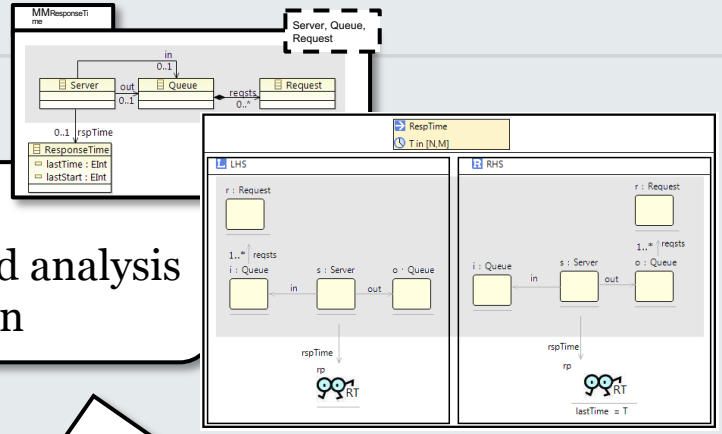
Why you should care about Model-Driven Engineering

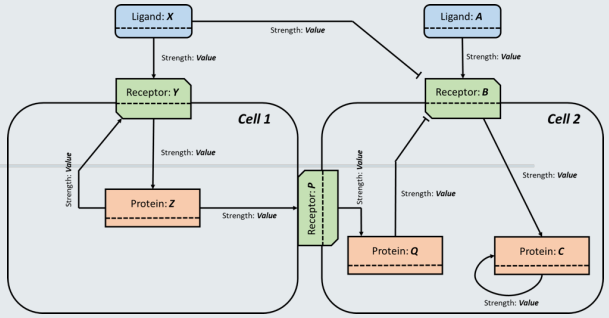
Steffen Zschaler
MDENet, King's College London

MDENet | The home of model-driven engineering

KING'S College LONDON

Who am I?





Foundations

Formal models of simulation-based analysis
Safe reuse and composition

$$[GTS_0]_T \rightsquigarrow_T GTS'_0 \xrightarrow{f} GTS_2$$
$$\downarrow g \qquad \downarrow g' \qquad \downarrow \hat{g}$$
$$[GTS_1]_T \rightsquigarrow_T GTS'_1 \dashrightarrow \widehat{GTS}$$

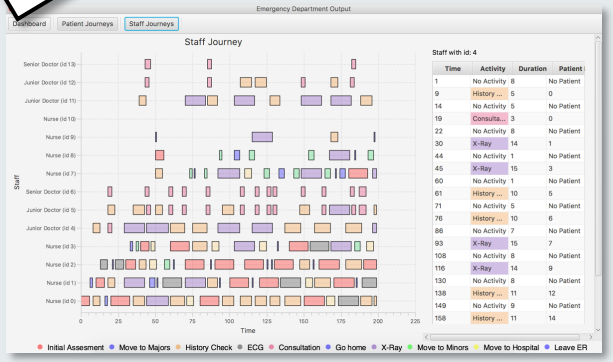
Participatory Modelling

Enabling stakeholder engagement with simulation models
DSMLs and processes

Model-Driven Engineering

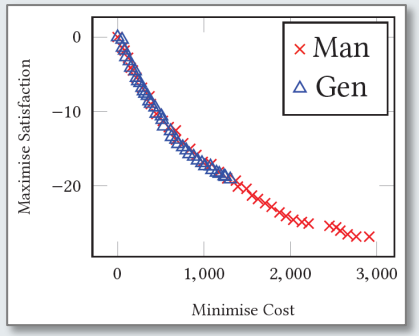
Applications

Biology
Healthcare
Robotics



Search-Based Exploration

Model-Based Search and Optimisation



Challenges in Software Engineering

Challenges in software engineering

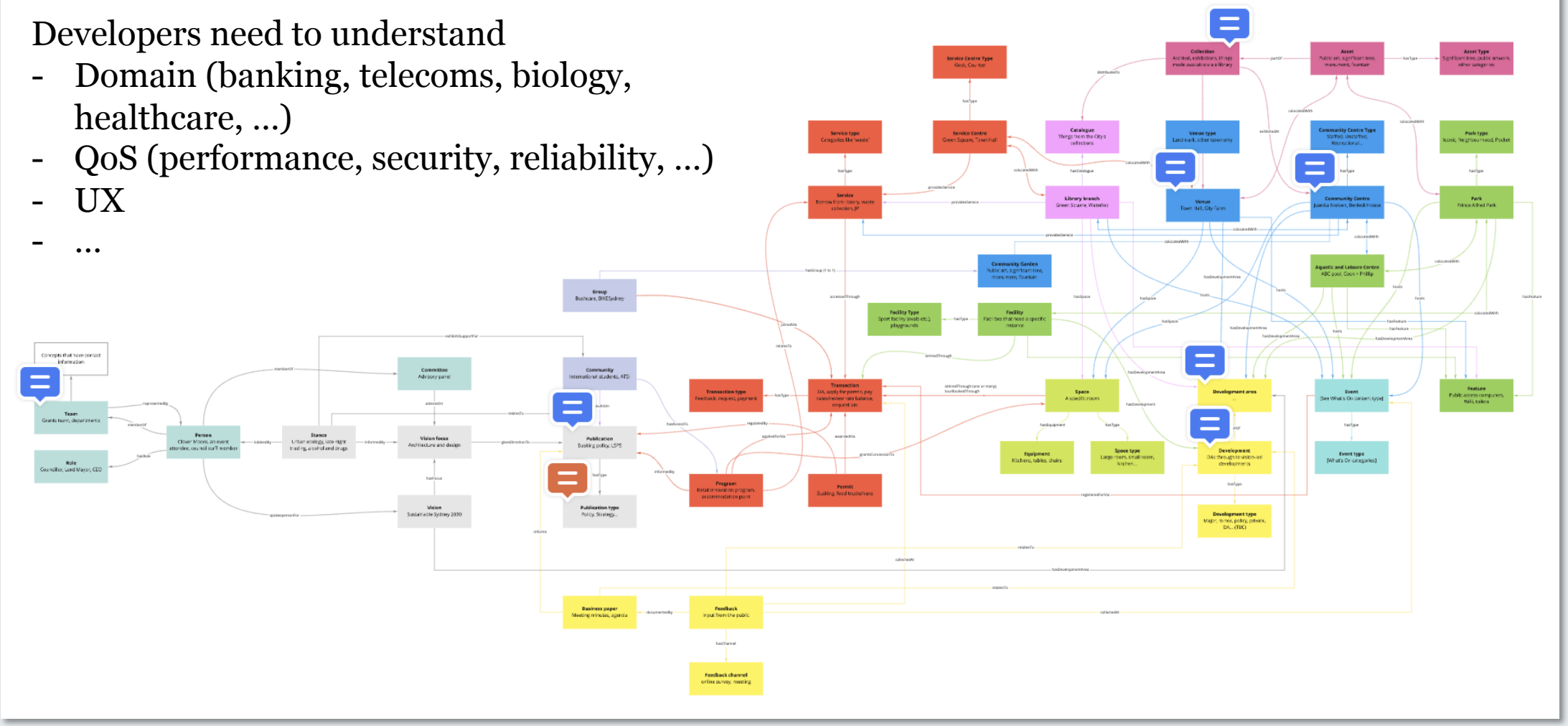
Dimensions of complexity

- Complexity of domain
- Size
- Reuse and choice
- Cognitive mismatch
- Distribution of knowledge and expertise

