Software Center

Key deliverables from the Software Center Projects

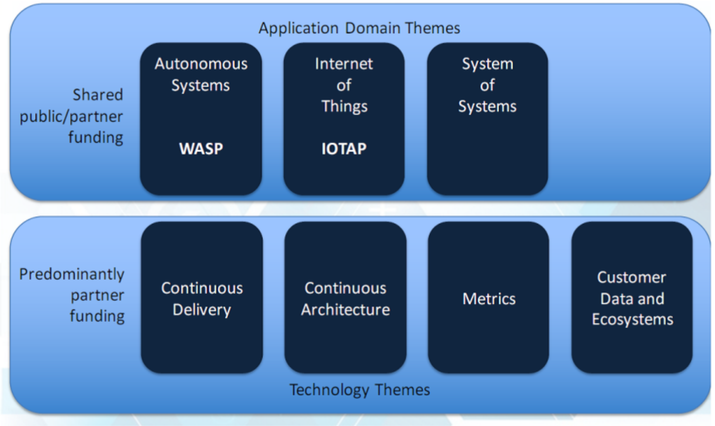
To be used for partner communication as well as attracting new partners and skilled researchers to our front-line research

Revision D 2017-06-20

Software Center Overview

- In Software Center, companies and universities work together to accelerate the adoption of novel approaches to software engineering

- Companies and universities work together in three different application domain themes and four different technology themes



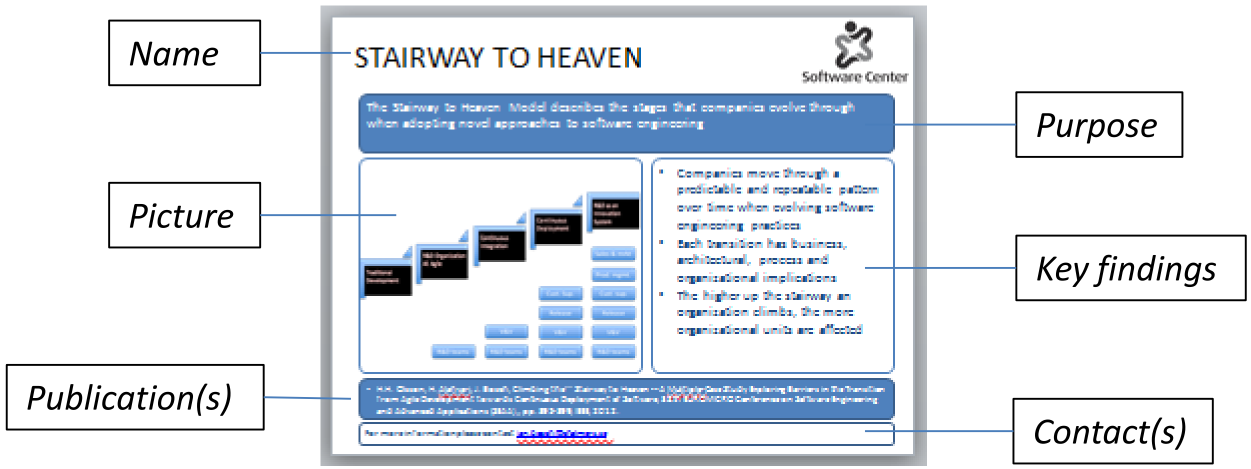
- Software Center is operated in partnership between Chalmers University of Technology, the University of Gothenburg, Malmö University, Linköping University, Mälardalen University and the ten companies Ericsson, Volvo Cars, Volvo AB, Saab Group, Axis Communications, Jeppesen, Grundfos, Tetra Pak, Verisure, Siemens and Bosch.

- More information at http://www.software-center.se/

About this slide-deck

- In this slide-deck we find approaches, models, methods and tools being delivered from research projects within the Software Center

- Each delivery is explained on a high level (according to below template) to be used for partner communication as well as attracting new partners and skilled researchers to our front-line research



- New deliveries are added as research sprints are finalized and new front-line research are developed

Content

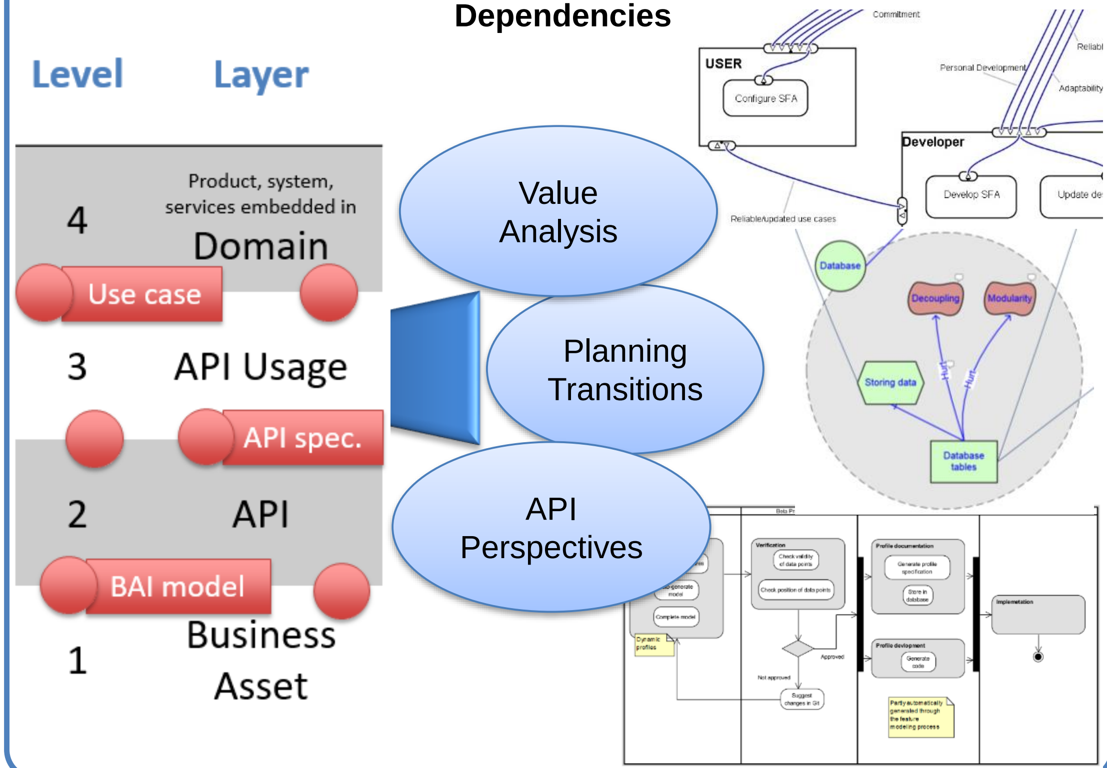
Software Center

APIS – API Strategy Model

Software Center

APIS describes how to investigate and evaluate API strategies for innovation as well as internal and externa value

