IARIA

The Fourteenth International Conference on
Software Engineering Advances – ICSEA 2019

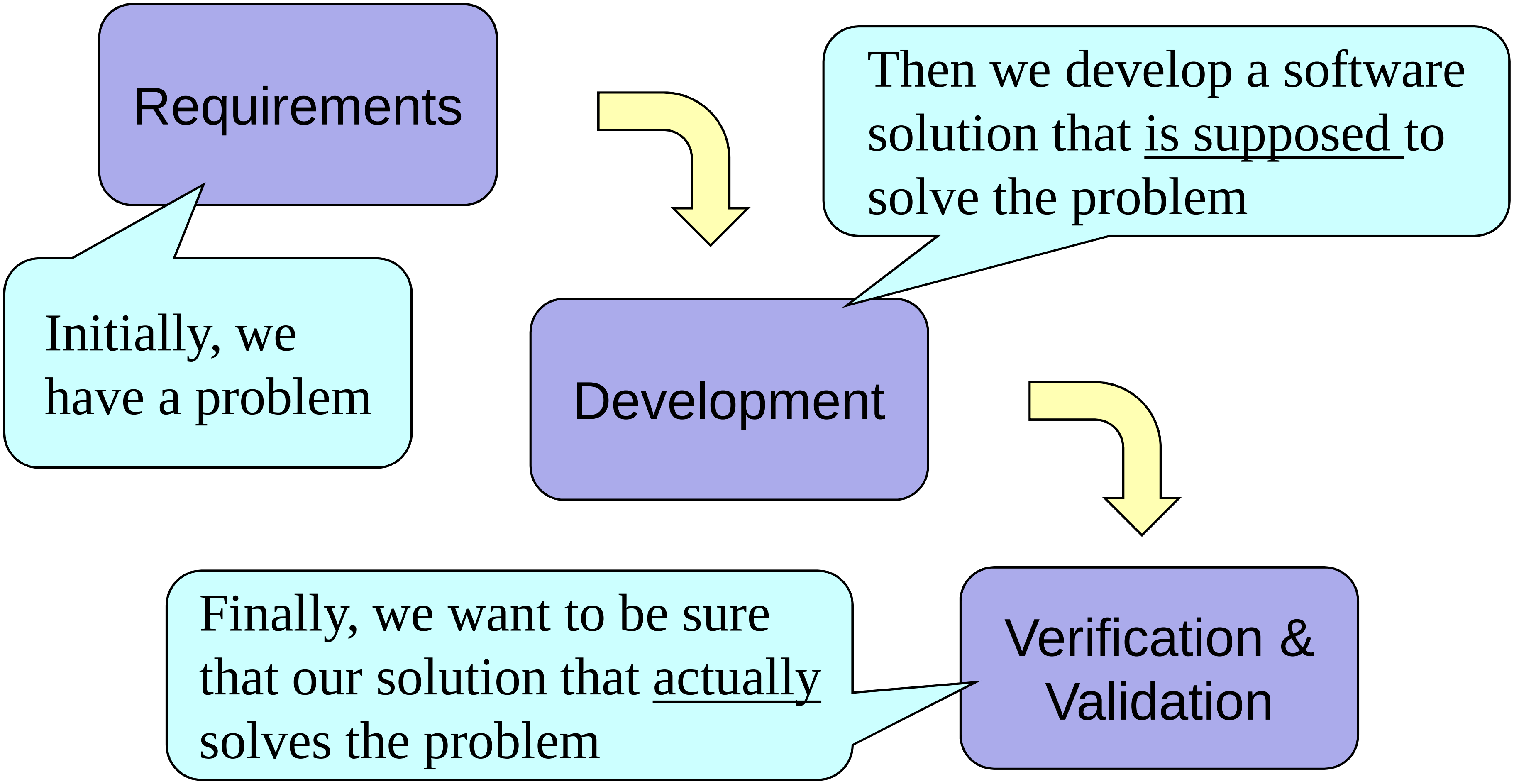
Trends in software development and verification