Complexity of domain

Developers need to understand

- Domain (banking, telecoms, biology, healthcare, …)
- QoS (performance, security, reliability, …)
- UX
- …



https://uxdesign.cc/creating-a-content-model-for-a-broad-subject-1e62129e3521?gi=7e2c074b1a0

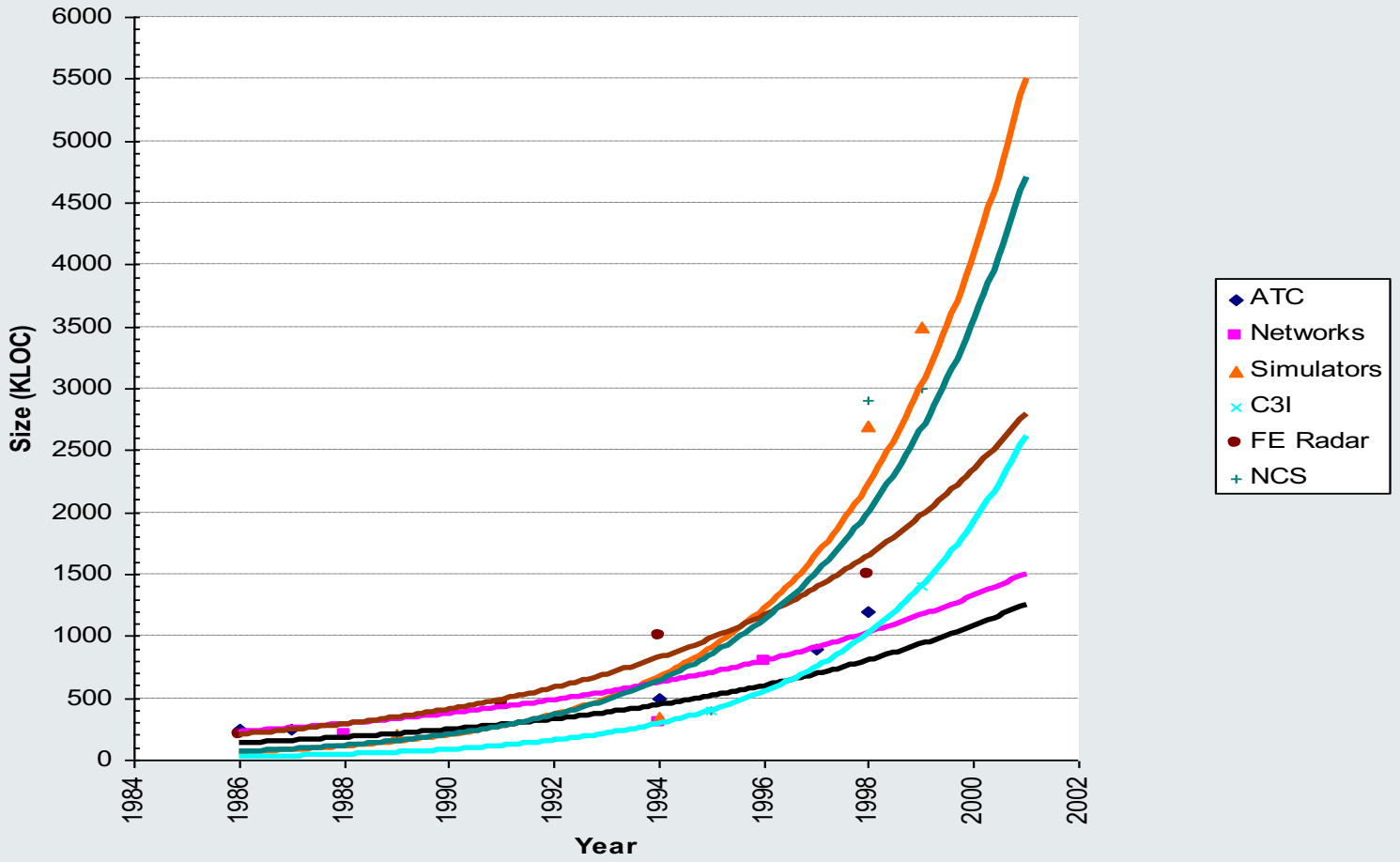
Challenges in software engineering

Dimensions of complexity

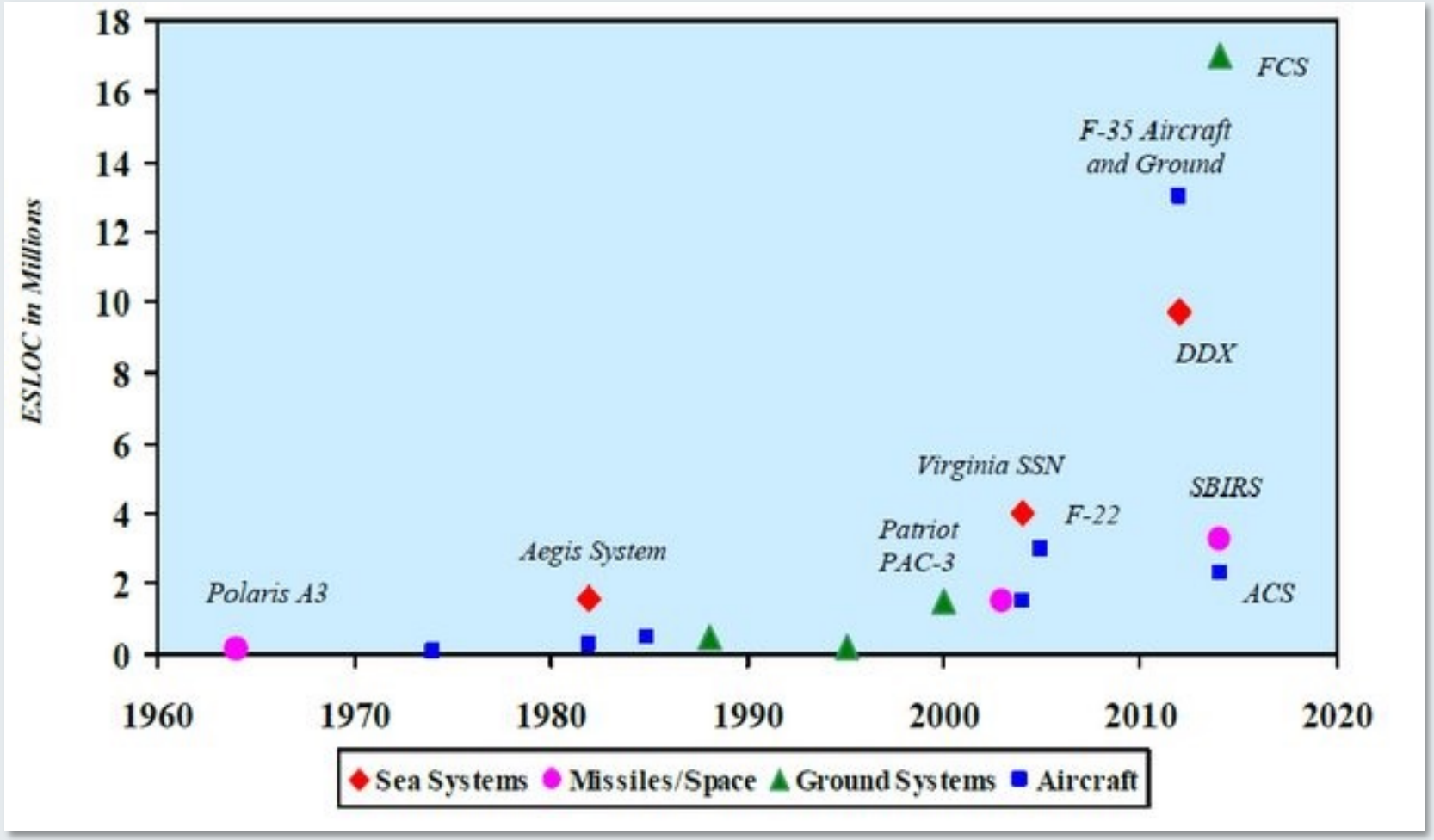
- Complexity of domain
- Size
- Reuse and choice
- Cognitive mismatch
- Distribution of knowledge and expertise