Strategic Analysis of API Goals, Values, Motivations, Workflows, Dependencies



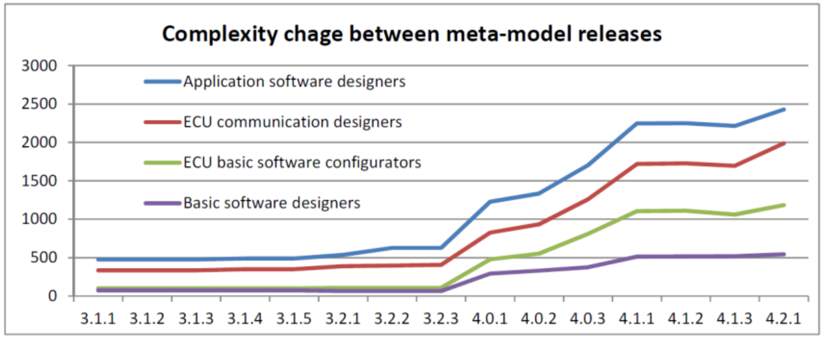
- Keeping up with ever-changing market needs requires a well-defined **API strategy**
- Perspectives:
  - API Layers (shown)
  - BAPO analysis
  - Governance framework
  - Strategic modelling (value, goal, workflow)
  - Transition over time

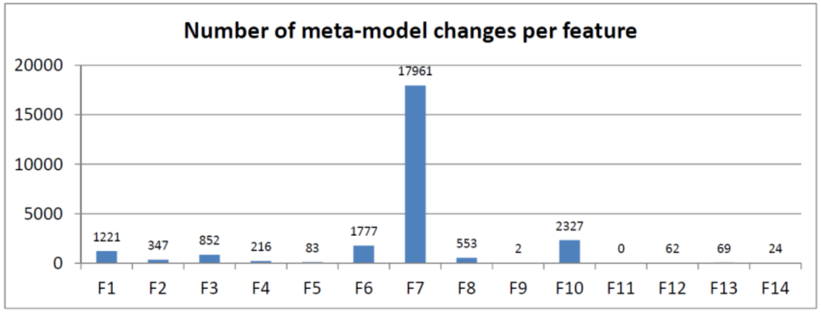
- I. Hammouda, J. Lindman, E. Knauss, J. Horkoff, Emerging Perspectives to API Strategy, IEEE Software, in preparation

For more information please contact Jennifer Horkoff <jenho@chalmers.se>

ARCA

The automated meta-model change analysis method (ARCA) supports the companies in deciding which architectural features to adopt from rapidly changing standards.





- The impact of new standardized meta-model releases on the modeling tools increases with the age of the meta-model
- However, certain standardized architectural features specified in new meta-model releases cause more impact than others
- Using the Pareto-front allows to find optimum (cost-wise) set of features to be implemented

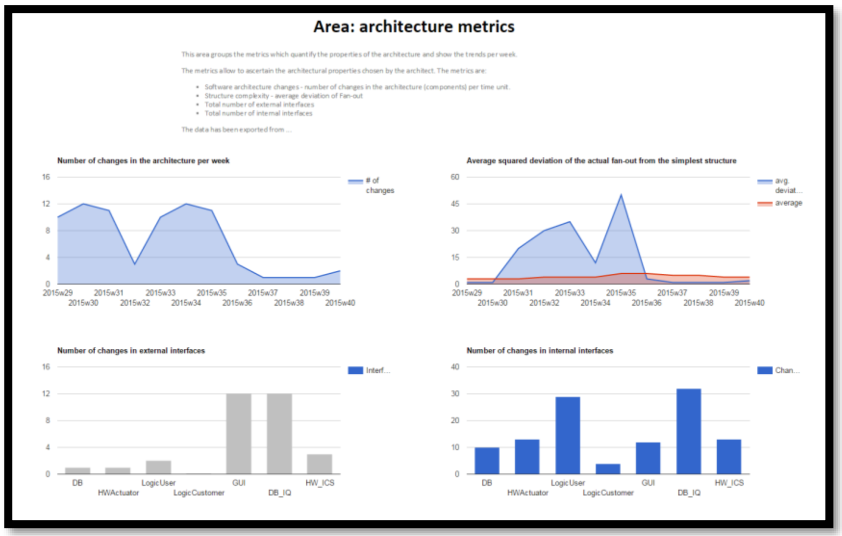
- Durisic, Darko, Miroslaw Staron, and Matthias Tichy. "ARCA: automated analysis of AUTOSAR meta-model changes." In *Proceedings of the Seventh International Workshop on Modeling in Software Engineering*, pp. 30-35. IEEE Press, 2015

For more information please contact [Darko Durisic]

ARCHITECT PORTFOLIO

Software Center

Software architecture can be constantly monitored using a small number of metrics. We chose 9 metrics to provide a good overview of the quality of the architecture.



- Architecture properties area has four metrics
- Architecture design stability has three metrics
- Architecture technical debt has two metrics

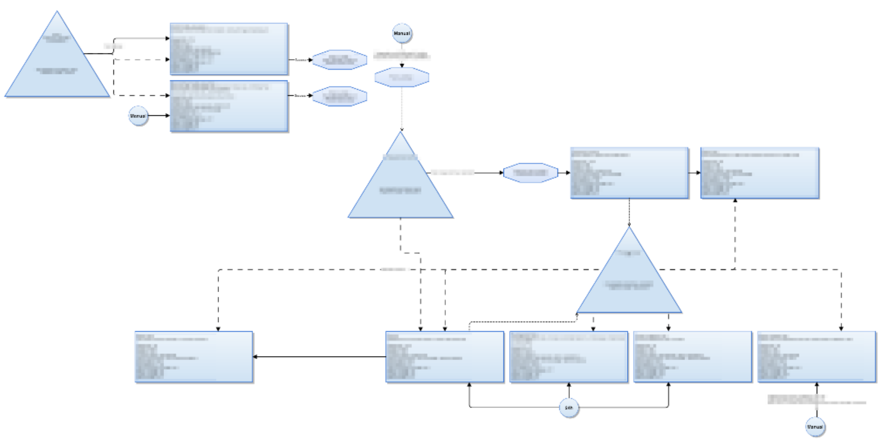
- M. Staron, W. Meding. "A portfolio of internal quality metrics for software architects" In *Proceedings of the 10th International Software Quality Days, Vienna, Austria, 2017*

For more information please contact miroslaw.staron@gu.se.

