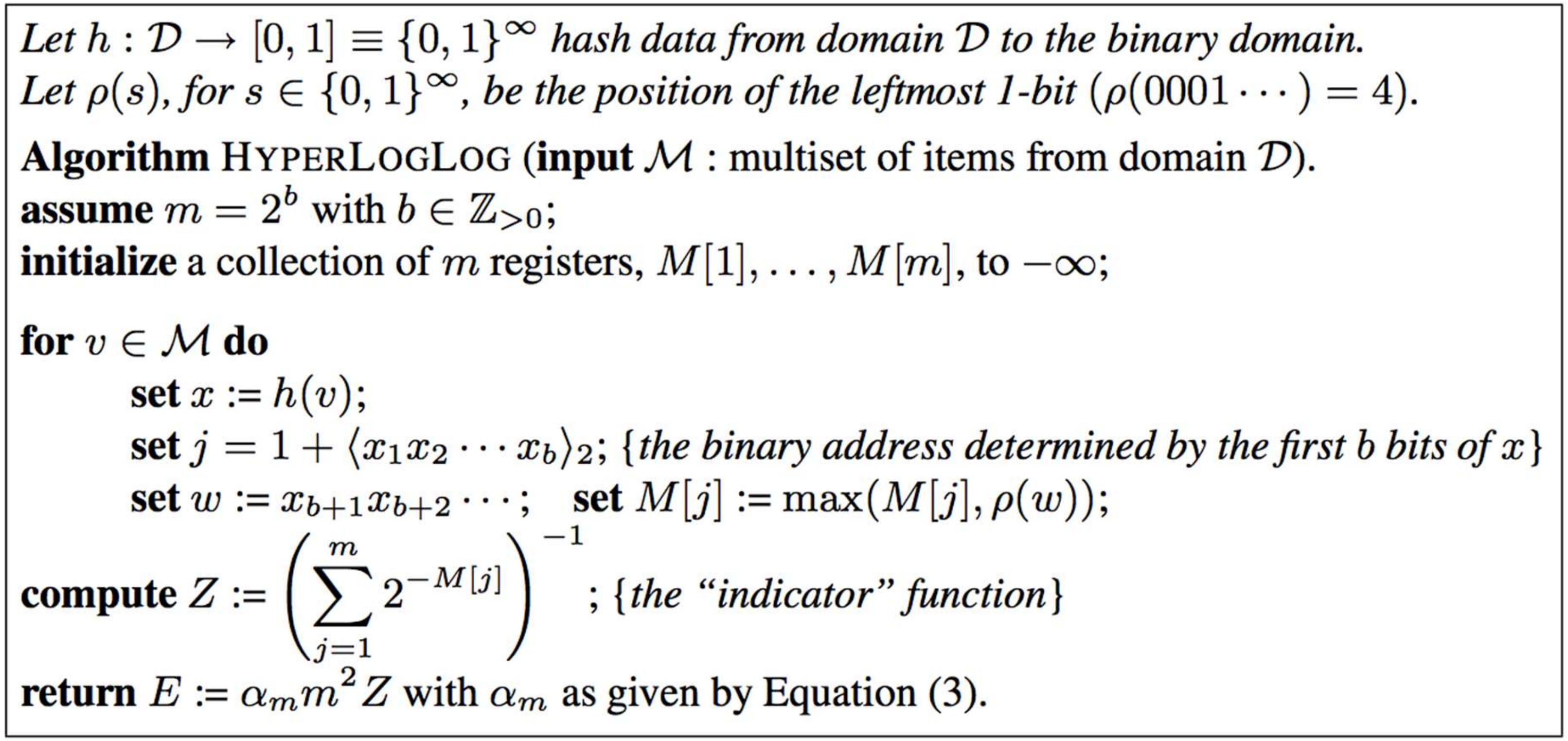
The Model

The model’s ODD specification

"Once readers know the full set of (low-level) state variables, they have a clear idea of the model’s structure and resolution." - Grimm et al (2006)