Challenges on Performance, Safety and Requirements Conformance

The panel

- Luigi Lavazza, Università degli Studi dell'Insubria, Italy (Moderator)
- Martin Zinner, Technische Universitaet Dresden, Germany
- Radek Koci, Brno University of Technology, Czech Republic
- Jos van Rooyen, Huis voor Software kwaliteit B.V. , The Netherlands

Whatever life-cycle you adopt, these core activities are necessary



Requirements

Initially, we have a problem

Development

Then we develop a software solution that is supposed to solve the problem

Verification & Validation

Finally, we want to be sure that our solution that actually solves the problem

Topics

- Of course, there are many issues connected with software development. Here we shall concentrate on just a few of them.
- Luigi:
  - Requirements modelling, tacit requirements, addressing quality assurance (from coding to deployment)
- Martin:
  - Data mining and related ana;lysis techniques to improve our knowledge of software development
- Radek:
- Jos:
  - How to verify quality of software in newer areas, such as self-driving cars or Virtual/augmented reality?

Future of Validation

Jos van Rooyen

identify

THE QUALITY DRIVER

- Employed at Identify as partner / principal consultant
- 30 years in software testing & quality management
- Co-author TestGrip, TestFrame, Project de Baas, Quality Supervision, Textbook; “Aan de slag met software testen”, Cleantxt, Test Automation Architecture (available soon)
- Test expert online magazine Computable
- Publication areas; Test process Improvement, BI-testing, Test automation, Test Education, Risk Based Testing, Quality Supervision
- Member NESMA working party; Metrics in Contracts
- Visiting lecturer Universities of Applied Science
- Member advisory board Hogeschool Utrecht
- Member of several working parties Dutch Testing Society
  - Member of the board
  - Test Education universities of applied science

Introduction

- Everybody knows that the development of technology is emerging very fast. New areas are developed and implemented into organisations and the society. Think about self-driving cars, enhancement of a chain between organisations or the applicability of robotics in daily life to support elderly people for instance.
- How reliable is the quality of these new areas? How reliable are the results?
- Looking at these developments, an interesting question is: how to validate these applications to be sure that the application is delivering the same high level results every time.
- Standard technologies are not sufficient anymore to verify the quality of these new developments.
- What kind of validation techniques are necessary to demonstrate the quality of the previously mentioned developments?

Introduction

- At the same time attention for validation is decreasing:
  - Companies research is shattered
  - Large consultancy firms are not interested anymore
    - Completely different 10 years ago
    - No student programs anymore for research and development
  - Universities???

Introduction

- The question is: what to do about it?

- Ideas are:
  - Make research operational as soon as possible like applying formal methods on a broad scale
  - Embed software quality into education (just started in the Netherlands)
  - Increase interest in software quality???

- Some ideas around techniques:
  - Datamining
  - Predict the quality of software in combination with business processes
  - Applying artificial intelligence to improve quality assurance
  - The applicability of VR/AR to simulate business processes

Introduction

Some other ideas?

Università degli Studi dell’Insubria
Dipartimento di Scienze Teoriche e Applicate

Trends in software development and verification

Challenges on Performance, Safety and Requirements Conformance

Luigi Lavazza

Dipartimento di Scienze Teoriche e Applicate

luigi.lavazza@uninsubria.it

My contribution

- Challenges on Performance, Safety and Requirements Conformance

Software must satisfy requirements.
There are both explicitly stated requirements and implicit (tacit) requirements.

What is the current state of requirements specification practices?

- To talk about Requirements Conformance, we need that requirements are precisely defined and agreed upon.
- Are requirements satisfactorily modelled today?
  - Via stories
  - Via UML
  - Via other specific notations
    - E.g., goal-oriented notations like KAOS
  - …
- Or we have the usual collection of heterogeneous documents, mainly written in natural language?