ASIF

Software Center

The Automated Software Integration Flows affords engineers the ability to model actual or hypothetical continuous integration and delivery systems, improving their ability to plan, analyze and troubleshoot.



- Improved understanding and ability to analyze continuous integration and delivery systems
- Unified view of status, problems and opportunities across disciplines and roles
- Supports troubleshooting and discovery of pain points
- Applicable in tandem with CIViT

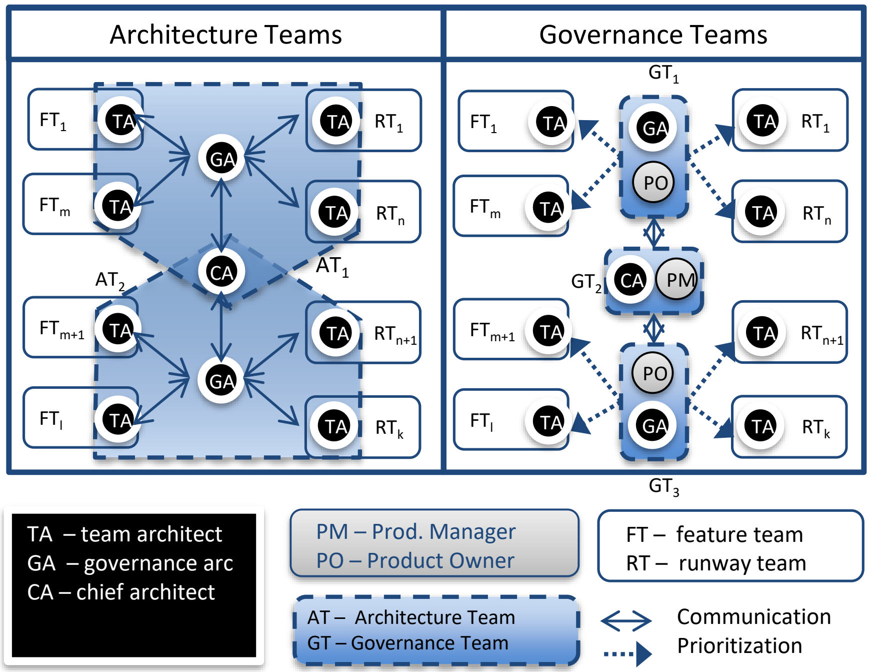
- Ståhl, Bosch. “Modeling continuous integration practice differences in industry software development” JSS 2014
- Ståhl, Bosch. “Automated software integration flows in industry: a multiple-case study” ICSE 2014
- Ståhl, Bosch. “Industry application of continuous integration modeling: a multiple-case study” ICSE 2016

For more information please contact daniel.stahl@ericsson.com and/or jan.bosch@chalmers.se

CAFFEA

Software Center

Architecture Management in Agile needs support: we developed the Continuous Architecture Framework For Embedded and Agile software development (CAFFEA), where the key architecture practices are mapped to necessary roles and virtual teams.



- Improved risk management:
  - Architectural Technical Debt discovered and managed
  - Better balancing of short-term and long-term goals
- Improved architectural decisions:
  - Tracking and follow-up
- Improved communication:
  - Architectural Knowledge spread to the teams
  - Current status of the system
- Improved architectural references

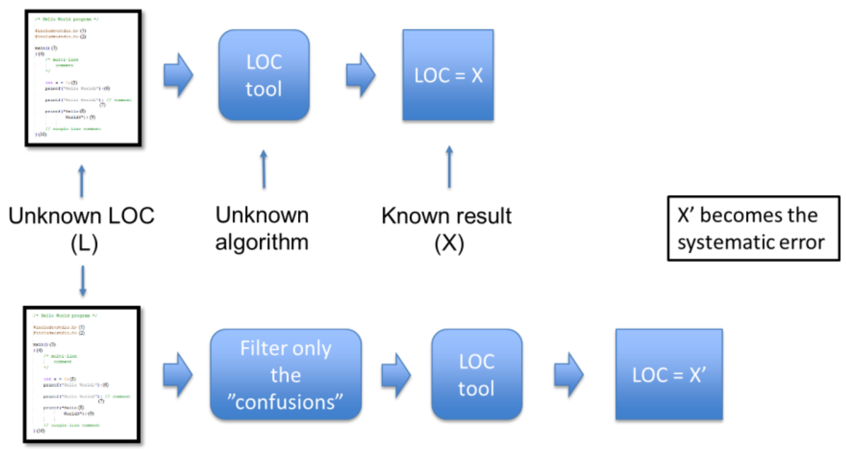
- Martini, Bosch: “A Multiple Case Study of Continuous Architecting in Large Agile Companies: current gaps and the CAFFEA Framework” WICSA 2016
- Martini, Pareto, Bosch: “A multiple case study on the inter-group interaction speed in large, embedded software companies employing agile,” Journal of Software: Evolution and Process, 2016

For more information please contact antonio.martini@chalmers.se

CALIBRATOR

Software Center

Measurement errors can be introduced in different ways in the measurement process. Our calibration model allows to estimate the measurement error of the measurement instruments and therefore reduce uncertainty



- Calibration can be done on a limited number of measured entities
- Measurement errors of LOC measurement can be up to 20%
- Reducing the measurement error in the lowest levels of ISO/IEC 15939 reduce the errors on the higher level ten-fold

- M Staron, D Durisic, R Rana. "Improving Measurement Certainty by Using Calibration to Find Systematic Measurement Error—A Case of Lines-of-Code Measure" In *Software Engineering: Challenges and Solutions, pp. 119-132, 2016*

For more information please contact Darko Durisic.