Software size grows exponentially

For example: Every 5 years, Thales systems grow by a factor of approx. 5 to 10

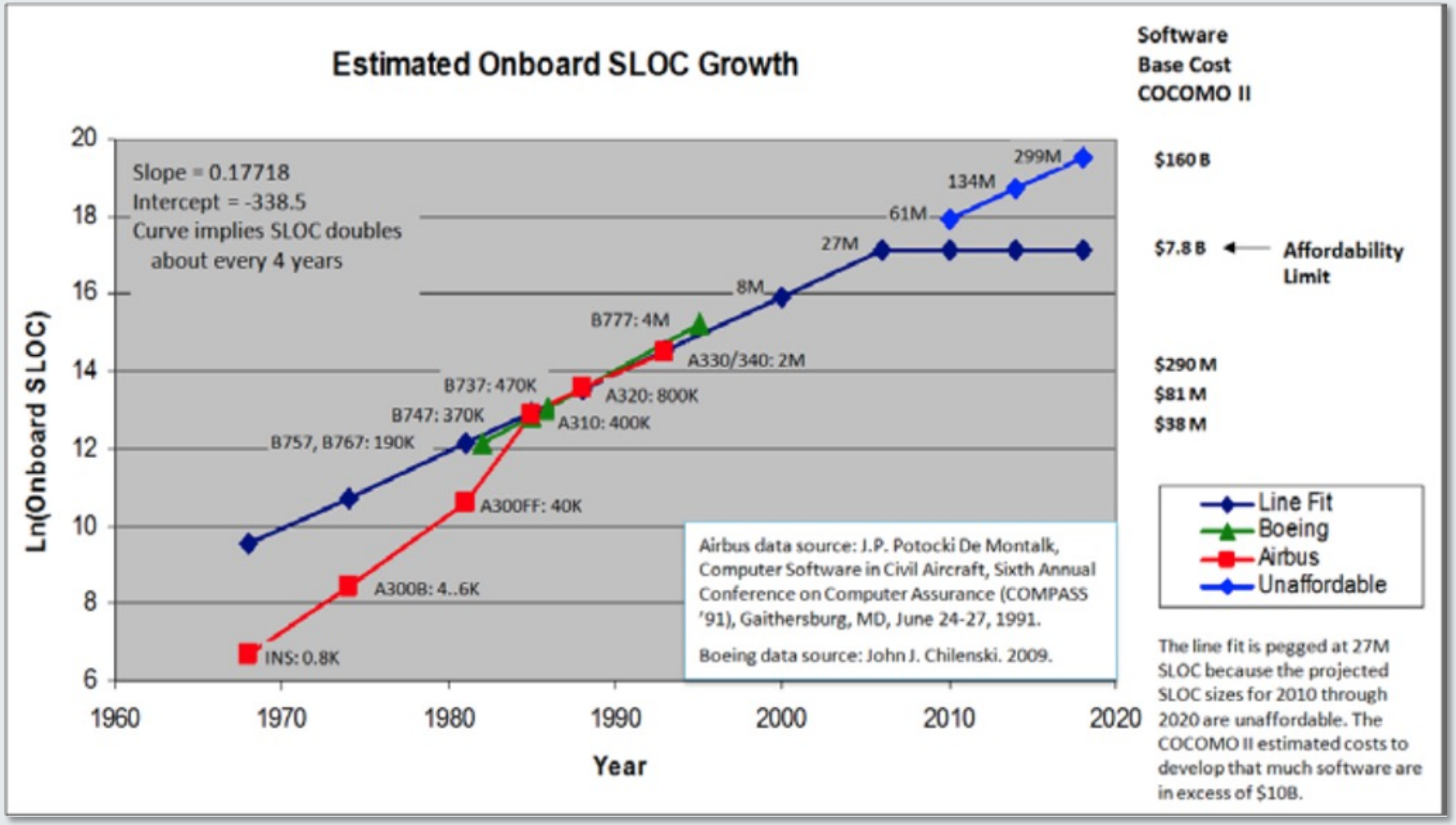


Software size grows exponentially – 2



Demir, Kadir & Caymaz, Ebru. (2016). Current Research Areas in Defense Software and Information Systems Project Management. 11th International Scientific Conference Defense Resources Management In The 21st Century.

Software size grows exponentially – 3



SEI, "Virtual Integration for Improved System Design", Redman et. al, 2010 https://wiki.sei.cmu.edu/aadl/images/d/de/SAVI_Virtual_Integration-AVICPS2010.pdf

Challenges in software engineering

Dimensions of complexity

- Complexity of domain
- Size
- Reuse and choice
- Cognitive mismatch
- Distribution of knowledge and expertise

Reuse and choice

Reuse is good, right?