Implicit (tacit) requirements

- Whatever functionality and application domain, there are some qualities that any piece of software should have.
  - Easy to understand
  - Easy to maintain
  - Reasonably efficient
  - Reasonably easy to use
  - Safe
  - . . . . .

- Summarizing, we want that our code is of good quality.

Addressing quality

- The coding phase should deliver only high-quality code
  - I mean: there are some kinds of trivial defects that should never appear in a piece of software code released in 2019

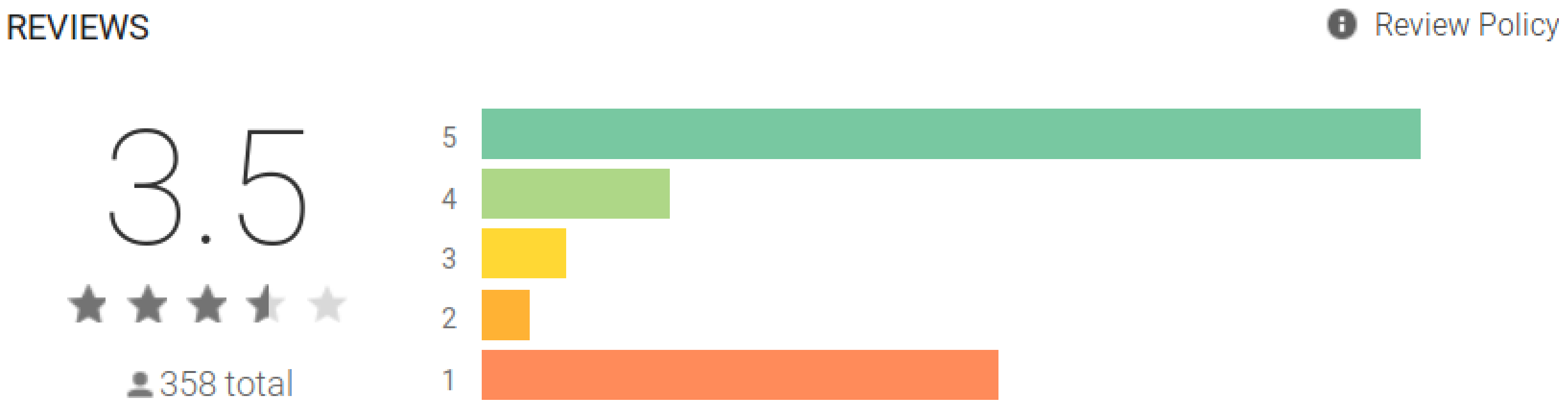
- There are some easily accessible techniques that support quality assurance and are not used on a regular basis.
  - Static analysis lets programmer find several types of defects
    - Static analysis is not used in industrial development processes
    - Static analysis is not used in the development of OSS

Traditional verification and validation

- Nothing special to say here

Addressing quality

- Testing mobile applications involves so many different operating conditions that going beyond lab testing is practically necessary.
- A typical example from the Google Play store:
  - a good app, that does not work properly on some devices.



- Beta testing is now often fragmented and distributed: crowd testing.

Finally…

- Are we talking about the software process?
- Well, yes. To achieve quality you have to organize you process properly!
  - Techniques and tools alone are not sufficient.

Center for Information Services and High Performance Computing (ZIH)

Panel Presentation: Role of Data Mining in Software Development and Verification

ICSEA 2019, Valencia (Spain)

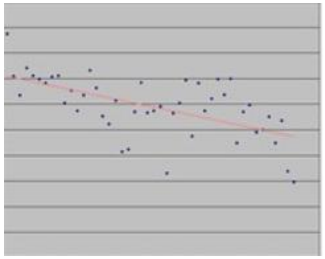
November 27, 2019

Martin Zinner (martin.zinner1@tu-dresden.de)

Definition

Data Mining is the process of extraction of hidden patterns and knowledge from large datasets:

- Involves methods of machine learning, statistics, and database systems.
- Is the analysis step of the “knowledge discovery in databases”.



