Panel on SOFTWARE ENGINEERING

Software Engineering Achievements and Their Evolution Transcending Multiple Disciplines: Celebrating 50 Years

Stephen Clyde, Utah State University, USA
Radek Koci, Brno University of Technology, Czech Republic
Luigi Lavazza, Università degli Studi dell'Insubria, Italy
Arash Ramezani, University of the Federal Armed Forces Hamburg, Germany
Roy Oberhauser, Aalen University, Germany (Moderator)

Software Engineering

"The application of a
systematic, disciplined, quantifiable approach
to the
development, operation, and maintenance
of software;
that is, the application of engineering
to software."

[ISO SEVOCAB]

"Transcending Multiple Disciplines"

- Computer Engineering
- Systems Engineering
- Computer Science
- Mathematics
- Project Management
- General Management
- Quality Management

Birth Pangs 1967: Call for 1st Software Engineering Conference

1968 NATO conference in Garmisch, Germany

Born in **pain**

50 years ago it was not a “Celebration”

SE Evolution: Boehm's Hegelian View

- 50's: SE is like HW engineering [Thesis]
- 60's: Software crafting [Antithesis]

SE Crisis

50 years

- 70's: Formality & waterfall processes [Synthesis & Antithesis]
- 80's: Productivity & scalability [Synthesis]
- 90's: Concurrent vs. sequential processes [Antithesis]
- 00's: Agility & value [Antithesis and Partial Synthesis]
- 10's: Globalization & SoS [Antithesis and Partial Synthesis]

[Boehm, B.: A View of 20th and 21st Century Software Engineering, ICSE'06]

Some things that brought us further

- Reuse+sharing: Building on tested SW
- Human- and value-centric, iterative processes
- Design patterns
- Integrative testing
- Coding practices
- Tool chains and automation

Current/Upcoming SE challenges

- Containment of defect costs to society
- Ulterior ethics seep into systems (power/manipulation)
  - Dieselgate, Exploits, etc.
- Complex, self-adapting, distributed SoS (IoE/CPS)
  - Risk containment, illusion of control, opaqueness
- IT and operational runtime integrity
  - Misconfiguration, model coherency verification
- SE education and certification
  - Rapid technology cycle challenges
- Maintenance (legacy, abandoned non-supported code)

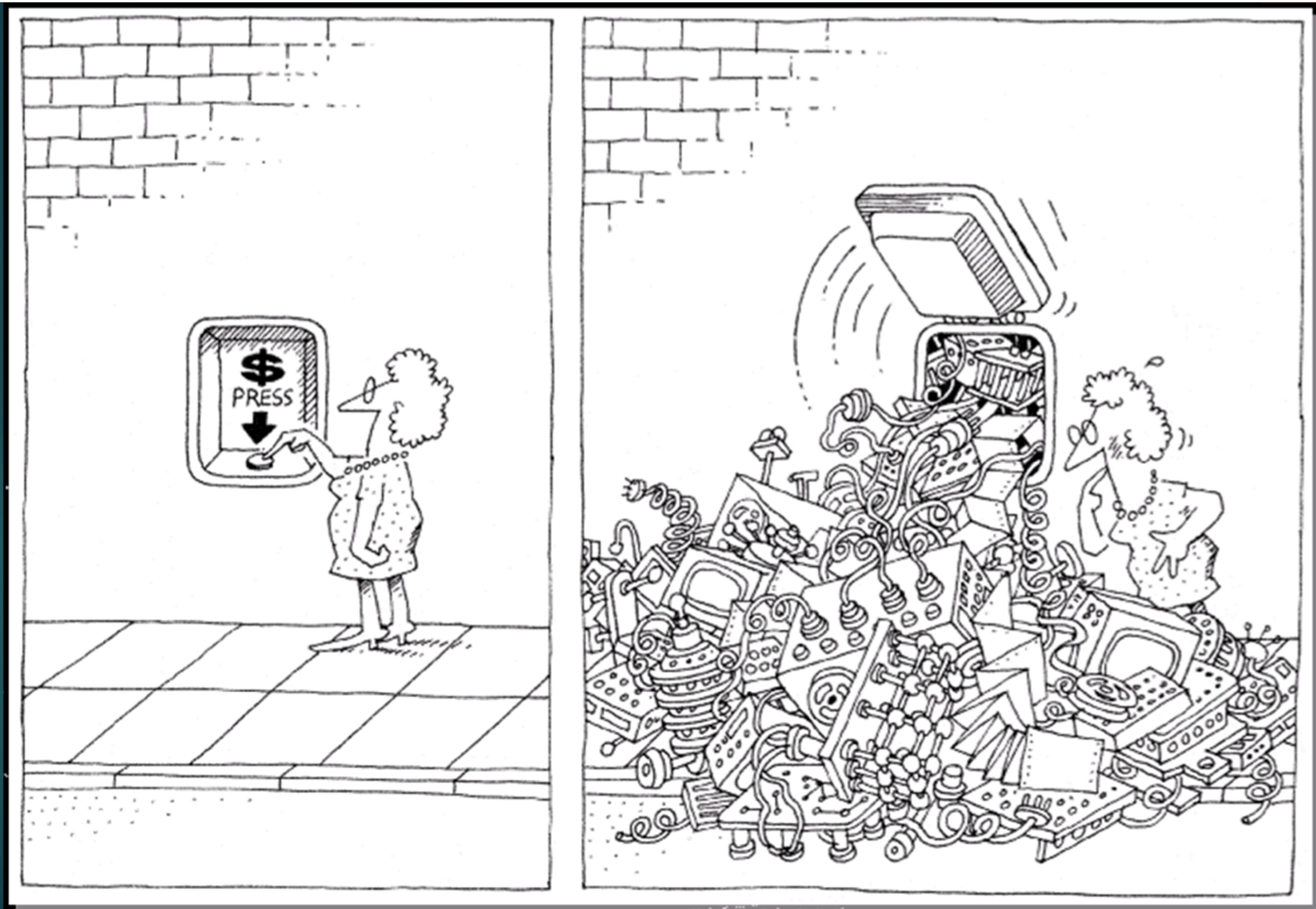
The next 50 years in SE: Predictions

- "Software Eats the World" and its variants
- Increased automation → Automated SE
  - Testing, bug fixing, maintenance, legacy code
- Cognitive processing / Intelligent adaptation
  - Testing, reproducibility challenges
- Attention on security
- Quantifiable quality analytics and assurance
- DIYSW / BYOS

Thoughts on “Software Engineering Achievements and Their Evolution Transcending Multiple Disciplines”

STEPHEN CLYDE

UTAH STATE UNIVERSITY



$
PRESS

Complexity – A Driver for Innovation

- Software systems can be extremely complex
  - Lots of components
  - Lots of “moving” parts
  - Lots of dependencies
  - Lots of stakeholders
- The processes of creating software systems are also complex
- The need to manage complexity as spawned innovations for
  - Conceptual modeling
  - Development languages and tools
  - Development processes

One transcendent contribution:

- "Agile" Methodology
  - Values
  - Principles
  - Practices
  - Processes

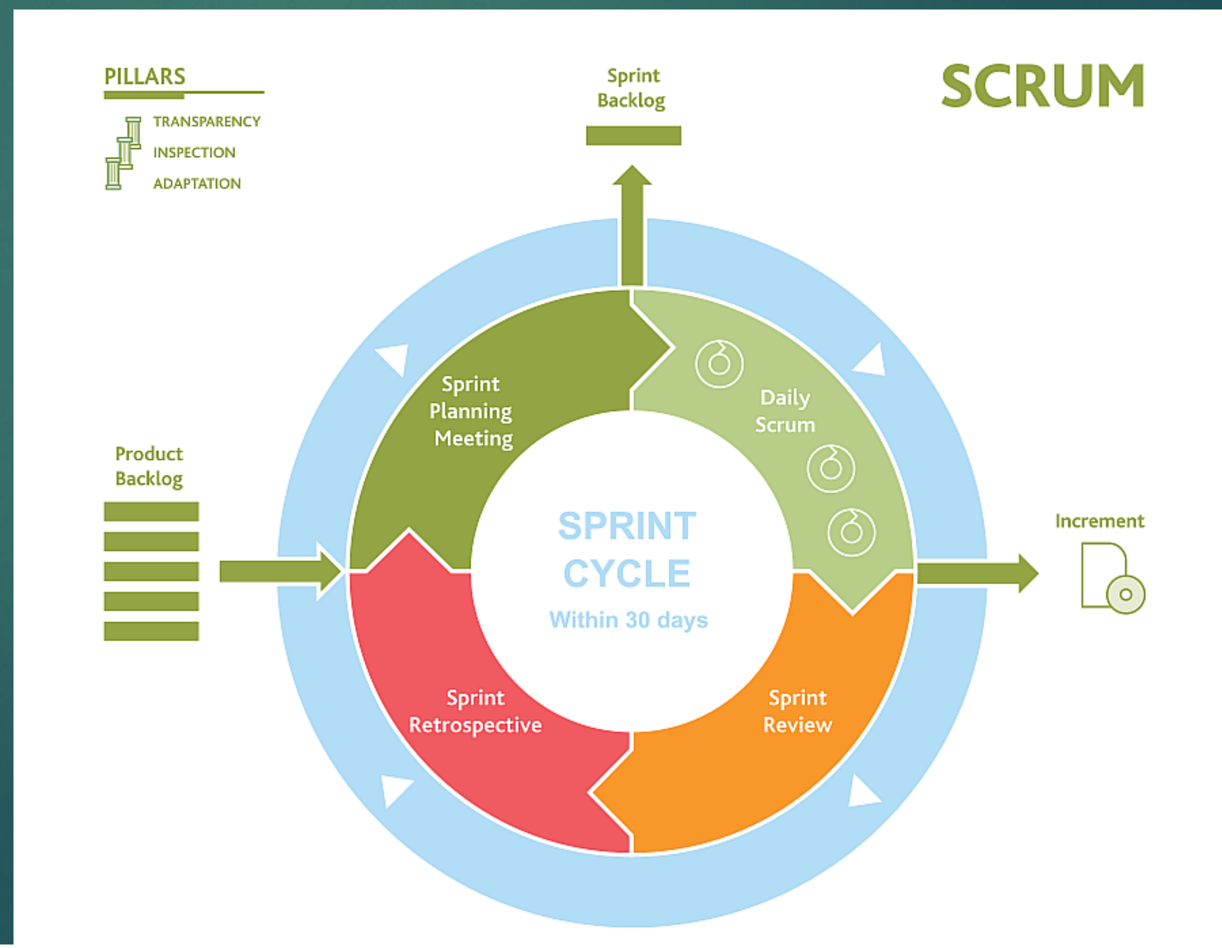
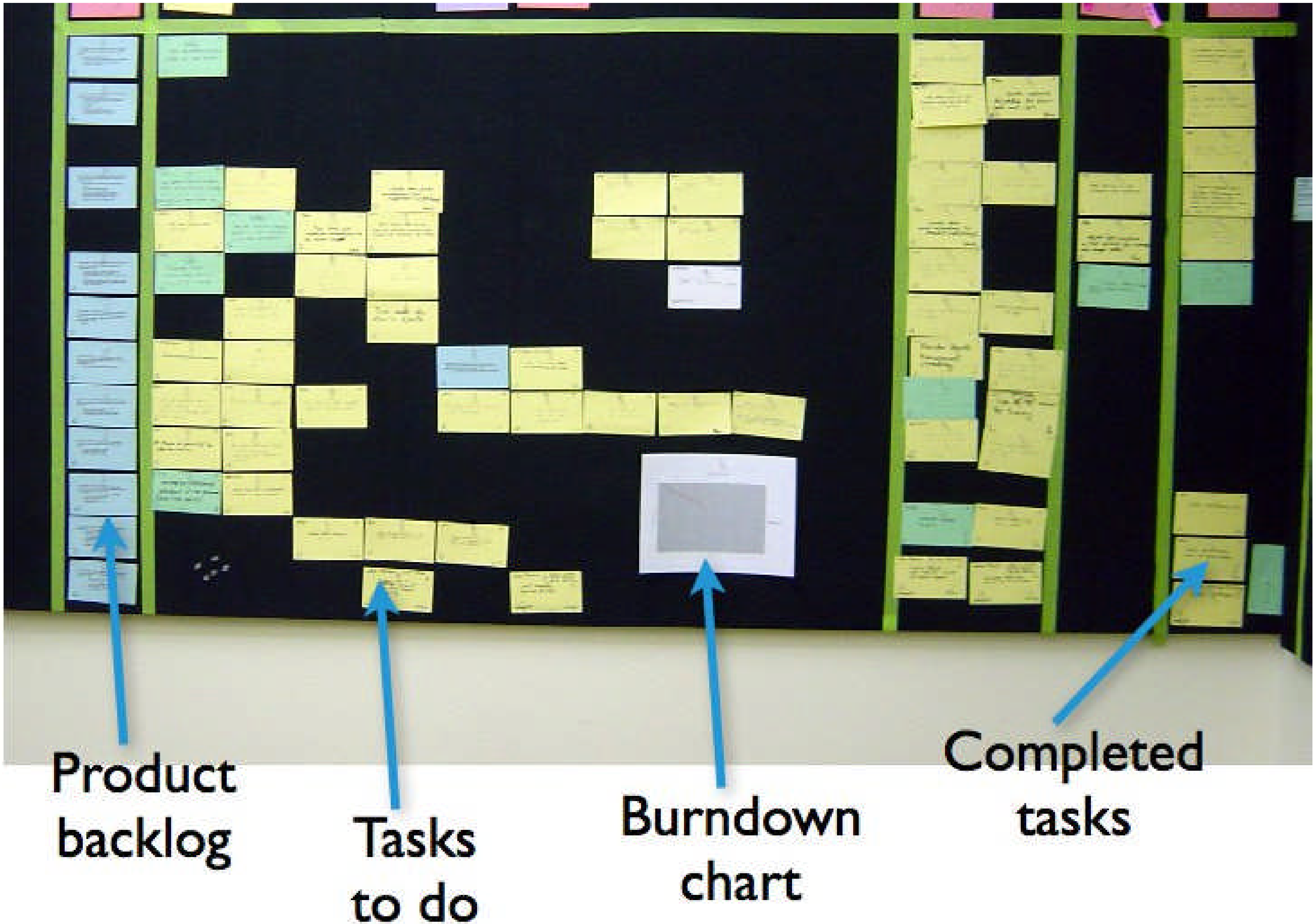


Diagram from Emergn


Product backlog
Tasks to do
Burndown chart
Completed tasks

Stephen: Professional Career Scrum Board

Backlog

- Research in Software Testing
- Sabbatical
- Learning to Paint

To Do

- Build Planning Tool for Hydrology
- Come Up to Speed on Simplicial Complexes
- Design mini-course on S.E. Principles and Practices