CCFlex

Software Center

Machine learning can replace programmers of measurement instruments when counting base quantities like LOC

```
                    JRIP rules:
===========

(comment = false) and (full_length >= 11) and (bracket >= 1) => Decision=Count (92.0/
(freq-; >= 1) => Decision=Count (45.0/1.0)
(freq-override >= 1) => Decision=Count (10.0/0.0)
(freq-{ >= 1) => Decision=Count (6.0/0.0)
 => Decision=Ignore (322.0/3.0)

Number of Rules : 5
```

```
package pl.put.poznan.ccflex.classifiers;

import pl.put.poznan.ccflex.resources.Line;

public interface LineClassifier {

        public Line classify(Line line) throws Exception;
```

- ML uses decision trees to learn how to count
- Learning-by-example makes measuring available to any role in the organization
- CCFlex is 96% correct compared to manual counting

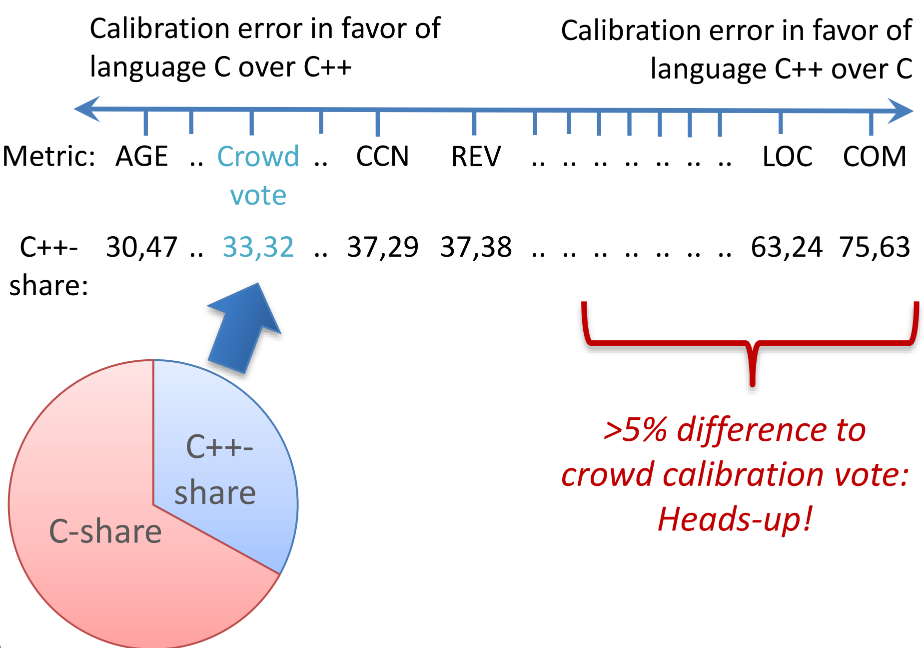
M. Ochodek, M. Staron, D. Bargowski, W. Meding, R. Hebig, Using Machine Learning to Design a Flexible LOC Counter, *Workshop on Machine Learning in Software Quality (MALTESQUE), co-located with SANER 2017, Klagenfurt, Austria*

For more information please contact miroslaw.staron@gu.se.

CC-RAY

Software Center

Productivity measures, cost prediction, and quality assessments rely on size measures. However, most of these metrics cannot be compared across programming languages. The CC-Ray Model allows companies to identify and exclude inadequate metrics.



- Existing metrics cannot be used to compare systems written in different languages
- Calibration errors are different between companies and open source
- At the current state of the art a crowd sourced size comparison is the best measure one can get

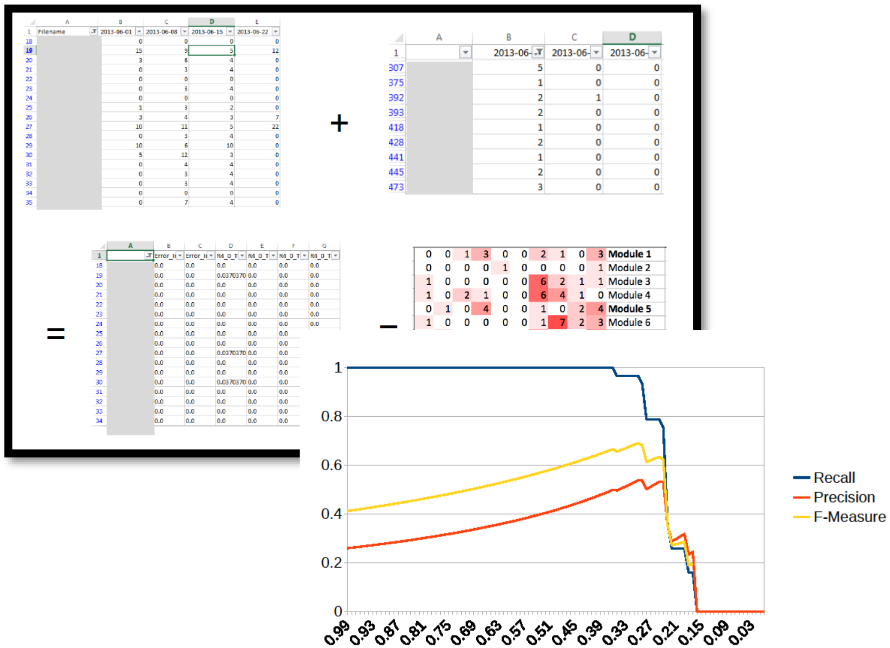
- Hebig, Regina, Jesper Derehag, and Michel RV Chaudron. "Identifying Metrics' Biases When Measuring or Approximating Size in Heterogeneous Languages." *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2015.

For more information please contact regina.hebig@csu.gu.se

CCTS

Software Center

The Code-Churn Test Selection model identifies the most optimal test suite based on the changes in the source code.



- Reduction of test suite by 73% without any loss of effectiveness
- Can speed up continuous integration and reduce cycle times
- Can be applied at all test levels

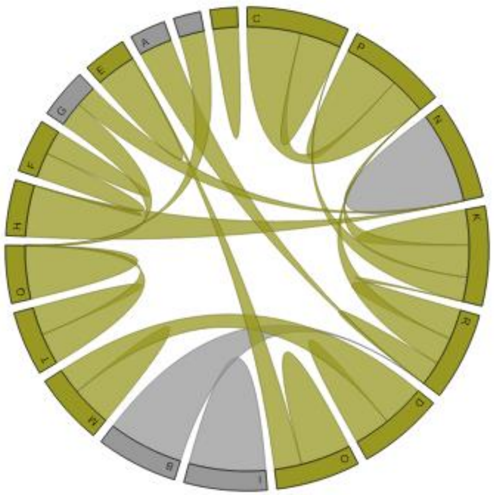
- E. Knauss, M. Staron, W. Meding, O. Söder, A. Nilsson, M. Castell, “Supporting Continuous Integration by Code-Churn Based Test Selection”, Proceedings of the 2nd International Workshop on Rapid and Continuous Software Engineering (RCoSE), ICSE 2015, Italy

For more information please contact eric.knauss@cse.gu.se, agneta.nilsson@cse.gu.se and/or miroslaw.staron@cse.gu.se

CHANGE WAVE

Software Center

The Change-wave model quantifies the changes in source code and identifies implicit dependencies to provide architects and test leaders with information on what to test and how.