Model Specification: Pseudo-Code

Common in computer science.
Suitable for short algorithms up to 25 lines.



```
Let $h : \mathcal{D} \to [0,1] \equiv \{0,1\}^\infty$ hash data from domain $\mathcal{D}$ to the binary domain.
Let $\rho(s)$, for $s \in \{0,1\}^\infty$, be the position of the leftmost 1-bit ($\rho(0001\cdots) = 4$).
Algorithm HYPERLOGLOG (input $\mathcal{M}$ : multiset of items from domain $\mathcal{D}$).
assume $m = 2^b$ with $b \in \mathbb{Z}_{>0}$;
initialize a collection of $m$ registers, $M[1], \ldots, M[m]$, to $-\infty$;

for $v \in \mathcal{M}$ do
        set $x := h(v)$;
        set $j = 1 + \langle x_1 x_2 \cdots x_b \rangle_2$; {the binary address determined by the first $b$ bits of $x$}
        set $w := x_{b+1} x_{b+2} \cdots$;   set $M[j] := \max(M[j], \rho(w))$;
compute $Z := \left( \sum_{j=1}^{m} 2^{-M[j]} \right)^{-1}$; {the "indicator" function}
return $E := \alpha_m m^2 Z$ with $\alpha_m$ as given by Equation (3).
```

Fig. 2: The HYPERLOGLOG Algorithm.

Model Specification: Source Code

Agent-based models are fundamentally algorithmic and often of non-trivial size. Also, they can be very sensitive to small changes, so providing incomplete specifications is not an option. The cleanest way to specify agent-based models is to use source code.

All other artifacts, such as flow charts, UML diagrams, natural language descriptions, ODD tables, and mathematical equations can be suitable means to describe the model at a higher level, but not to fully specify it. If an interested reader wants to know about the complete model in all its details, she should consult the source code, with the paper serving as a guide.

"In software development, the design document is a source code listing."
- Reeves (2005)

Replicability

- Replicability is the ability to repeat an experiment and to get the same results
- Replication is the foundation of empirical science
- In agent-based modeling, running the simulation is the experiment
- Seeing the code as the specification eliminates the error-prone translation step, thereby improving replicability.
- Replicating agent-based models can “simply” be done by compiling and running them.

Chang and Phillip [2015] try to reproduce the results of 67 economics papers published in a selection of 13 reputable journals. They could only replicate 33% of them on their own, and an additional 10% with the authors’ assistance. The primary reason for a failure to replicate results was missing software or data – even for journals that in theory have a policy of requiring source code and data.

Verifying a model’s description and the presented interpretation of the results is also important, but not part of replication. (Müller et al. [2014] disagree: "although the provision of source code technically facilitates model replication, it may circumvent the consistency check between the conceptual model and its implementation (one purpose of model replication) by encouraging “replicators” to simply copy the source code.”)

What is replication?

Should replication of a flawed experiment include these flaws?

Example: Dr. Zoidberg measures whether an apple or an orange fall faster. To do so, he lets both fruits fall at the same time and lets Fry time the apple and Leela time the orange. He writes a scientific paper reporting that oranges fall faster.
(But actually, he just measured that Leela reacts faster than Fry.)

Should someone replicating this experiment:

- Use the same flawed method?
- Use a correct method?

Replicability Checklist Part 1 of 2

1. Your paper must contain a high-level description of the model.
2. Your paper must link to the model's source code. Preferrably, the code is hosted in a browsable web repository such as github.com.
3. Along with the link to the code, the fingerprints (hash) of the discussed versions should be provided. This proves that the code was not changed after the submission of the paper.
4. Your code must include a readme file with instructions on how to compile and run the simulation. This should include the program inputs and the expected outputs for each discussed result.
5. Your simulation should be deterministic. Running the same configuration twice should yield the exact same result.