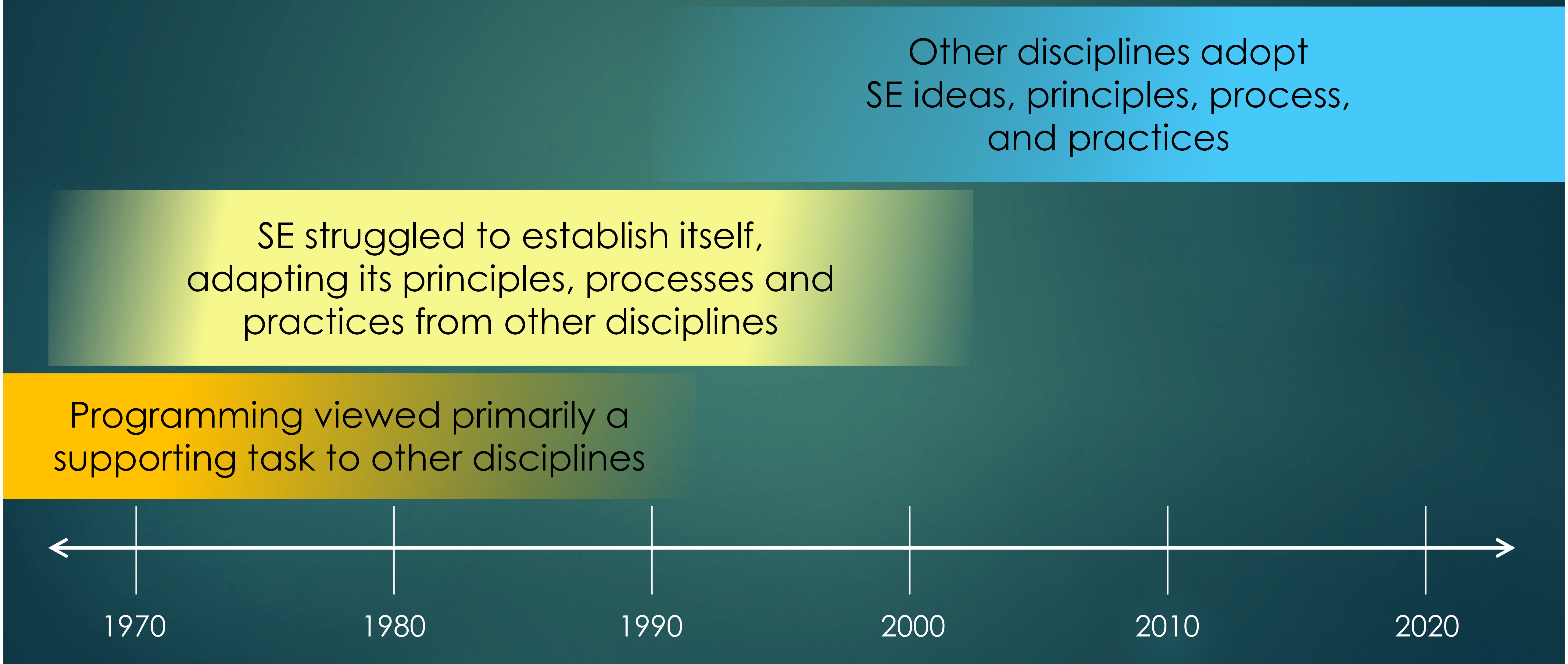
In Progress

- Become a Better SE Teacher
- Research in Core SE Principles
- Research in Comm. Design Patterns
- Research in High-level Aspects
- Enhance Health Information Exchange

Done

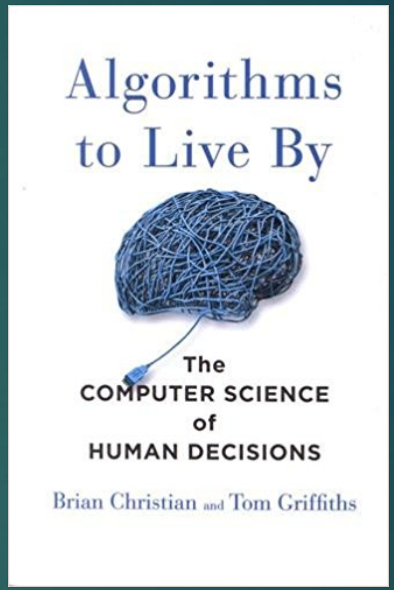
- Ph.D. is CS / Software Engineering
- Commercial Apps Health Care
- Business For Driver Education App
- Comme Apps Transportation
- Commercial Apps

Cross-discipline Retrospective on Software Engineering



Examples

- Algorithms
  - Optimal stopping
  - Explore/Exploit
  - Sorting
  - Caching
  - Scheduling
  - Randomness
  - Networking
  - are more
- Conceptual modeling
- Agile principles, practices, and processes



Questions

What's next?

What ideas or lessons learned from software engineering can be generalized and adapted into other disciplines?

What do software engineers need to do to leverage advances from other disciplines and conversely?

Modeling and simulation in software engineering: Can we effectively involve software models in development processes?

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 2017, 8.-12.10.2017, Athens, Greece

Software engineering

- Software engineering is about managing changes – requirements change
- Identifying and clarifying incomplete and inconsistent requirements, as well as managing changes, are an important part of requirements engineering.
- complex system $\Rightarrow$ a need to describe all aspects of designed systems, **impossible without an abstraction**, i.e., impossible without **models**!

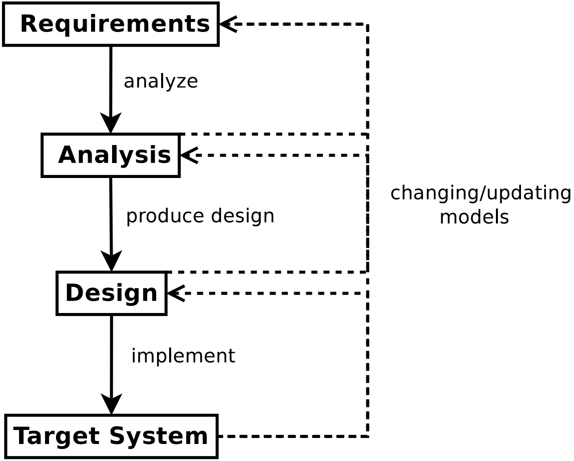
How models can be used

- models can be an "intermezzo" in the process of requirements specification as well as the result of this process
- models can be used for requirements validation – a need for model simulation
- models can be used as prototypes or application – a need for model simulation

⇒ different levels of **abstraction** and **formalization**

Specification documents

- plain text, tabulars, . . .
- semi-formal laguages (diagrammatic notation)
  - diagrams, models like ERD, UML, . . .
  - elements and relationships have formal base, properties or characteristic are decribed informally
  - only simulation of control flows
- formal languages
  - provide higher precision and richer forms of analysis
  - simulation works with all modeled characteristics
  - (but) are usually harder to use and less widely applicable


Requirements
analyze
Analysis
produce design
Design
implement
Target System
changing/updating models

Model Driven Engineering

- models having different levels of abstraction
- Computational Independent Model (CIM) – use cases, activity, sequence, ...
- Platform Independent Model (PIM) – does not include any technology specific details; other UML models, detailed UML models, ...
- Platform Specific Model (PSM) – includes technology specific details, library classes, etc.

The transformation becomes more demanding with a higher degree of abstraction of models

Problem with modeling and changing requirements

- if a requirement is depicted in analysis, it is repeated in design and implementation
- code changes to fix problem with the original requirements $\Rightarrow$ requiring changes to the analysis and design models
- the boundaries between models are blurred, it is not always apparent what needs updating when the code is changed

Modeling is time-consuming activity

Can we eliminate the overhead caused by creating different models and managing the relationship between models and code?