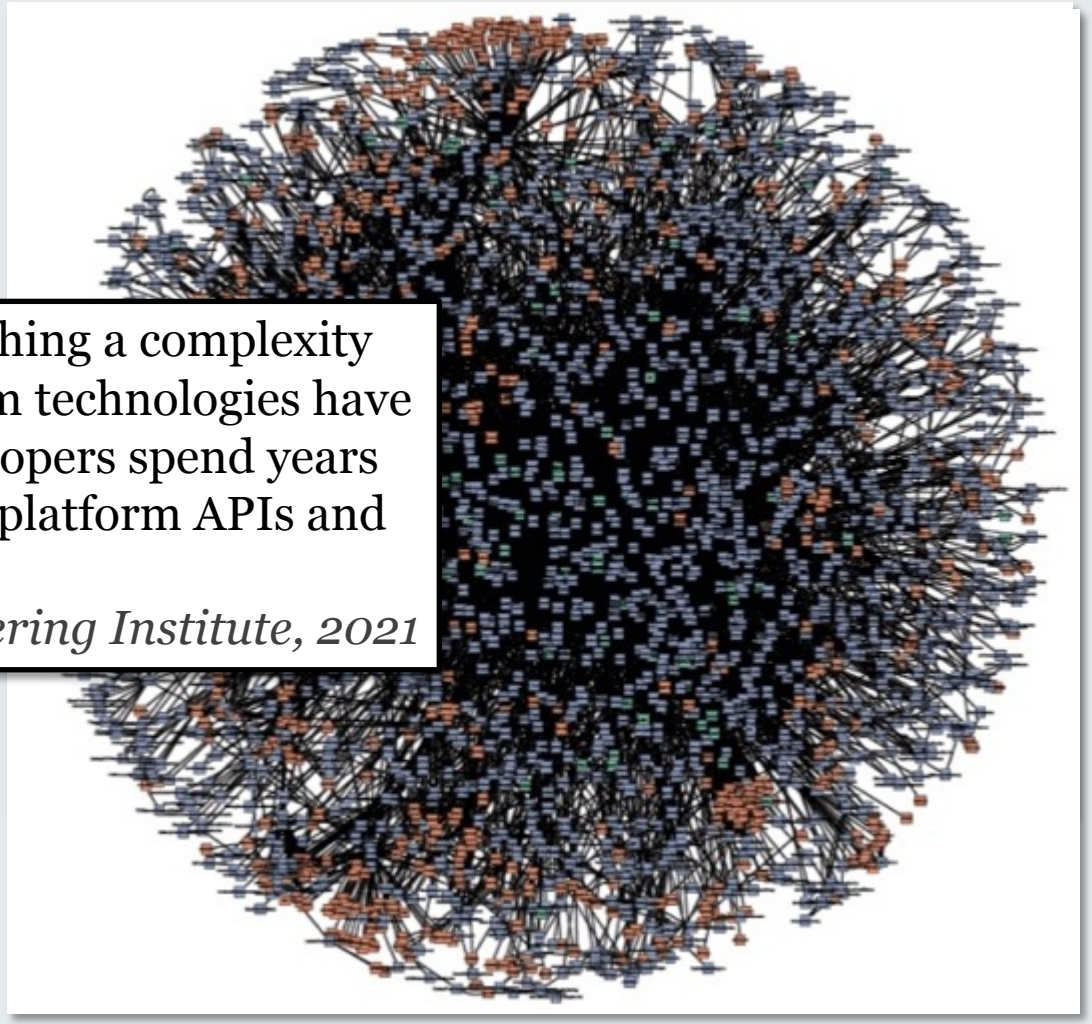
- Avoid reinventing the wheel
- Benefit from experience

Choice is good, right?

- Select the right tool for t

But:

- What to reuse and how?
- Who has the knowledge in those components?
- Proliferation of options (eg AWS vs Azure vs …)
- How do you maintain the system over time?



> "The software industry is reaching a complexity ceiling where modern platform technologies have become so complex that developers spend years mastering and wrestling with platform APIs and usage patterns."
>
> *Software Engineering Institute, 2021*

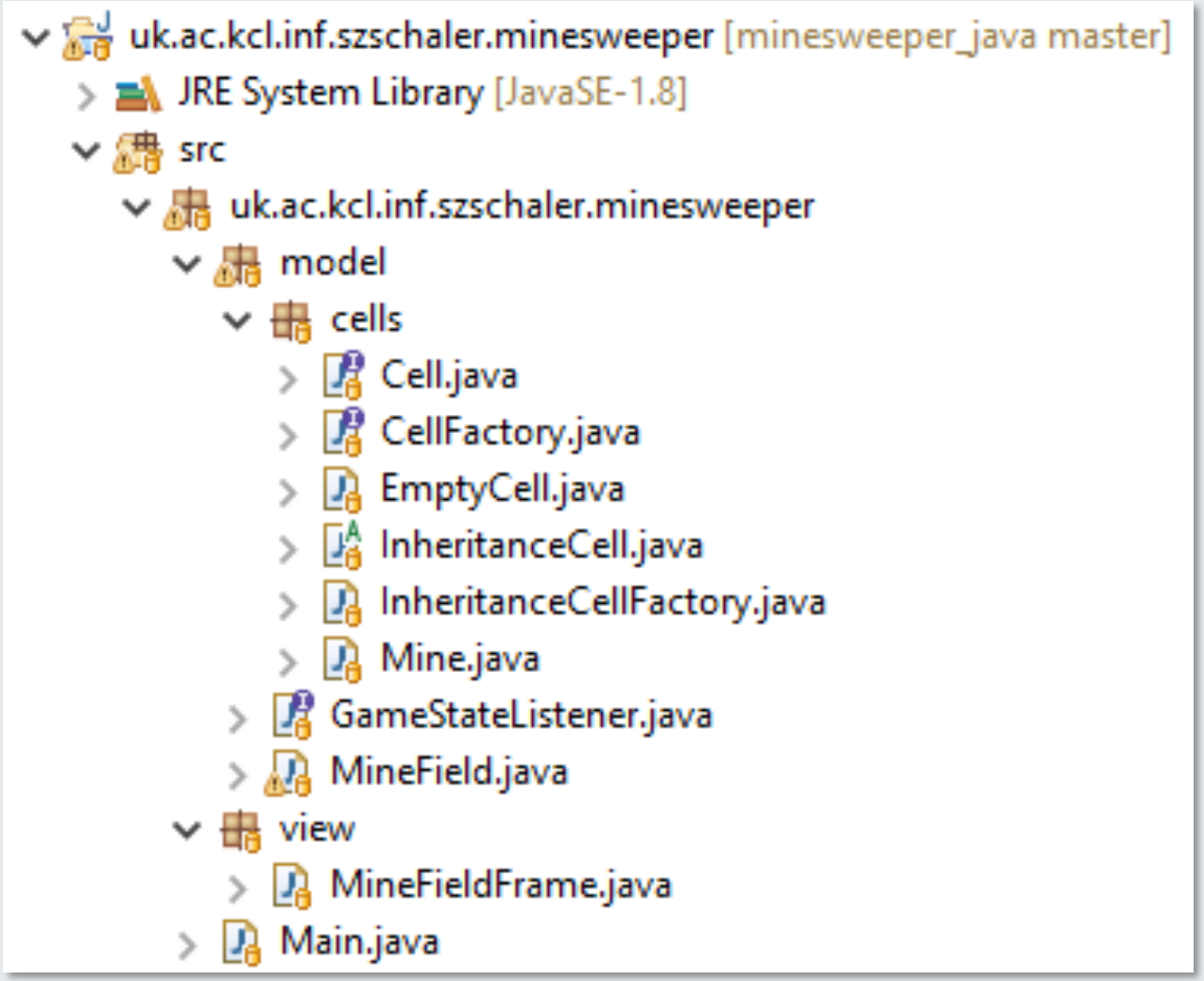
The Software Engineering Institute. https://www.sei.cmu.edu/news-events/news/article.cfm/Software-Engineering-at-a-Inflection-Point-Agenda-for-Software-Architecture-Research-and-Development-2021

Challenges in software engineering

Dimensions of complexity

- Complexity of domain
- Size
- Reuse and choice
- Cognitive mismatch
- Distribution of knowledge and expertise

Cognitive mismatch



What are the rules of the game?