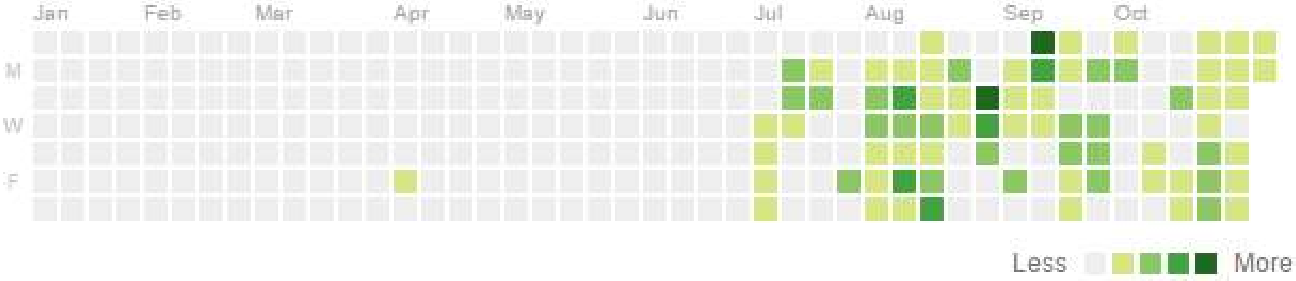
Replicability Checklist Part 2 of 2

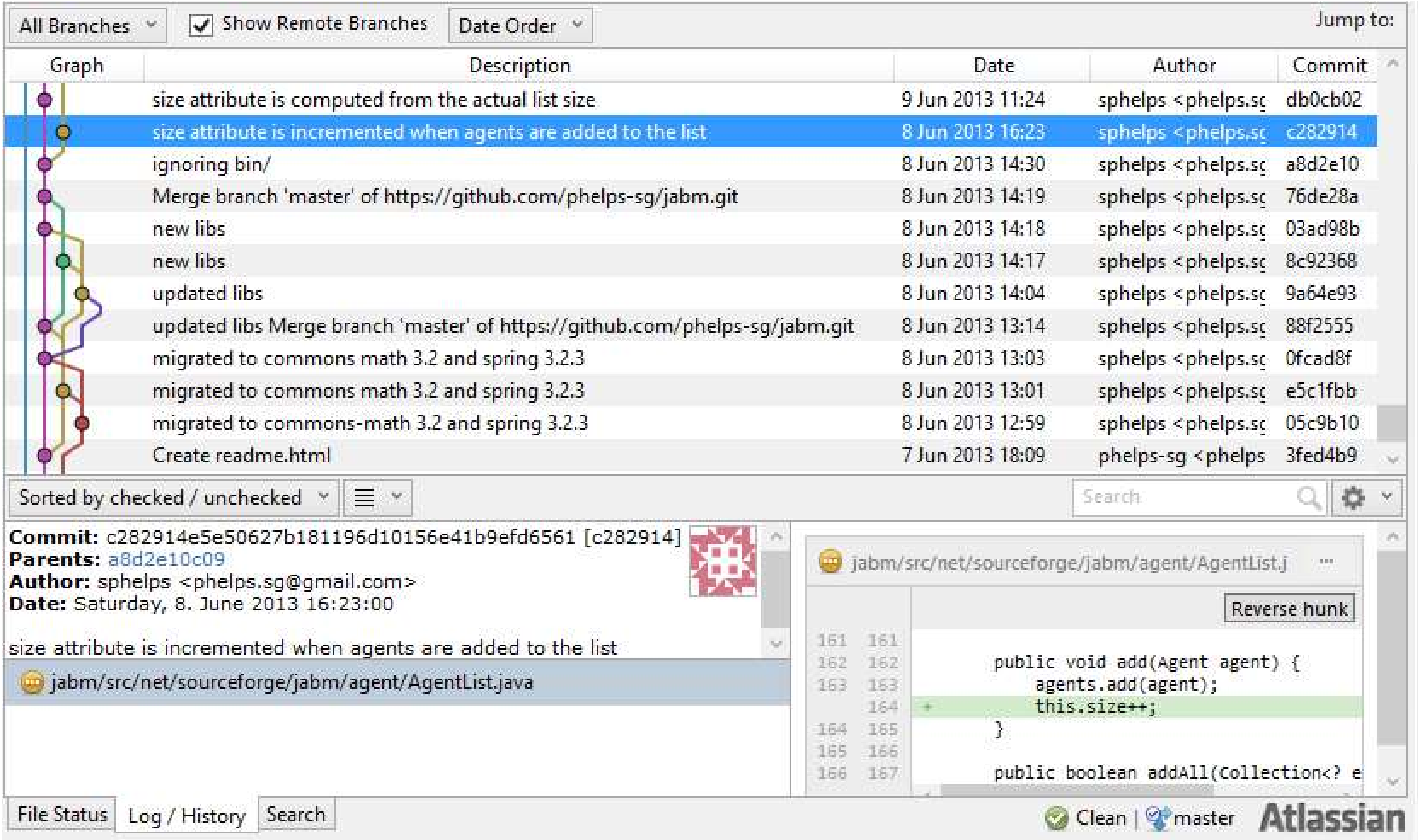
6. You should specify under which conditions the code can be reused, for example under the MIT license (MIT [1988]). Academic use under the condition of proper attribution must be permitted.
7. You should encourage others to clone your model into their own repositories in order to improve long-term availability. Prominently add the title of your paper to the readme file so the repository can be found with a web search even when the original links are broken.
8. The tools and libraries required to compile and run the software should be freely available for academic use. For example, Jupyter notebooks should be preferred over Mathemica notebooks.
9. You are encouraged to cross-reference the paper from the code and vice versa. In particular, you should make clear how the variable names from the paper (usually single-lettered) map to the ones in the code (should be long and descriptive).
10. Keep your model simple and accessible by following the software design hints from the paper.