- continuous incremental development of models, no need of transformation
- models combine formal structures with programming languages
- models are simulated in live system under real conditions
- no need of implementation or code generation
  - for efficiency reasons, the resulting models can be transformed into code (code generation)
  - there has to be no difference between code and models from the developer point of view

$\Rightarrow$ model continuity

Thank you for your attention!

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

Past and future of software engineering (nothing less!)

Luigi Lavazza

Dipartimento di Scienze Teoriche e Applicate

luigi.lavazza@uninsubria.it

What is SE?

- Methods
  - How to deal with software development and related activities
- Techniques
  - How the software should be written and structured
- Tools (it would be funny if we did not use computers …)

A few “milestones”

- When I began writing code (1982)
  - Tools: editor, compiler, make, symbolic debugger, sccs, diff
  - Methods: structured programming, modularity principles
- In 1996 (interviews of newly graduated software engineers)
  - Q: What is used in your organization?
  - A: IDE, configuration management
- Today
  - Methods
    - Lifecycles, reuse, refactoring, continuous integration, etc.
  - Techniques
    - Patterns, services, architectures, ...
  - Tools
    - Plenty, covering all development activities, from requirements gathering and modelling to automated testing.

A breakthrough happened in the late 90's

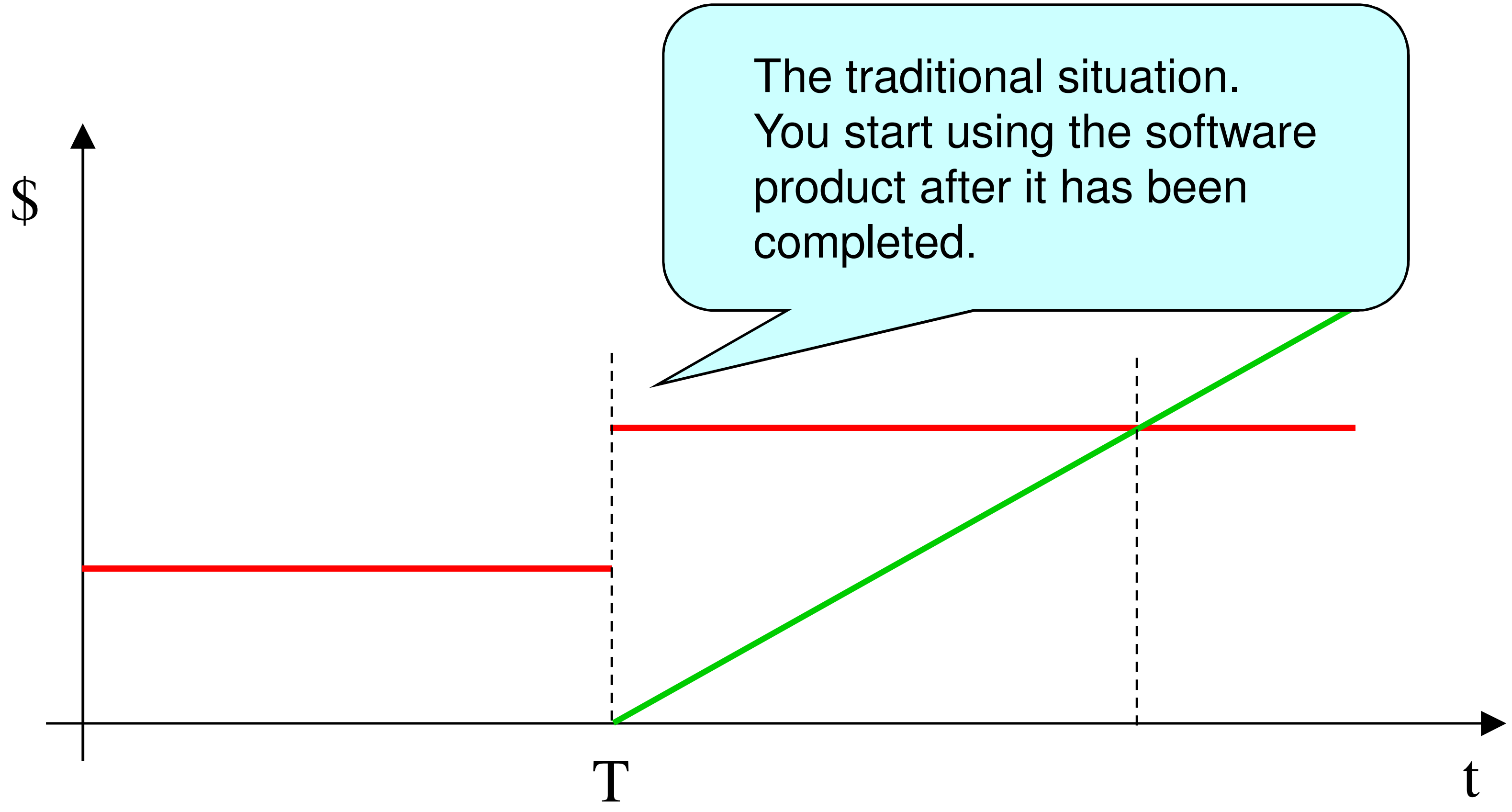
- More and better exchanges between industry and research
  - Demands from developers
    - Better support for coding, simpler and affective processes, best practices, …
  - Proposals from researchers
    - Object-oriented programming, Java, patterns, distributed architectures, …
  - Proposals whose origin I do not know with certainty
    - Continuous integration, devops, agile processes, microservices, …

The (already started?) future

- What is Software Engineering?
  - The same as before (methods, techniques, tools, etc.)
  - BUT
  - With tight connections with “other worlds”
    - Big data
    - Security
    - User experience
    - Blockchain-related developments
    - …