Where would you start looking?

Challenges in software engineering

Dimensions of complexity

- Complexity of domain
- Size
- Reuse and choice
- Cognitive mismatch
- Distribution of knowledge and expertise

Where is the knowledge?

“The code is the documentation” – but is it?

Most knowledge is in

- Developers’ heads
  - High-level overviews kept stable by reification
  - Other information very unevenly distributed (“Gurus”)
- Written down in pull requests, Wiki pages, Google docs, developer notebooks, Slack channels, post-it notes, …

Causes challenges:

- How to find information?
- Is the information still up to date?
- Loss of rationale and traceability to original requirements / domain understanding

Model-Driven Engineering

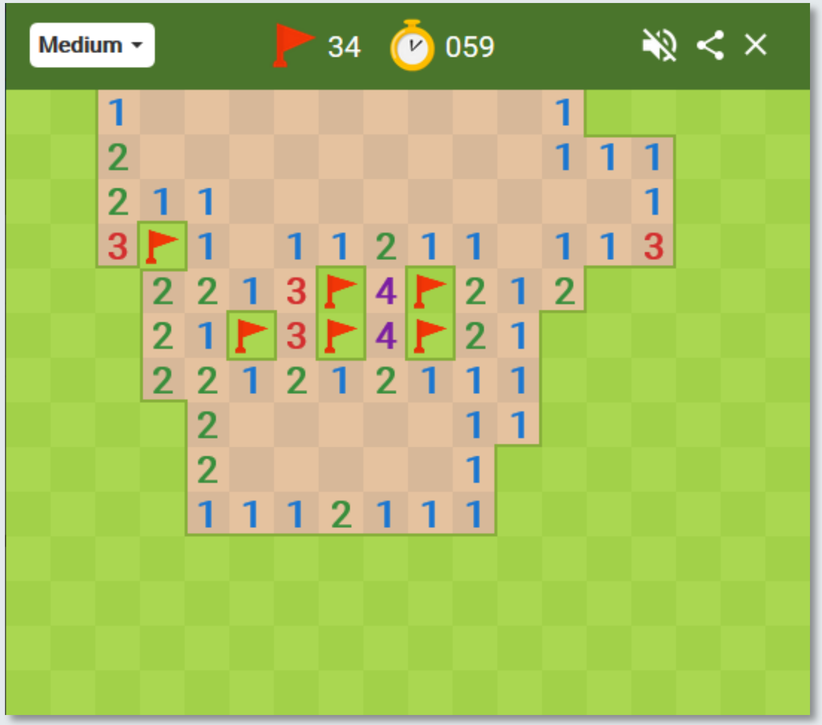
```
/*
 * Cells have states. The state machine is pretty much the same for all types of cells except
 * for the label to be shown on discovered cells and the action to be triggered when a cell is discovered.
 */
states mineStates (String discoveredLabel, Behaviour discoveredBehaviour) {
    hidden {
        display {
            as button
            text ""
        }

        transitions {
            on mouse-left goto discovered
            on mouse-right goto flagged
            on context (filter(_empty).inState(discovered).notEmpty()) goto discovered
        }
    }
    flagged {
        display {
            as button
            text "F"
        }

        transitions {
            on mouse-left goto discovered
            on mouse-right goto question
        }
    }

    question {
        display {
            as button
            text "?"
        }

        transitions {
            //on mouse-left goto discovered
            on mouse-right goto hidden
        }
    }

    discovered {
        display {
            as label
            var discoveredLabel
        }

        onEnter {
            discoveredBehaviour
        }
    }
    start=hidden
}
```

What about the rules of the game?

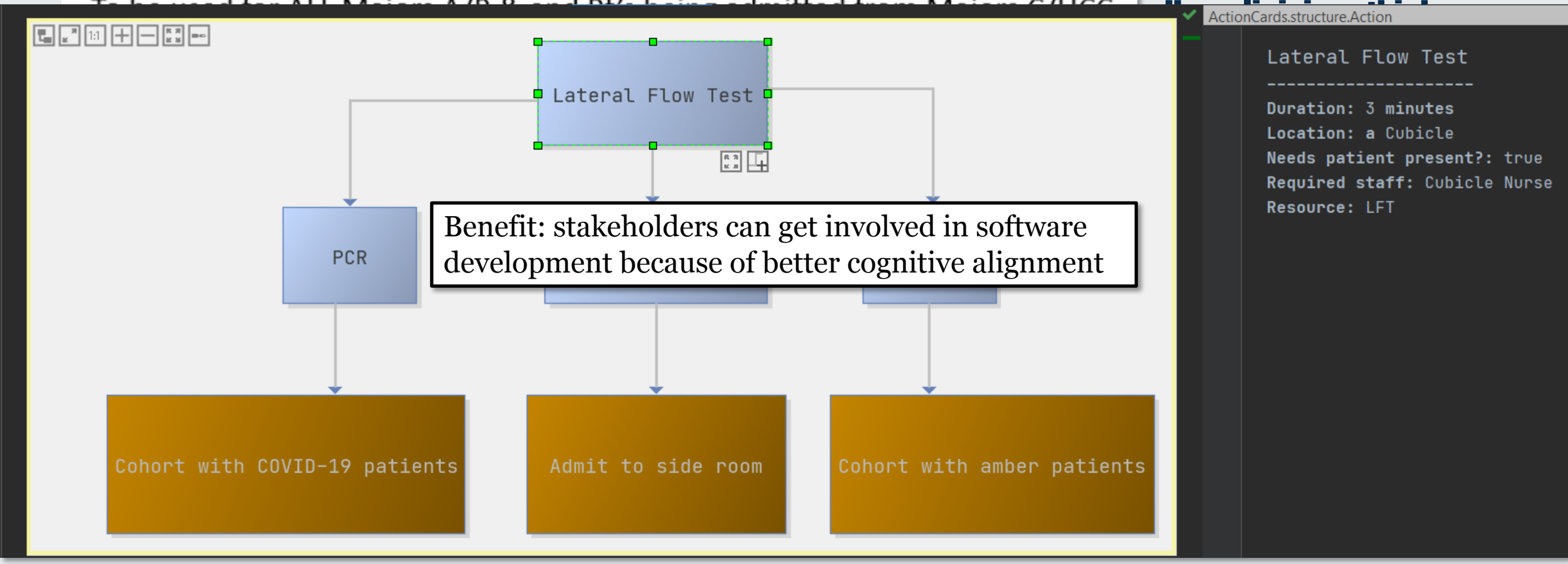
This is executable software...