Uses methods from[15]-[21]:

- Regression analysis (estimates the relationship between variables)
- Neural networks
- Cluster analysis
- Genetic algorithms
- Decision trees
- Outlier detection analysis

Overview

Data mining technology:

- Can accelerate the speed of software development
- Can find valuable data in the databases[13]

Software developers:

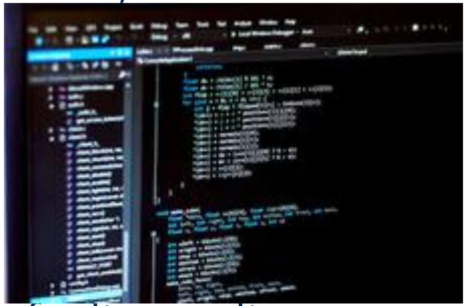
- Extract the required data information from large amounts of data[14][19]-[21]
- Process the collected data

Data Mining Technology [1][14][19]-[21]

Data Mining in Programming:

Developers need information regarding:

- Code structure, similar functions, and patterns which can be reused.
- Static rules for reusing some patterns for example class methods, inheritance relationship, etc.



Software fault detection:

- Extract the required data information from program code
- Compare the running of the software with expected and /or faulty results

Software management:

- Organizational management
- Version control

Data Mining Technology – cont. [2]

Mining Software Repositories:

- Discover hidden patterns and trends
- Use repositories to guide prediction and decision taking processes

Historical Repositories[19][21]:

- Bug repositories (Bugzilla[7], JIRA [8])
- Development collaboration sites (StackOverflow[9])

Code Repositories[19][21]:

- Code bases (SourceForge[10], GoogleDeveloper[11])
- Project ecosystems (GitHub[12])

Runtime Repositories:

- Crash reports
- Field logs
- Execution traces

Example CAR-Miner: Association Rule [3][4]

```
1.4: try{
…
1.8:    statement=conn.create Statement();
1.9:    statement.executeUpdate(“DELETE FROM table1“ );
1.10: conn.commit();}
1.11: catch(SQLException se){
1.13:          logger.error(“Exception occured“);}
1.14: finally{
…
```

Should every connection to be rolled back when SQLException occurs ?

Missing “conn.rollback()”

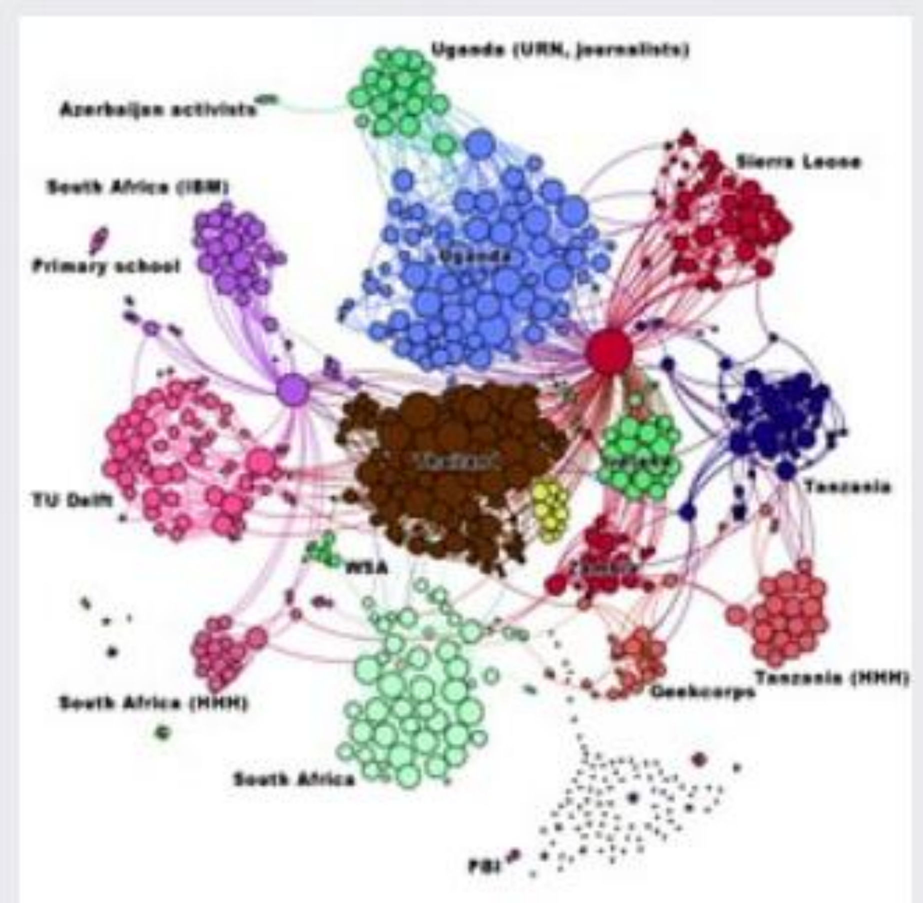
Cluster Analysis [6]