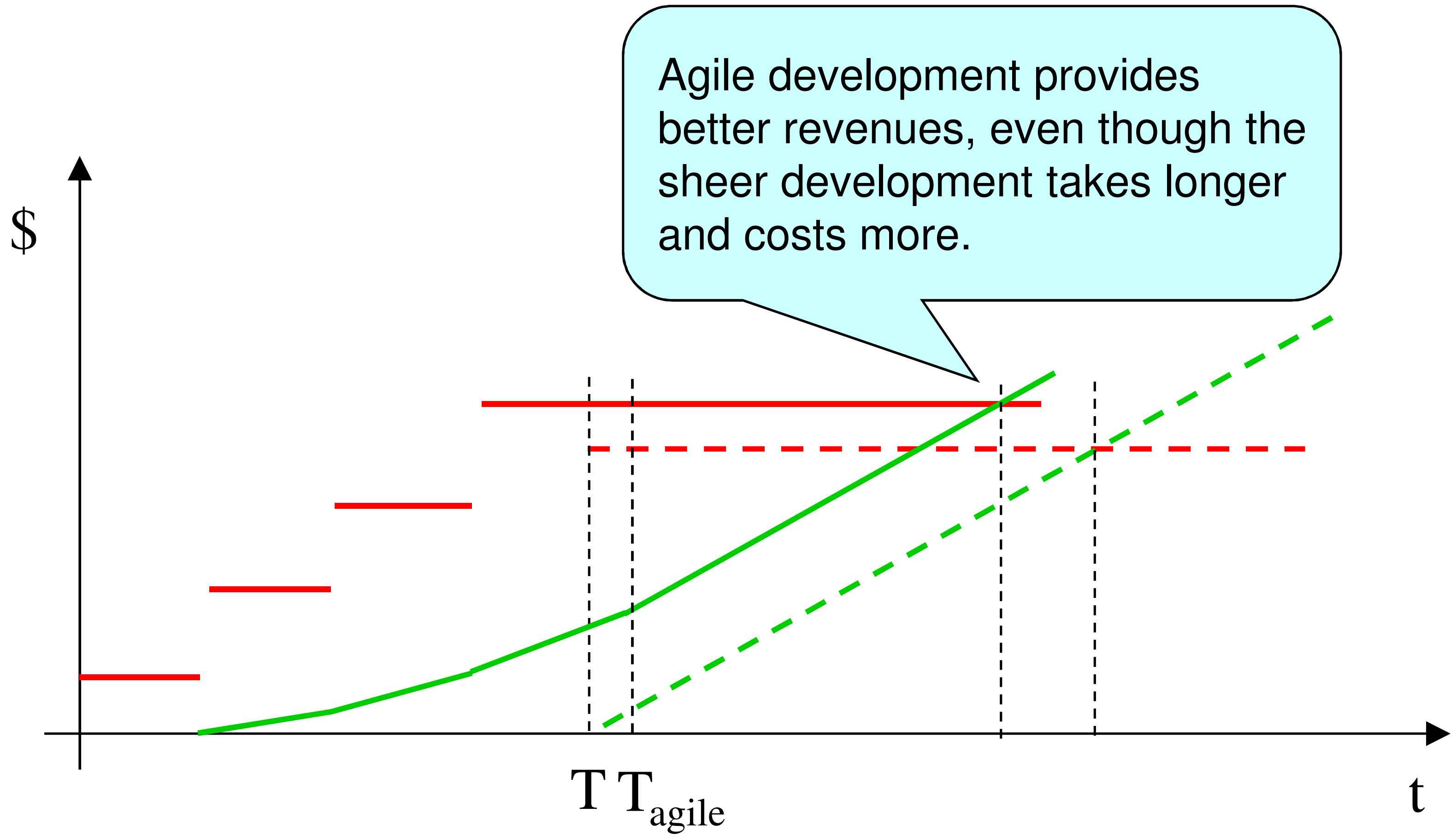
The (already started?) future

- Economics of software development (wrt to economics of business) is going to be quite DIFFERENT with respect to a few years ago.



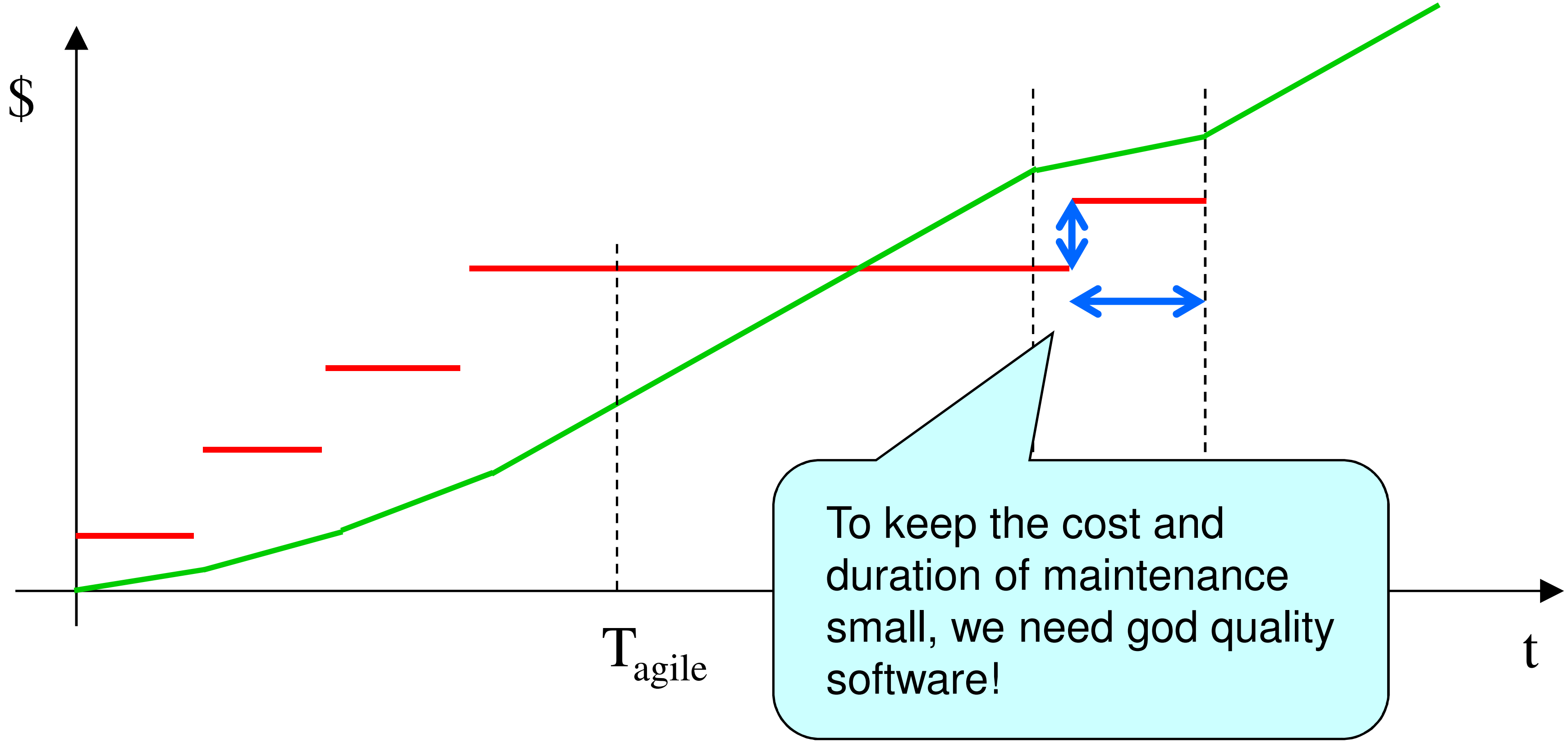
Economics of software development (wrt to economics of business)

- The Agile (incremental) approach



Technical issues

Requirements change. Being able to adapt software wrt to changes in the (business) environment is of outmost importance.



My conclusions

- As software engineers, our mission will not change:
  - We shall work at improving software development
    - To make it as easy, fast, reliable, etc. as possible, to meet the requirements from users and the market
  - But, we shall need to
    - use knowledge, techniques, tools, etc. from other “foreign” domains
    - stay as close to the business needs as possible
    - pursue high quality levels, especially concerning the adaptanbility and extendibility of software

- Software is growingly pervasive and complex
- And so is software engineering

The Twelfth International Conference on Software Engineering Advances

PANEL on SOFTWARE ENGINEERING

Topic: Software Engineering Achievements and Their Evolution

Software Driven Development of Modern Armor Structures

October 11, 2017

Dr.-Ing. Dipl.-Math. Arash Ramezani &
Univ.-Prof. Dr.-Ing. habil. Hendrik Rothe
Helmut-Schmidt-University
University of the Federal Armed Forces Hamburg
Holstenhofweg 85, D-22043 Hamburg

Biography

- Arash Ramezani currently works for the University of the Federal Armed Forces.
- He has studied Applied Mathematics at the University of Bremen and the University of Queensland in Australia and received his Diploma degree in 2010.
- In 2015 he received his doctor's degree in engineering science with his studies on
  - "Numerical Simulation of Terminal Ballistic Processes for the Analysis of Selected Armor Structures and the Optimization of Modern Security Vehicles".
- His research interests include modeling, simulation and visualization of ballistic problems.