Consequences for Publishing

- Papers should follow a replicability checklist.
- Papers present the gained insights and serve as a guide to the source code, but do not contain a full specification.
- Code should be fingerprinted and provided in a public, browseable repository.
- Only readers interested in the details of the model should have to inspect the source code.
- The peer review should include a replication check. If replication is not possible with reasonable effort, the paper can be rejected.
- Ideally, the reviewers also scrutinize the source code.

Github





☑ Show Remote Branches

Date Order

Graph	Description	Date	Author	Commit
	size attribute is computed from the actual list size	9 Jun 2013 11:24	sphelps <phelps.sç	db0cb02
	size attribute is incremented when agents are added to the list	8 Jun 2013 16:23	sphelps <phelps.sç	c282914
	ignoring bin/	8 Jun 2013 14:30	sphelps <phelps.sç	a8d2e10
	Merge branch 'master' of https://github.com/phelps-sg/jabm.git	8 Jun 2013 14:19	sphelps <phelps.sç	76de28a
	new libs	8 Jun 2013 14:18	sphelps <phelps.sç	03ad98b
	new libs	8 Jun 2013 14:17	sphelps <phelps.sç	8c92368
	updated libs	8 Jun 2013 14:04	sphelps <phelps.sç	9a64e93
	updated libs Merge branch 'master' of https://github.com/phelps-sg/jabm.git	8 Jun 2013 13:14	sphelps <phelps.sç	88f2555
	migrated to commons math 3.2 and spring 3.2.3	8 Jun 2013 13:03	sphelps <phelps.sç	0fcad8f
	migrated to commons math 3.2 and spring 3.2.3	8 Jun 2013 13:01	sphelps <phelps.sç	e5c1fbb
	migrated to commons-math 3.2 and spring 3.2.3	8 Jun 2013 12:59	sphelps <phelps.sç	05c9b10
	Create readme.html	7 Jun 2013 18:09	phelps-sg <phelps	3fed4b9

Sorted by checked / unchecked

Commit: c282914e5e50627b181196d10156e41b9efd6561 [c282914]
Parents: a8d2e10c09
Author: sphelps <phelps.sg@gmail.com>
Date: Saturday, 8. June 2013 16:23:00

size attribute is incremented when agents are added to the list

jabm/src/net/sourceforge/jabm/agent/AgentList.java

jabm/src/net/sourceforge/jabm/agent/AgentList.j

```
    public void add(Agent agent) {
        agents.add(agent);
+       this.size++;
    }

    public boolean addAll(Collection<? e
```