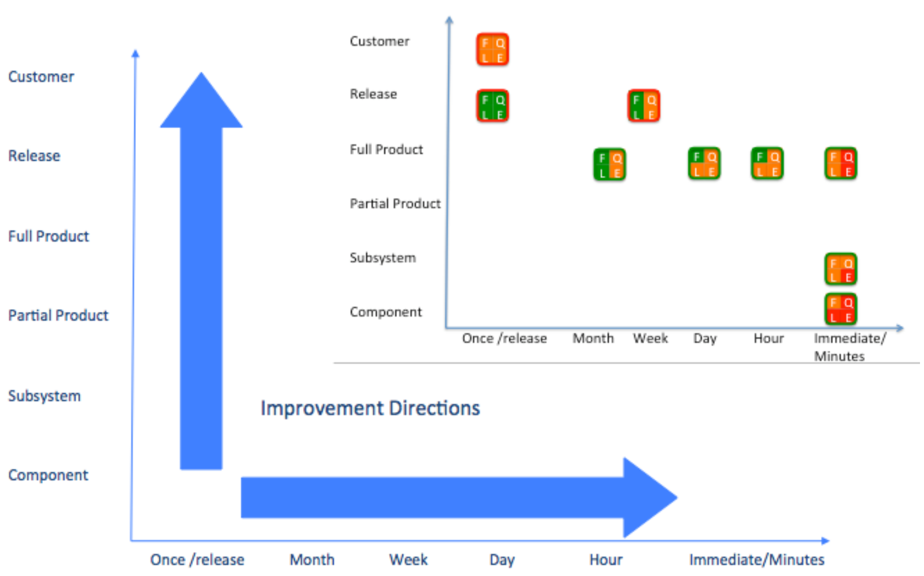
- Source components that change together should be tested together
- Replacing advanced static code analyses with simple statistics gives 80% of the same picture
- Having a fast feedback on the design saves test and maintenance effort

- Staron, Miroslaw; Meding, Wilhelm; Höglund, Christoffer; Ericsson, Peter; Nilsson, Jimmy; Hansson, Jörgen: Identifying Implicit Architectural Dependencies using Measures of Source Code Change Waves, SEAA, Software Engineering and Advanced Applications, Conference Proceedings, 2013

For more information please contact miroslaw.staron@cse.gu.se

CIVIT

The CIVIT model is a test process improvement technique with the purpose to visualizing the end-to-end testing activities involved (from component to product level) to create a shared understanding of the current situation and support the identification of improvement areas.



- Often a lack of an adequate overview
- Tend to lead to double work, slow feedback loops, issues found too late, disconnected organizations, unpredictable release schedules
- It enables a solid understanding of the end-to-end testing activities
- Particularly useful as a basis for discussion, to identify problems and to reason about suitable measures

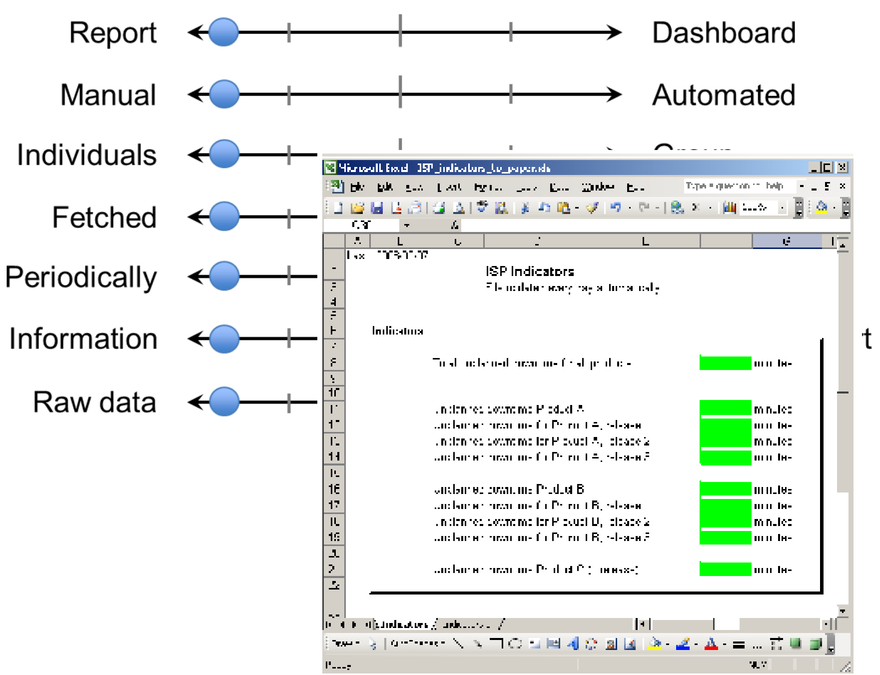
- Nilsson, A., Bosch, J. and Berger, C. (2014) 'Visualizing testing activities to support continuous integration: A multiple case study'. In proceedings of Agile Software Development, XP, Rome, Italy, 26-30 May, 2014. Springer Volume 179, pp. 171–186

For more information please contact agneta.nilsson@gu.se

DASHBOARD SELECTION

Software Center

The dashboard selection model allows to quickly identify which kind of dashboard is needed by the company and which technology should be used to implement it.



- Business analytics tools are good for individuals whereas Dashing-like tools are good for landscapes
- Custom-build dashboard tools are the most cost-inefficient solutions in the long-run
- Modularization of the data flow given the largest short- and long-term benefits