Software Driven Development of Modern Armor Structures

- The threat imposed by terrorist attacks is a major hazard for military installations, vehicles and other items
- An important endeavor of international research and development is to avert danger to life and limb
- Ballistic testing is limited due to costs and permissions for experimental results
- This is why numerical simulations are more frequently applied than experimental tests which are thus being replaced gradually

Software Driven Development of Modern Armor Structures

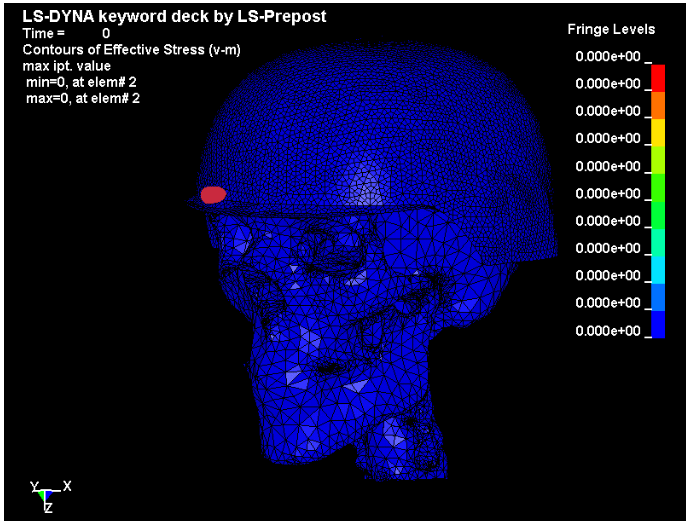
In order to deal with problems involving the release of a large amount of energy over a very short period of time, e.g. explosions and impacts, there are three approaches:

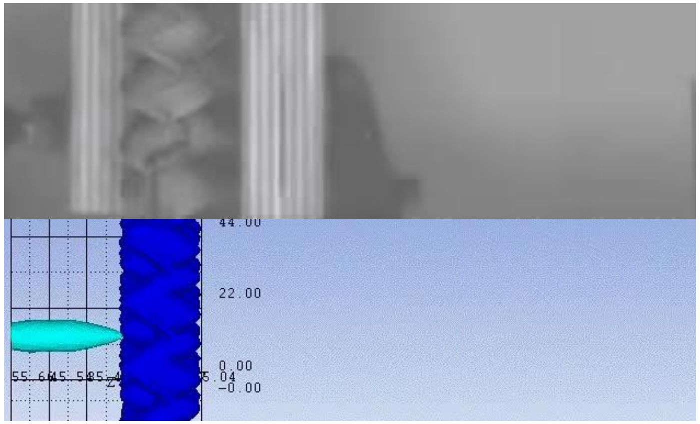
- As the problems are highly non-linear and require information regarding material behavior at ultra-high loading rates which is generally not available, most of the work is **experimental** and thus may cause tremendous expenses
- **Analytical** approaches are possible if the geometries involved remain relatively simple and if the loading can be described through boundary conditions, initial conditions or a combination of the two
- **Numerical** solutions are far more extensive in scope and remove any difficulties associated with geometry

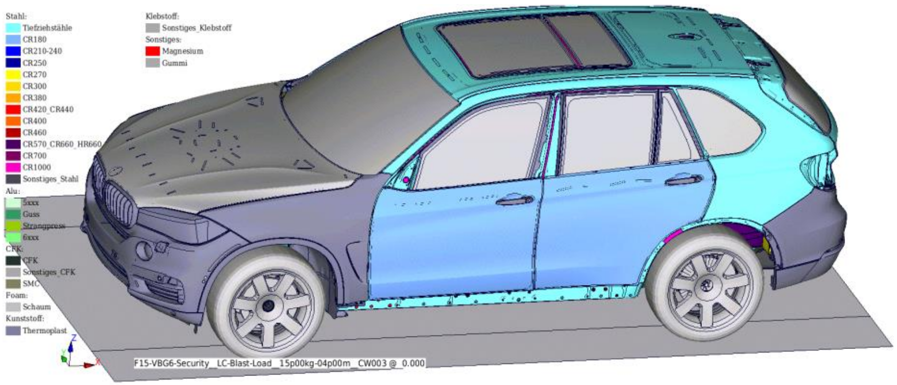
Software Driven Development of Modern Armor Structures

Fields of application:

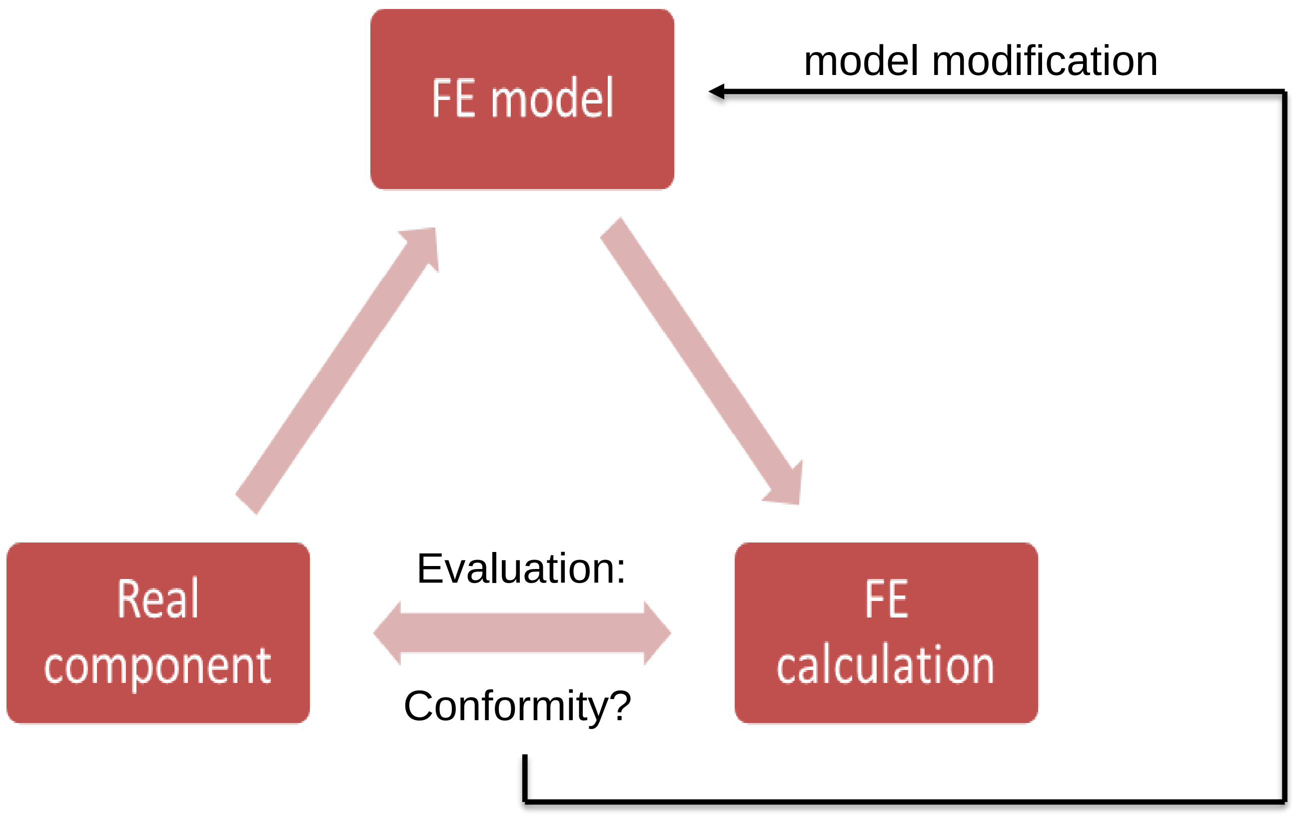
- Simulation of impacts
- Ballistic protection
- Energetic systems
- Wave propagation
- Force of detonation
- Testing of materials