K-means clustering[21]:

- — Aims to partition n observations into k cluster
- — Each observation belongs to the cluster with the nearest mean

Outliers:

- — Big distance to their cluster center



Tools for Data Mining

An excerpt:

- RapidMiner
- Weka
- Knime
- Spark
- R
- Python

For specific tools for Software Engineering see [5]

Bibliography

[1] Deng, Fengxian. "Research Progress on Software Engineering Data Mining Technology." *2015 International Conference on Education Technology, Management and Humanities Science (ETMHS 2015)*. Atlantis Press, 2015.

[2] Gómez, María “Mining Software Data” Software Engineering Course – Summer Semester 2017 URL https://www.st.cs.uni-saarland.de/edu/se/2017/files/slides/15_Mining_Sw_Data.pdf Retrieved Nov. 2019

[3] Xie, Tao, et al. "Data mining for software engineering." *Computer* 42.8 (2009): 55-62.

[4] Thummalapenta, Suresh “CAR-Miner: Mining Exception-Handling Rules as Sequence Association Rules” https://pdfs.semanticscholar.org/59d3/ae001440d57b99953dbcc5616ecaed197792.pdf Retrieved Nov. 2019

[4] Xie, Tao, et al. "Data mining for software engineering." *Computer* 42.8 (2009): 55-62.

[5] Dhamija, Ankit, and Sunil Sikka. "A review paper on software engineering areas implementing data mining tools & techniques." *International Journal of Computational Intelligence Research* 13.4 (2017): 559-574.

Bibliography – cont.

[6] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." *ACM computing surveys (CSUR)* 41.3 (2009): 15.

[7] https://www.bugzilla.org/ Retrieved Nov. 2019

[8] https://en.wikipedia.org/wiki/Jira_(software) Retrieved Nov. 2019

[9] https://en.wikipedia.org/wiki/Stack_Overflow Retrieved Nov. 2019

[10] https://en.wikipedia.org/wiki/SourceForge Retrieved Nov. 2019

[11] https://en.wikipedia.org/wiki/Google_Developers Retrieved Nov. 2019

[12] https://en.wikipedia.org/wiki/GitHub Retrieved Nov. 2019

[13] https://www.lis-solutions.es/blog/herramientas-del-data-mining/ Retrieved Nov. 2019

[14] https://abogadotic.com/2019/04/01/data-mining-y-su-proceso-de-valor-para-la-gestion-empresarial/ Retrieved Nov. 2019

[15] https://en.wikipedia.org/wiki/Regression_validation Retrieved Nov. 2019

Bibliography – cont.