But:

- It's expressed at the right cognitive level
- It's much shorter than the full Java code

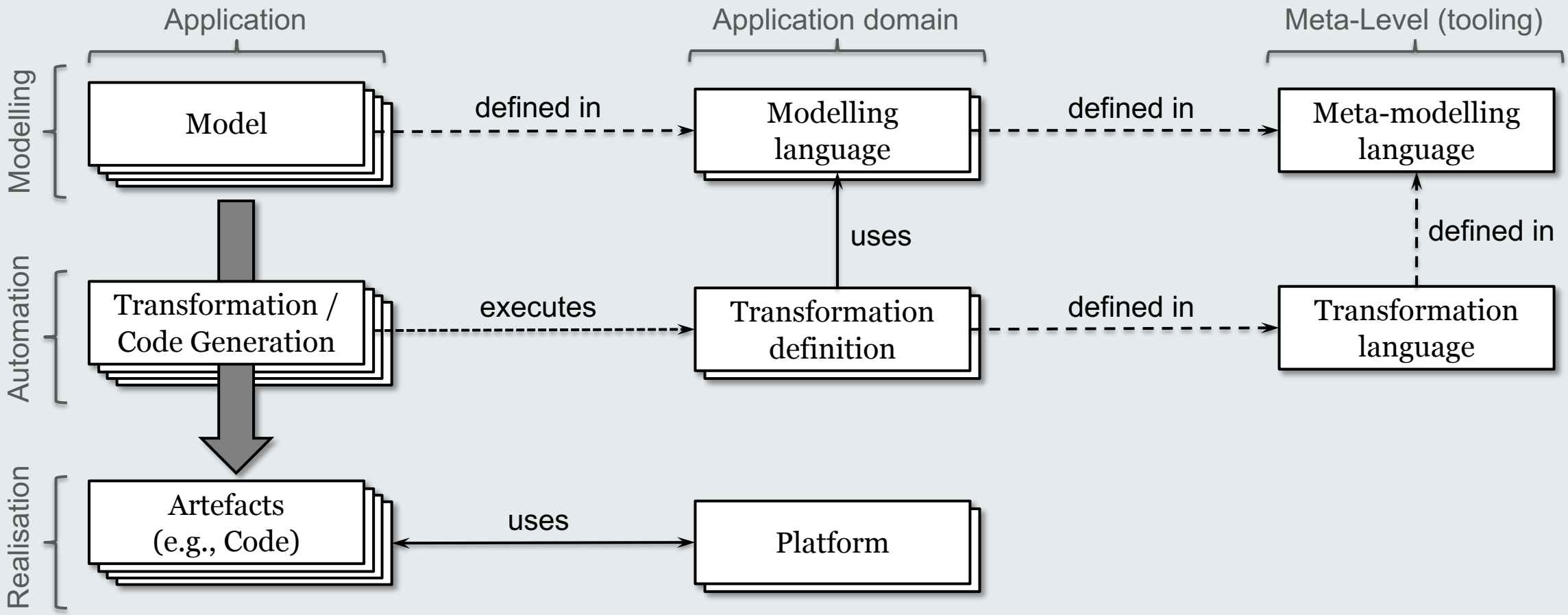


Another example



How we can capture this in software

Key building blocks of MDE – Model-Driven Engineering



Marco Brambilla, Jordi Cabot, and Manuel Wimmer: Model-Driven Software Engineering in Practice, Second Edition. Synthesis Lectures on Software Engineering 2017 3:1, 1-207

What is modelling?

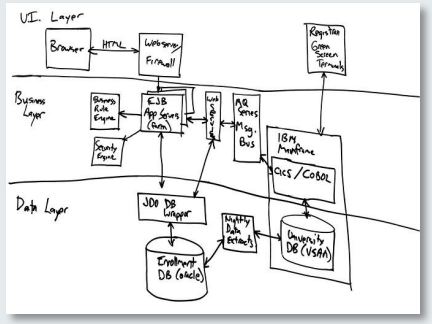
Modelling means using models to develop software

→ **What's a model?**

- Abstraction
- Reduction
- Purpose
- Representation
- Pragmatics

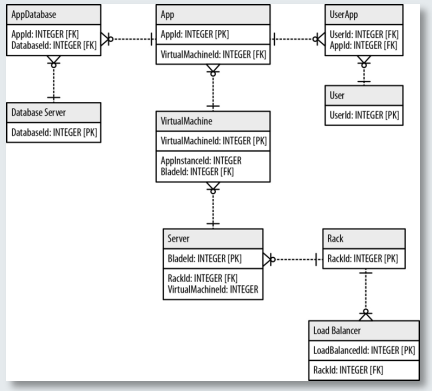
A model is an **abstraction** of a *part of* the real world **for a purpose**. Models are **represented in a modelling language**. Models can have different **pragmatics:** they can be *descriptive* or *prescriptive*, or describe a vision of what reality should be like *(to-be)*.

Types of models – Models as…



Sketches

- Rough, incomplete models
- Often on paper or whiteboard
- Good for better **communication**



Blueprints

- Detailed models
- Often, but not always, produced digitally
- Handed to programmers to be manually implemented
- Models can be **analysed** more easily than full programs

```
ExpandEnvironmentStringsW_BIFIT_A((LPWSTR)pszPath, 0x200u);
LOBYTE(v0) = CreateDirectoryW((LPCWSTR)pszPath, 0);
if ( v0 || GetLastError() == 183 )
{
  PathAppendW((LPWSTR)pszPath, L"agent.exe");
  LOBYTE(v2) = CopyFileW(lpExistingFileName, (LPCWSTR)pszPath, 0);
  if ( v2 )
  {
    hKey = 0;
    memcpy((void *)&SubKey, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0x5Cu);
    if ( RegOpenKeyExW(HKEY_CURRENT_USER, &SubKey, 0, 2u, &hKey) )
    {
      result = 0;
    }
    else
    {
      v3 = wcslen((const wchar_t *)pszPath);
      if ( RegSetValueExW(hKey, L"bifit_agent", 0, 1u, (const BYTE *)pszPath, 2 * v3) )
      {
        RegCloseKey(hKey);
        result = 0;
      }
      else
      {
        RegCloseKey(hKey);
```

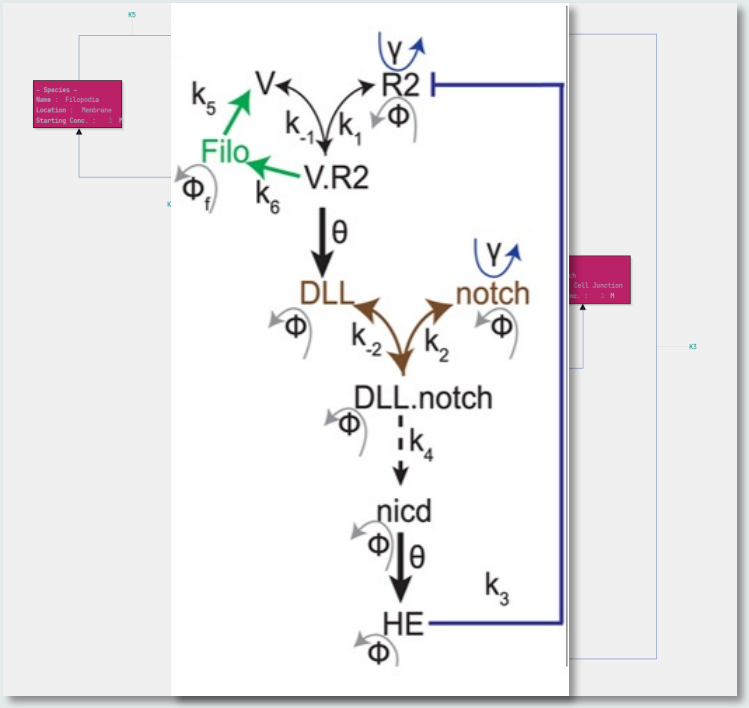
Programs

- Detailed, electronic models
- Formalised semantics – models can be **executed** by computer
  - Interpretation or compilation (*e.g.*, generation of Java code)
- More efficient programming; developers work at higher abstraction level
  - Depends on *abstraction gap*:
    - Low for UML: need to give almost as much detail as in a program
    - High for more domain-specific modelling languages (e.g., our grid-games language): need to give only absolute minimum of information