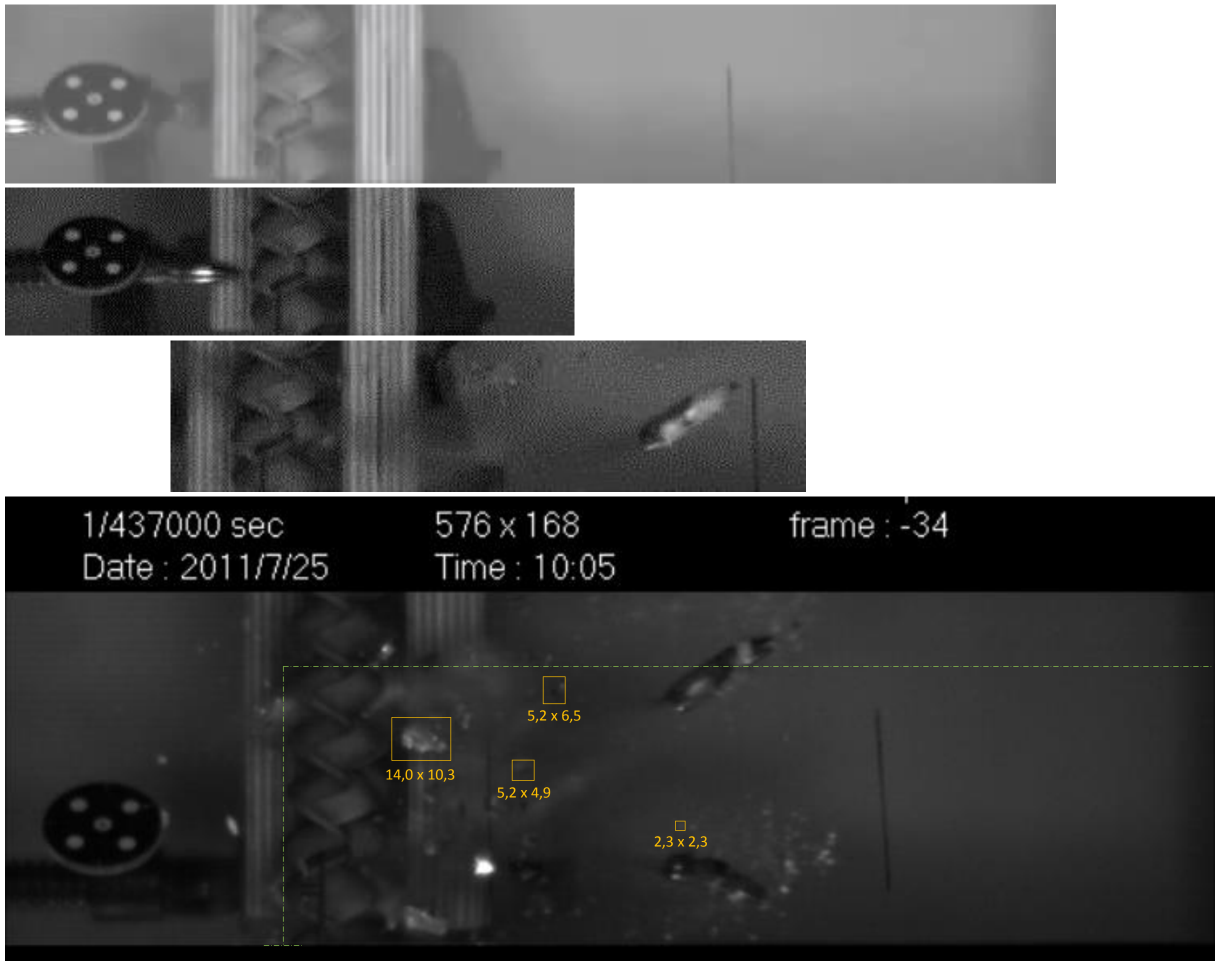
Software Driven Development of Modern Armor Structures



Software Driven Development of Modern Armor Structures

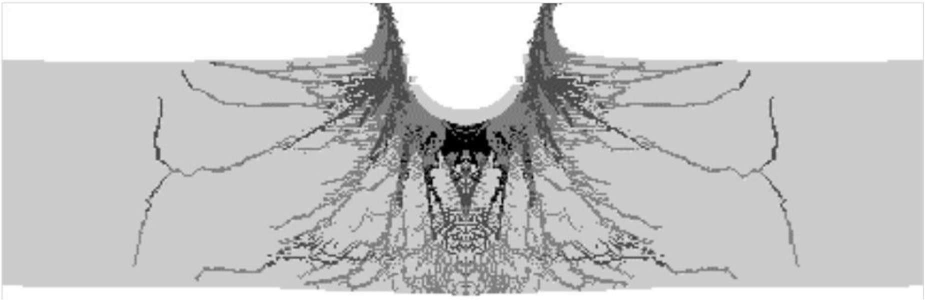
- In order to have a sufficient data base for the simulation, some actual testing must be done prior to the simulation
- Each shot is being recorded with a high-speed-camera and then analyzed in detail
- The fragments of the projectile must be caught and analyzed in the following

Software Driven Development of Modern Armor Structures

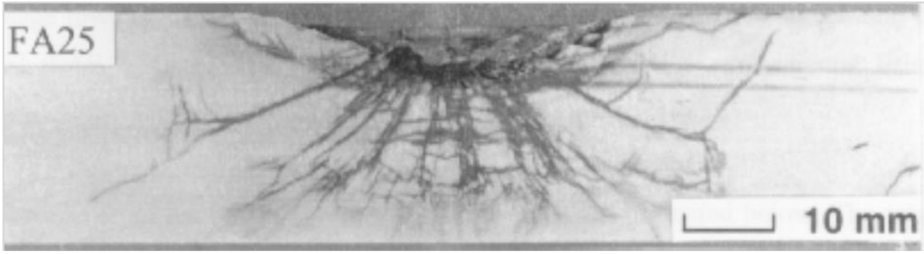


Challenge:

- The materials of the test objects are normally unknown – they have to be created and optimized for the calculation, so that the material behavior in the simulation can be conveyed in an exact manner