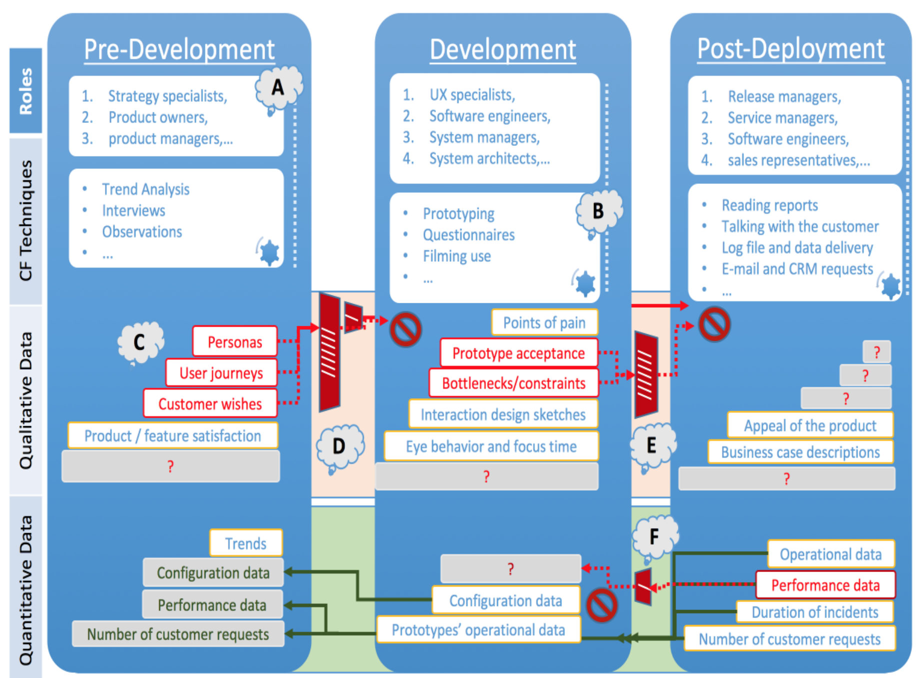
- M Staron, K Niesel, and W Meding, 'Selecting the Right Visualization of Indicators and Measures–Dashboard Selection Model', in International Conference on Software Measurement (Mensura), 2015

For more information please contact miroslaw.staron@cse.gu.se

DATA SHARING MODEL

Software Center

The Data Sharing model identifies (1) what customer data is collected, (2) by whom it is collected and (3) the development phases in which it is used. The model helps companies identify critical hand-overs where data gets lost and the implications of this.



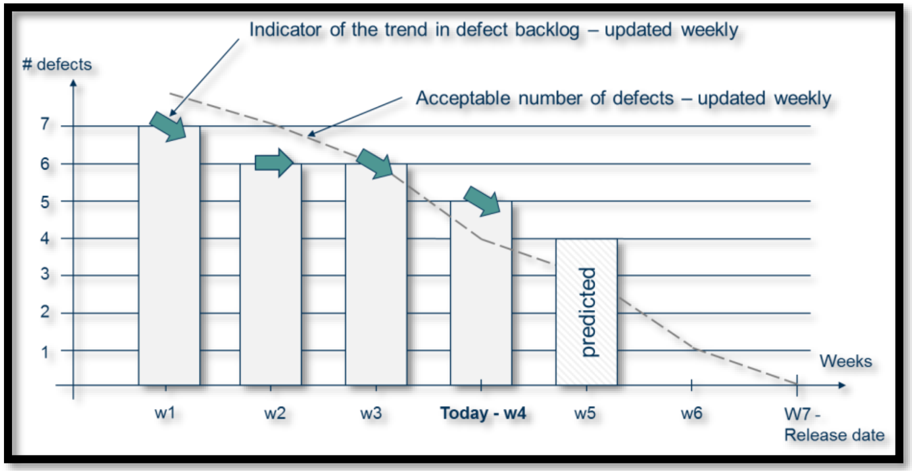
- Companies benefit from a very limited part of all the data they collect from customers.
- The model identifies (1) fragmented collection,(2) filtering of data and (3) overrepresentation of quantitative and "measurable" aspects as the main challenges associated with sharing of customer data.
- The model shows how lack of sharing of data leads to an inaccurate understanding of what constitutes customer value.

- Fabijan, A., Olsson, H.H., and Bosch, J. (2016). The Lack of Sharing of Customer Data in Large Software Organizations: Challenges and Implications. *In Proceedings of XP 2016, May 24-27th, Edinburgh, Scotland.*

For more information please contact aleksander.fabijan@mah.se, helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

DEFECT FORECAST

Forecasting defect inflow on a weekly basis can be difficult, but forecasting the BACKLOG is much easier. This method supports development programs in resource allocation.



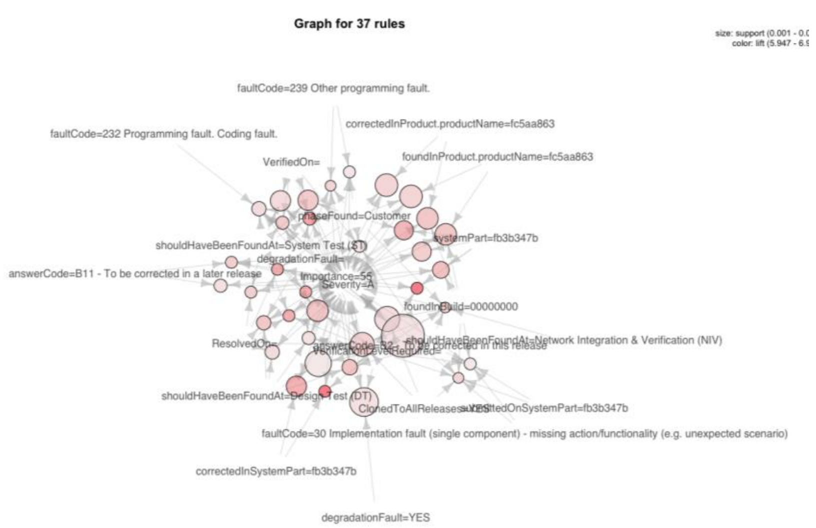
- Weekly defect backlog can be predicted with up to 92% accuracy
- Predicting with up to 3 weeks in advance allows to make decisions in time
- Combining with the long-term predictions allows the best prediction horizon

- Staron, M., Meding, W. and Söderqvist, B., 2010. A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation. *Information and Software Technology*, *52*(10), pp.1069-1079

For more information please contact miroslaw.staron@gu.se

DFlex

Machine learning can replace software quality managers in assigning severity to a defect when it is applied; machine learning can also predict the impact of a defect on the product.