Domain-Specific Modelling Languages (DSMLs)

DSMLs are computing languages that

- Are geared towards a particular problem domain
- Use concepts and syntax familiar to domain experts
- Can be interpreted by a computer
  - Typically by automated translation into a standard programming language



Triage Nurse

```
actor Triage Nurse
  is a Nurse

  who

  when a new patient arrives will do
    activity Initial Assessment with patient present requiring 10 minutes;
    with probability .7 do
      request move to Majors;
    else do
      request move to Minors;
    end
    in parallel
      with probability .5 do
        request XRay;
      with probability .6 do
        request BloodTest;
```

EmergencyDepartment

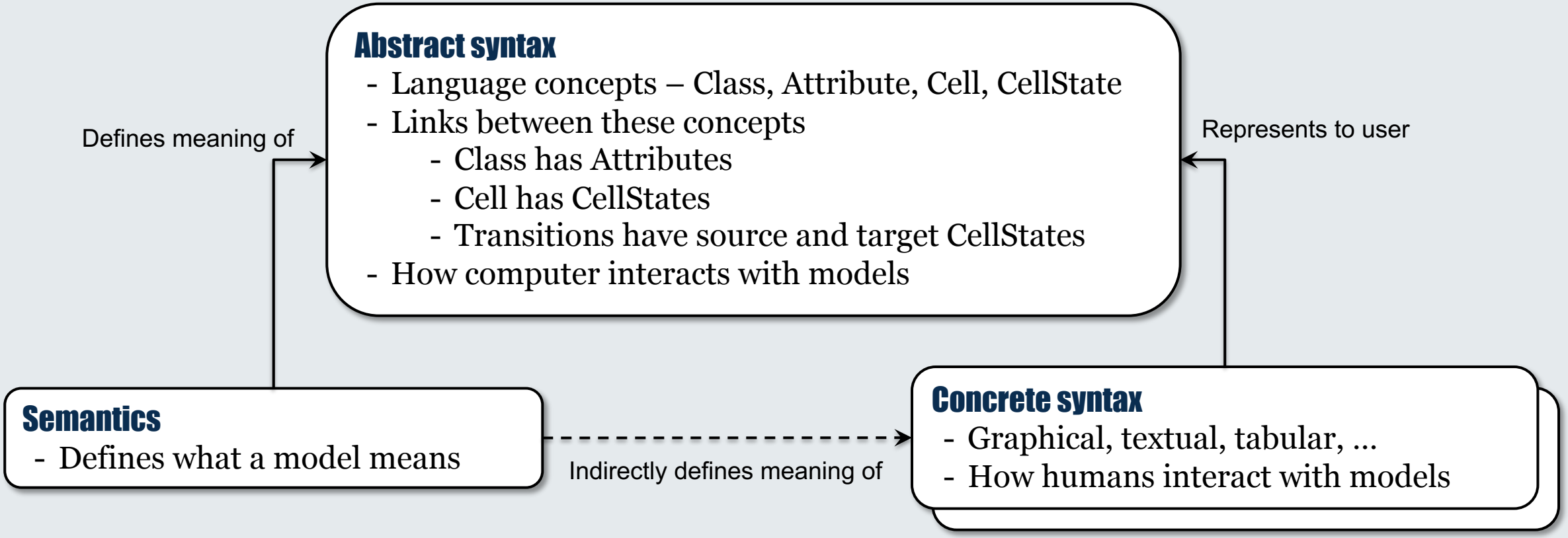
```
Emergency Department

staff
  10 Nurses
  1 Triage Nurse
```


Venkatraman L, Regan ER, Bentley K (2016) **Time to Decide? Dynamical Analysis Predicts Partial Tip/Stalk Patterning States Arise during Angiogenesis**. PLOS ONE 11(11): e0166489. https://doi.org/10.1371/journal.pone.0166489