[16] https://developer.oracle.com/databases/neural-network-machine-learning.html Retrieved Nov. 2019

[17] https://phys.org/news/2016-11-genetic-algorithm-carbon-dioxide-sponge.html Retrieved Nov. 2019

[18] https://study.com/academy/lesson/what-is-a-decision-tree-examples-advantages-role-in-management.html Retrieved Nov. 2019

[19] https://pixabay.com/photos/code-html-digital-coding-web-1076536/ Retrieved Nov. 2019

[20] https://www.shutterstock.com/image-illustration/version-control-system-form-binary-code-351408083?irgwc=1&utm_medium=Affiliate&utm_campaign=Pixabay+GmbH&utm_source=44814&utm_term=https%3A%2F%2Fpixabay.com%2Fphotos%2Fdevops-business-process-improvement-3155972%2F Retrieved Nov. 2019

[21] https://search.creativecommons.org/ Retrieved Nov. 2019

Thank you for your attention!

Utilization of formal models for continuous simulation/formal analyzes of the system under development.

Radek Kočí

Brno University of Technology, Faculty of Information Technology
Czech Republic

koci@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

ICSEA 2019, 24.-28.11.2019, Valencia, Spain

Formal methods

Formal specification

- predefined rules for determining the meaning of specifications
- written in formal languages
- supported by tools

⇒ enable rigorous software development

Formal description

- specifying requirements and desired properties
- modeling internal behavior
- the description is typically at certain level of abstraction
- precise, consistent and unambiguous

Formal languages

Formal languages

- algebraic specification techniques (CASL)
- rewriting systems (OBJ3)
- Model-oriented languages (Z, VDM)
- UML + OCL; MOF + Alf language
- Petri nets
- logics
- . . .

Formal methods

Formal specification

- formal specification let designers use abstractions and reducing the conceptual complexity of the system under development
- formal specification formalizes the statements describing element properties
- precise formulation of statements permits machine manipulation
- a more sophisticated form of validation and verification that can be automated using tools
- the specification may be mechanically transformed into another, more detailed, one, and, eventually, into executable program

Formal methods

Formalization properties

- (+) formal methods can be beneficial even if no formal verification is used at all – since since the rigorous specification is required the designer has to do the job more thoroughly, reaches a better understanding of the problem and it leads to better solution
- (-) can be difficult to understand not only for users but also for developers

General problems

- a formally verified program is only as good as its specification
- it is very easy to create a wrong specification that does not meet the user needs (requirements)

Formal methods

- How to validate documents/formalized documents against user's real needs?
    - only the user can say
    - a combination of the formal notation and prototyping

- Formal methods can be difficult to understand
    - requirements specification has to be clear and comphrehensible to users as well as developers
    - a possibility of formal notation as well as graphical modeling

⇒ formal models that can be simulated, graphically represented, and formally processed

Simulation Driven Development

Motivation

- reduce the gap between real needs and specified needs to sofware system under development
- combination of semi-formal and formal models

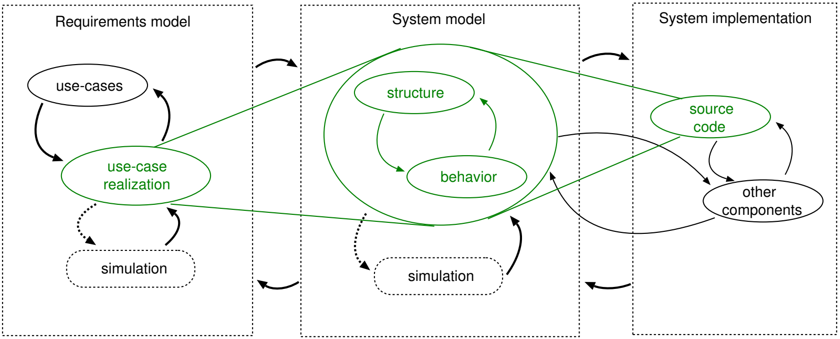
Model continuity

- elimination of the overhead caused by creating models at different level of abstraction
- continuous incremental development of models
- models can work in live system
- no need of implementation or code generation (mainly for validation purposes)

Simulation Driven Development

Essential parts of the systems are presented through simulation (formal) models

- simulation
- continuous validation


Requirements model
use-cases
use-case realization
simulation
System model
structure
behavior
simulation
System implementation
source code
other components