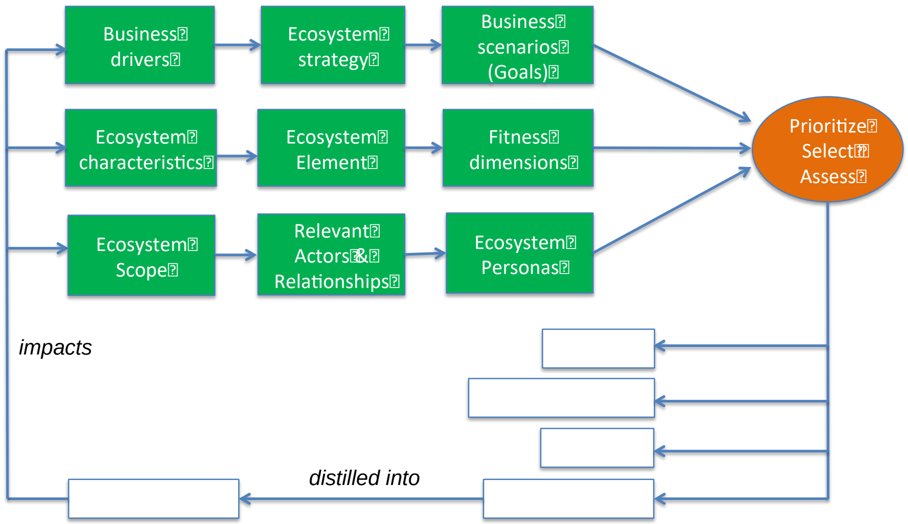
- ML uses decision trees to classify the defects
- Learning-by-example makes measuring available to any role in the organization
- DFlex is 88% correct compared to manual classification

In preparation

For more information please contact miroslaw.staron@gu.se.

EAM

EAM (ecosystemability assessment method) supports organizations in assessing the ecosystemability of their software systems. We define ecosystemability as the degree to which a software system and its development environment support the vision of ecosystem.



Successful use cases:

- API design for the ecosystem
- Collaboration in Shared Tooling Infrastructure Ecosystem

The conceptual flow of the EAM includes

1. the analysis of business goals, strategy, and scenarios,
2. the structure of the ecosystem and its main elements (such as platforms)
3. the dynamics between the various ecosystem actors and their personas

On service provider side, we applied this conceptual flow in two workshops and two surveys to analyze the suit- ability of API designs to support an ecosystem. On service consumer side, we analyzed the feedback cycle time for requirements and identified (shared) tool-support for integration and verification as an important focus for future analysis

- Imed Hammouda, Eric Knauss, and Leonardo Costantini. Continuous API-Design for Software Ecosystems. In Proceedings of 2nd International Workshop on Rapid and Continuous Software Engeering (RCoSE '15 @ ICSE), Florenz, Italy, 2015
- Eric Knauss and Imed Hammouda: EAM: Ecosystemability Assessment Method. In: Proc. of 22nd Int. Requirements Engineering Conf. (RE '14), pg. 319-320, Karslkrona, Sweden, 2014

For more information please contact imed.hammouda@cse.gu.se and/or eric.knauss@cse.gu.se

ECE

The Evolution of Continuous Experimentation

Software Center

Companies struggle to become data-driven at scale. The model below addresses this challenge by providing guidance on how to develop and evolve from the first controlled experiment towards continuous controlled experimentation at scale.

	Category/ Phase	Crawl	Walk	Run	Fly
Technical Evolution	Technical focus of product dev. Activities	**(1) Logging of signals** **(2) Work on data quality issues** **(3) Manual analysis of experiments** Transitioning from the debugging logs to a format that can be used for data-driven development.	**(1) Setting-up a reliable pipeline** **(2) Creation of simple metrics** Combining signals with analysis units. Four types of metrics are created: debug metrics (largest group), success metrics, guardrail metrics and data quality metrics.	**(1) Learning experiments** **(2) Comprehensive metrics** Creation of comprehensive set of metrics using the knowledge from the learning experiments.	**(1) Standardized process for metric design and evaluation, and OEC improvement**
	Experimentation platform complexity	**No experimentation platform** An initial experiment can be coded manually (ad-hoc).	**Platform is required** 3rd party platform can be used or internally developed. The following two features are required: • **Power Analysis** • **Pre-Experiment A/A testing**	**New platform features** The experimentation platform should be extended with the following features: • **Alerting** • **Control of carry-over effect** • **Experiment iteration support**	**Advanced platform features** The following features are needed: • **Interaction control and detection** • **Near real-time detection and automatic shutdown of harmful experiments** • **Institutional memory**
	Experimentation pervasiveness	**Generating management support** Experimenting with e.g. design options for which it's not a priori clear which one is better. To generate management support to move to the next stage.	**Experiment on individual feature level** Broadening the types of experiments run on a limited set of features (design to performance, from performance to infrastructure experiments)	**Expanding to (1) more features and (2) other products** Experiment on most new features and most products.	**Experiment with every minor change to portfolio** Experiment with any change on all products in the portfolio. Even to e.g. small bug fixes on feature level.
Organizational Evolution	Engineering team self-sufficiency	**Limited understanding** External Data Scientist knowledge is needed in order to set-up, execute and analyse a controlled experiment.	**Creation and set-up of experiments** Creating the experiment (instrumentation, A/A testing, assigning traffic) is managed by the local Experiment Owners. Data scientists responsible for the platform supervise Experiment Owners and correct errors.	**Creation and execution of experiments** Includes monitoring for bad experiments, making ramp-up and shut-down decisions, designing and deploying experiment-specific metrics.	**Creation, execution and analyses of experiments** Scorecards showing the experiment results are intuitive for interpretation and conclusion making.
	Experimentation team organization	**Standalone** Fully centralized data science team. In product teams, however, no or very little data science skills. The standalone team needs to train the local product teams on experimentation. We introduce the role of Experiment Owner (EO).	**Embedded** Data science team that implemented the platform supports different product teams and their Experiment Owners. Product teams do not have their own data scientists that would analyse experiments independently.	**Partnership** Product teams hire their own data scientists that create a strong unity with business. Learning between the teams is limited to their communication.	**Partnership** Small data science teams in each of the product teams. Learnings from experiments are shared automatically across organization via the institutional memory features.
Business Evolution	Overall Evaluation Criteria (OEC)	OEC is **defined** for the first set of experiments with a few key signals that will help ground expectations and evaluation of the experiment results.	**OEC evolves** from a few key signals to a structured set of metrics consisting of Success, Guardrail and Data Quality metrics. Debug metrics are not a part of OEC.	OEC is **tailored** with the findings from the learning experiments. Single metric as a weighted combination of others is desired.	OEC is **stable**, only periodic changes allowed (e.g. 1 per year). It is also used for setting the performance goals for teams within the organization.

The model helps software companies to develop and evolve continuous controlled experimentation.

- We identify four stages of continuous controlled experimentation: **Crawl**, **Walk**, **Run** and **Fly**.
- In each of the four stages, we describe the key activities to evolve and scale data-driven practices (e.g. new platform features, organizational arrangements, and evaluation criteria development).