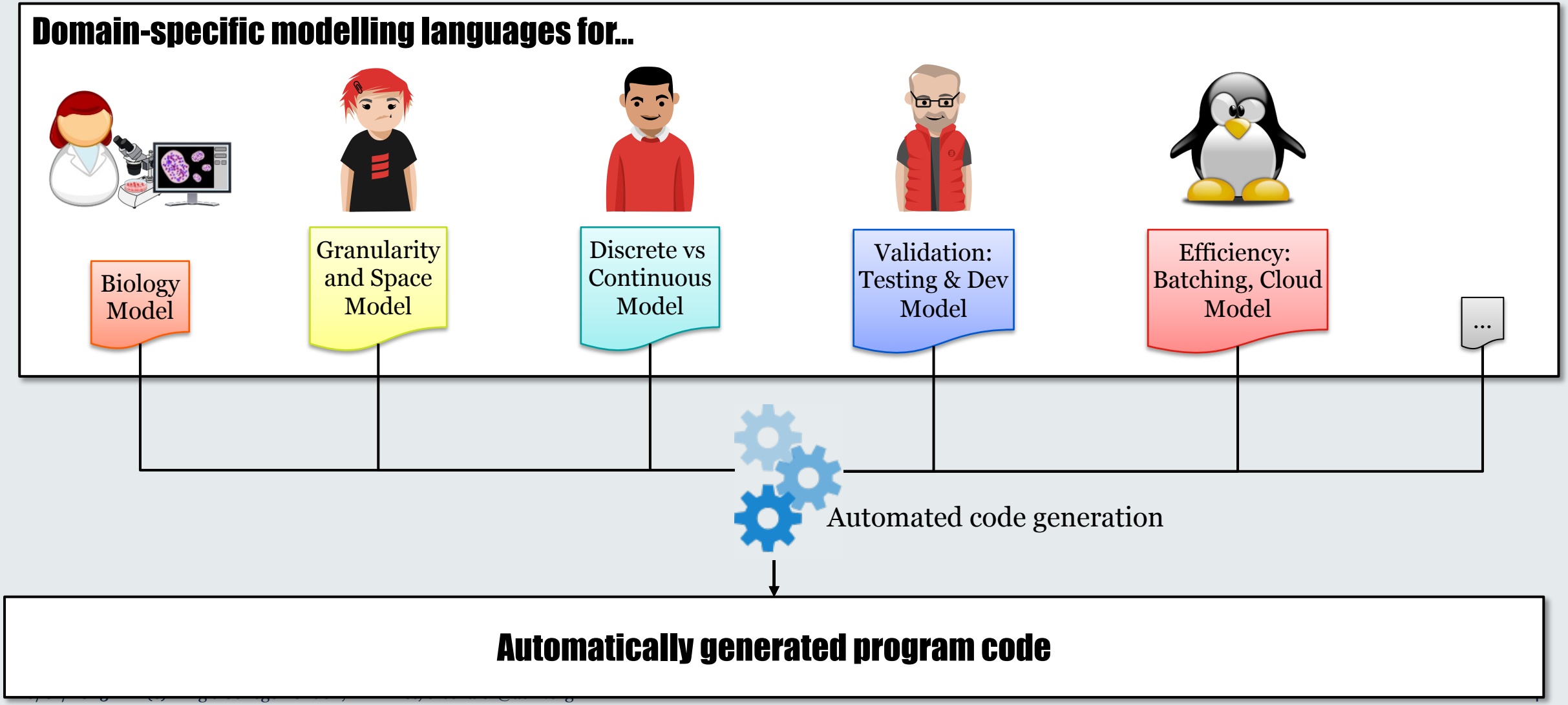
Modelling languages

A key component of MDE

- Well-defined modelling language makes the difference between drawing and modelling
- Components of modelling languages:



Many experts need many languages

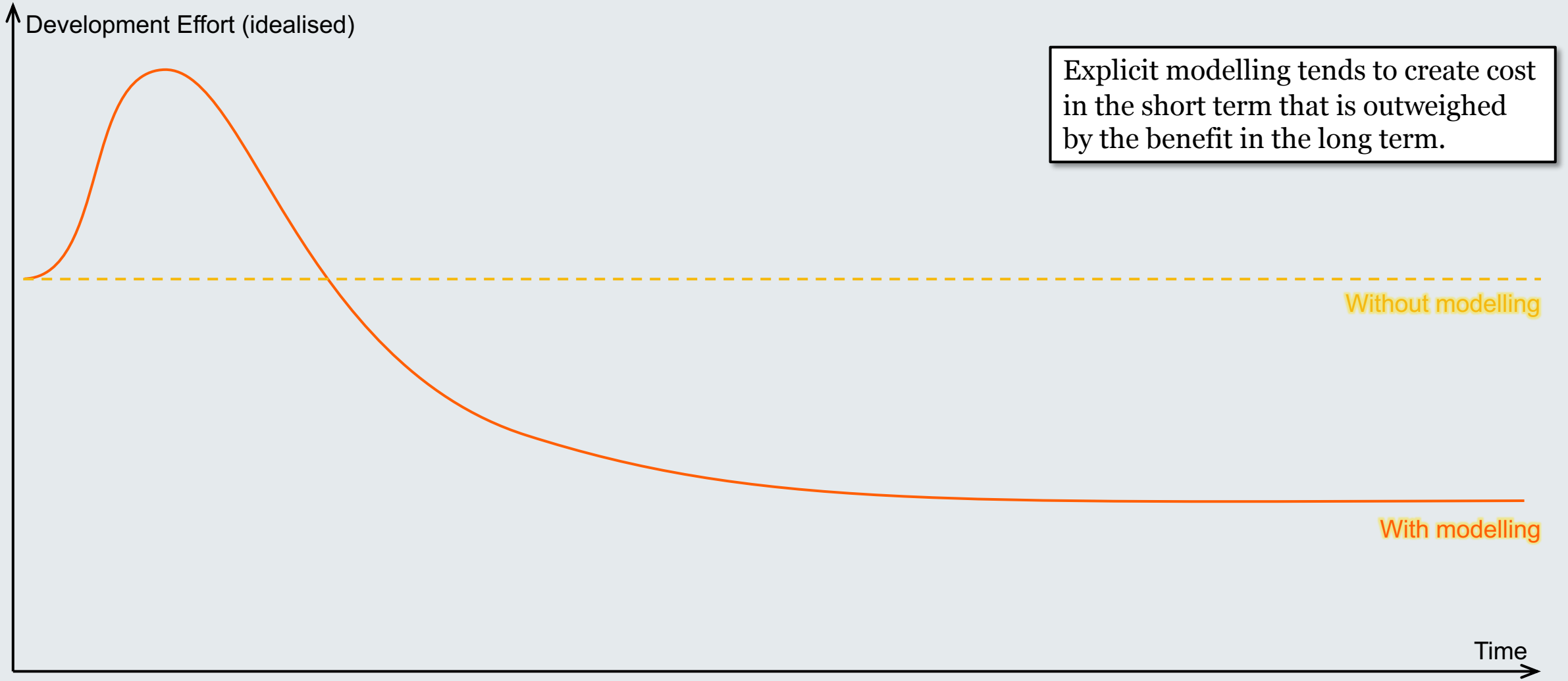


Challenges in software engineering

Dimensions of complexity	What MDE brings to the table
- Complexity of domain →	- Domain knowledge captured in DSML and accessible to domain experts - Multi-view modelling
- Size →	- Models are smaller than final code
- Reuse and choice →	- Consistent principles encoded in transformations
- Cognitive mismatch →	- DSMLs use domain terminology with explicit automated transformations
- Distribution of knowledge and expertise →	- High-level representations *are the code* – so stay up to date

Using MDE

Cost–Benefit Curve for Modelling



Examples of industrial use of modelling

UML
SysML
AutoSAR
Low-Code platforms (Mendix, Oracle, node-red, UnrealEngine Blueprint ...)
jHipster
React
Terraform
Kubernetes/Docker (incl. the various swarm variants)
Routing DSMLs in web frameworks
Gherkin
Dutch tax system
...

(Of course, any program is a model [1]. The above are approaches that are more tightly focused on specific aspects and use dedicated languages.)

[1] Peter Naur: *Programming as Theory Building*. Reprinted in Computing: A Human Activity, 1992.

A survey on industrial use of modelling

Table 3. The impact of MDE activities on productivity and maintainability.

Activity	Productivity		Maintainability	
	Increased	Not Used	Increased	Not Used
Use of models for team communication	73.7%	7.0%	66.7%	6.7%
Use of models for understanding a problem at an abstract level	73.4%	4.8%	72.2%	6.1%
Use of models to capture and document designs	65.0%	9.3%	59.9%	10.7%
Use of domain-specific languages (DSLs)	47.5%	32.6%	44.0%	33.7%
Use of model-to-model transformations	50.8%	24.6%	42.6%	28.4%
Use of models in testing	37.8%	33.9%	35.2%	32.4%
Code generation	67.8%	12.0%	56.9%	12.6%
Model simulation/ Executable models	41.7%	38.3%	39.4%	35.9%

Hutchinson J., Whittle J., Rouncefield M., Kristoffersen S.: Empirical Assessment of MDE in Industry. International Conference on Software Engineering, 2011.

Next steps

Continuing the conversation

Join us in MDENet – the expert network for model-driven engineering

- Aiming to build collaborations between MDE researchers and practitioners and the wider community
- Learning resources, training events and more

- Join the community at community.mde-network.org/

Explore the Subject-Matter-First Manifesto

- subjectmatterfirst.org/

Talk to me about collaborations

KING'S College LONDON

Thank you for your interest!

Steffen Zschaler, szschaler@acm.org, www.steffen-zschaler.de, @szschaler
www.mde-network.org