Simulation

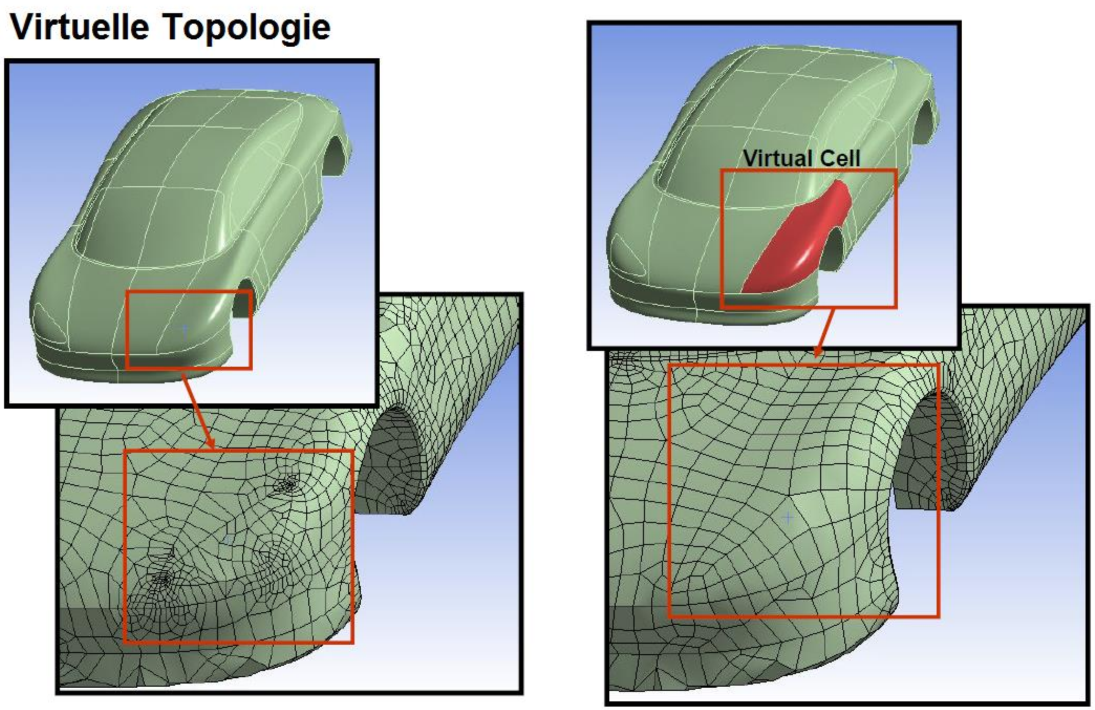


Experiment

Software Driven Development of Modern Armor Structures

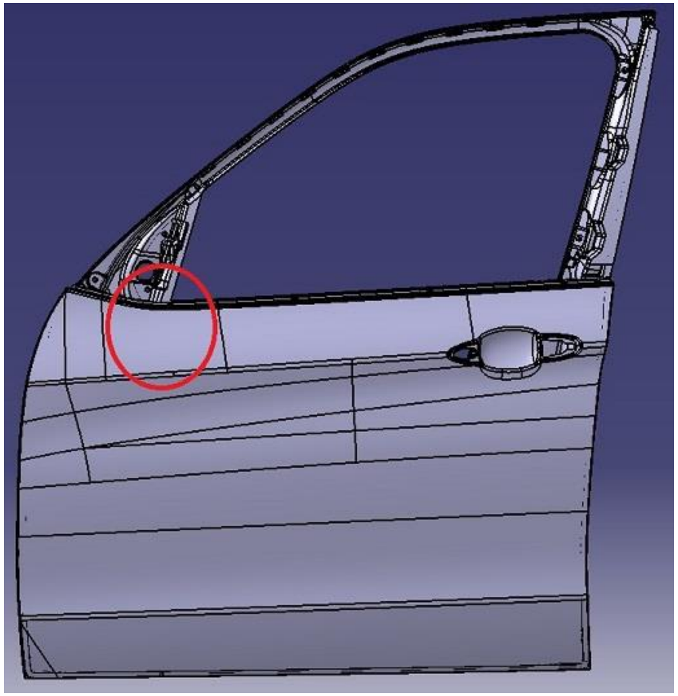
Challenge:

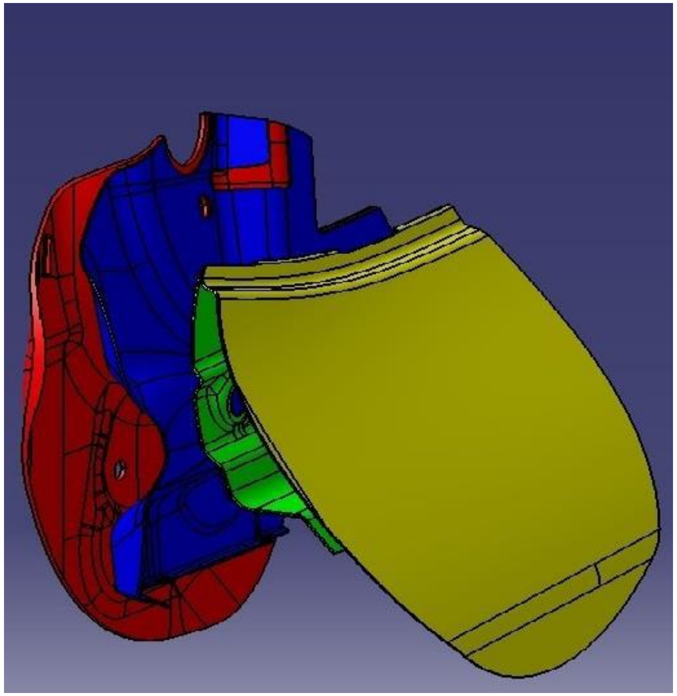
- The mesh used in CAD model has to be as detailed as possible, as particularised as necessary



Challenge:

- Regarding significant places, the models must be refined and the elements must be minimized

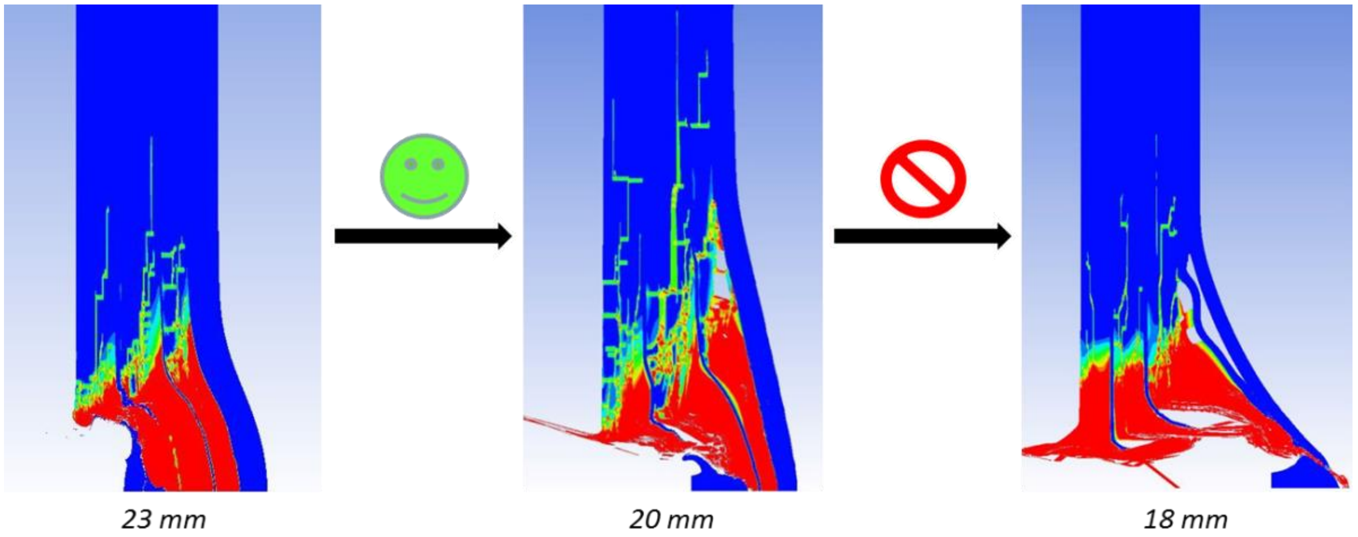




Software Driven Development of Modern Armor Structures

Outlook:

- Based on a detailed CAD model, the actual behavior can be described in a **virtually exact manner**
- New concepts and models can be **optimized** using numerical simulations
- Due to the acquired findings, facilities and vehicles can be **more accurately protected** against terrorist threats



Traditional ballistic testing:

Testing a bulletproof vest in Washington, D.C. (September 1923).