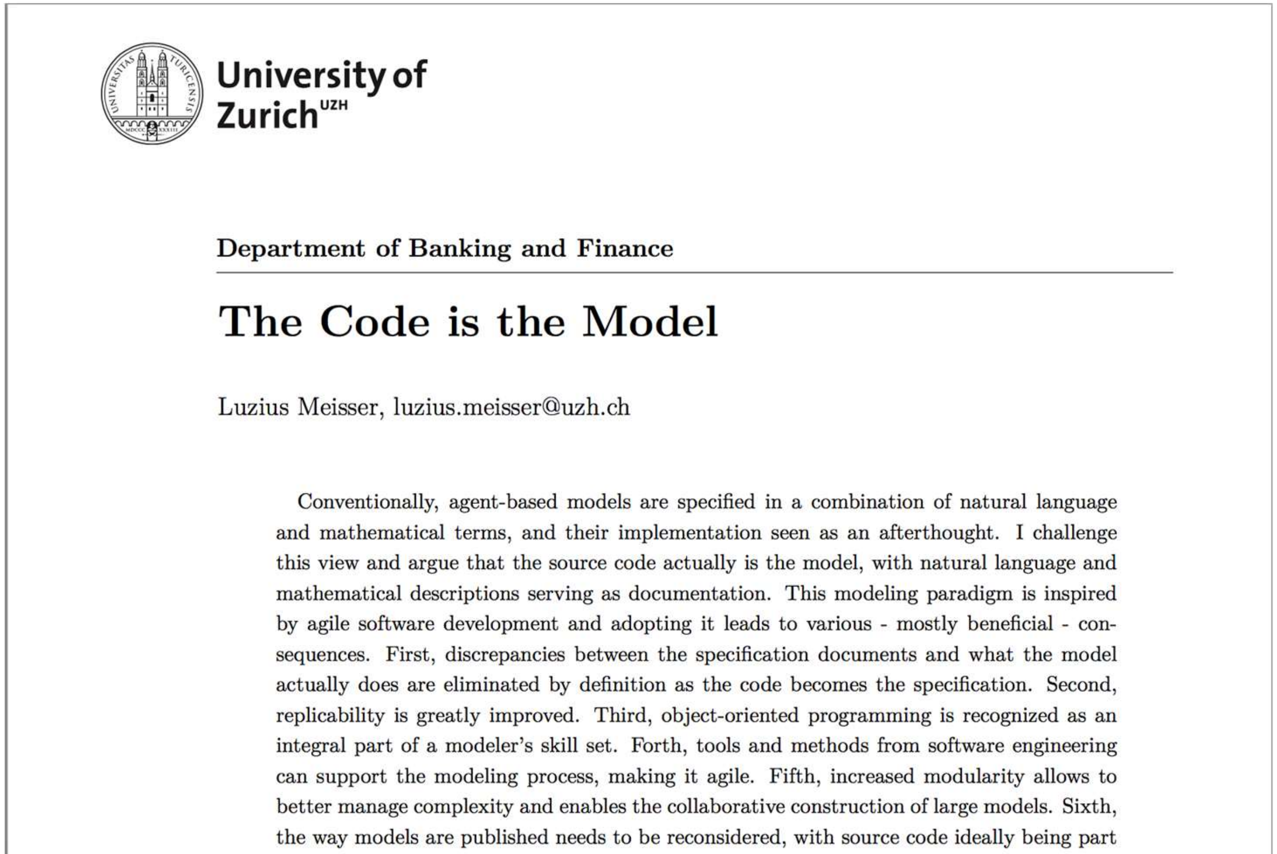
Impact on Model Quality

Towards a new generation of higher quality models:

- Agile software engineering puts the code at center stage. As more attention is paid to it, its quality increases.
- Focus on continuous, incremental improvements.
- High code quality is part of the model's quality (simplicity, conciseness, etc.)
- Better replicability, which also is part of a model's quality.
- Code reviews can increase code & model quality further. (Proven, measurable effect.)
- Inventives: The more attention code receives, the higher the incentive of the author to deliver high quality.

Questions?

Paper available at meissereconomics.com
Contact: luzius@meissereconomics.com



University of Zurich^UZH

Department of Banking and Finance

The Code is the Model

Luzius Meisser, luzius.meisser@uzh.ch

Conventionally, agent-based models are specified in a combination of natural language and mathematical terms, and their implementation seen as an afterthought. I challenge this view and argue that the source code actually is the model, with natural language and mathematical descriptions serving as documentation. This modeling paradigm is inspired by agile software development and adopting it leads to various - mostly beneficial - consequences. First, discrepancies between the specification documents and what the model actually does are eliminated by definition as the code becomes the specification. Second, replicability is greatly improved. Third, object-oriented programming is recognized as an integral part of a modeler's skill set. Forth, tools and methods from software engineering can support the modeling process, making it agile. Fifth, increased modularity allows to better manage complexity and enables the collaborative construction of large models. Sixth, the way models are published needs to be reconsidered, with source code ideally being part