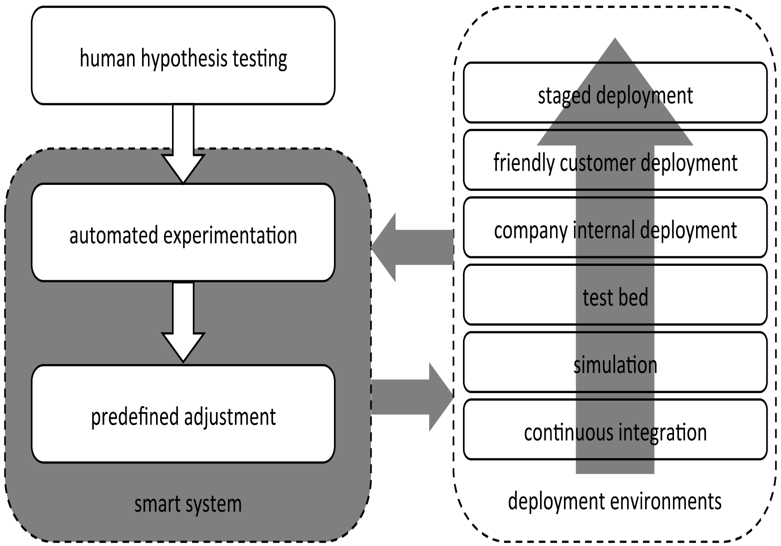
- A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The Evolution of Continuous Experimentation in Software Product Development," *to appear in: Proceedings of the 39th International Conference on Software Engineering - ICSE '17*, 2017

For more information please contact aleksander.fabijan@mah.se

EDAX

Software Center

The EDAX model is a development model for autonomous systems as an integrated effort between R&D teams and the system itself. R&D teams build part of the functionality and the system experiments and adjusts its behaviors autonomously.



- The systems that we build today and in the future exhibit levels of autonomy that put new demands on SE practices
- The EDAX model presents a method for systematically building autonomous systems that employ modern SE technology
- The EDAX model defines three loops of data-driven adjustment of system behaviors

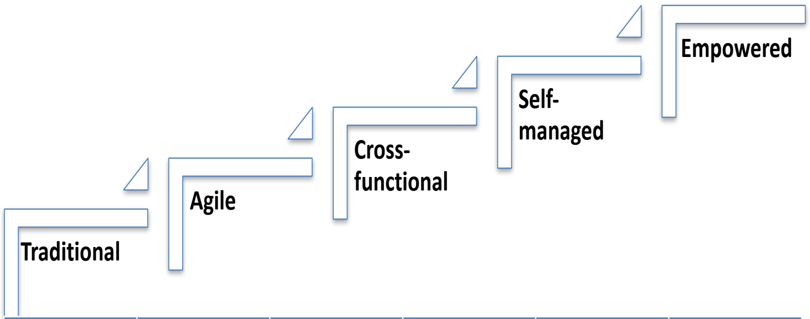
- Bosch, J., and Olsson, H.H. (2016). DataDriven Continuous Evolution of Smart Systems. In Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), May 16-17, 2016, Austin, Texas

For more information please contact jan.bosch@chalmers.se and/or helena.holmstrom.olsson@mah.se

EMPOWERED ORGANIZATIONS

Software Center

The 'Empowered Organizations' model details five steps that organizations take when transitioning from hierarchical structures to empowered ways-of-working characterized by decentralized decision-making and autonomous teams.



	Traditional	Agile	Cross-functional	Self-managed	Empowered
Culture	Hierarchical	Hierarchical	Hierarchical	Hierarchical	Empowered
General Mgmt.	Hierarchical	Hierarchical	Hierarchical	Empowered	Empowered
Inter-team (PdM/R&D)	Hierarchical	Hierarchical	Empowered	Empowered	Empowered
Local (R&D)	Hierarchical	Empowered	Empowered	Empowered	Empowered

- Traditional hierarchical organizations have challenges meeting rapidly changing market and customer needs and need guidance for how to organize to address these challenges
- The 'Empowered Organizations' model provides guidance for how to transition towards an organization characterized by empowered and autonomous teams
- Companies adopting this paradigm shift early will improve competitiveness by increasing responsiveness to customers and effectiveness of R&D

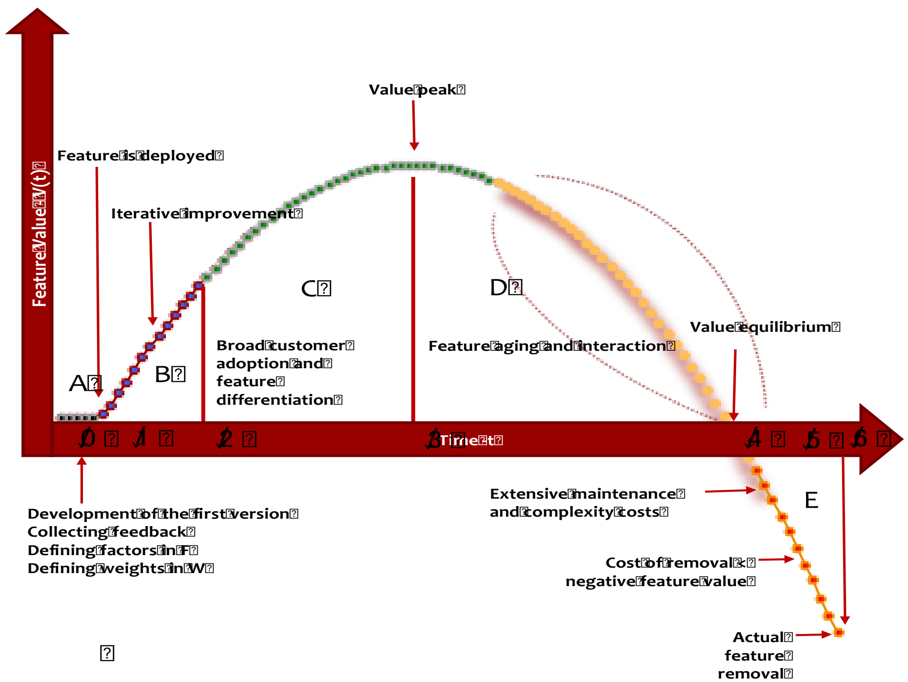
- Olsson, H.H., and Bosch, J. (2016). No More Bosses? A multi-case study on the emerging use of non-hierarchical principles in large-scale software development. *In Proceedings of the 17th International Conference on Product-Focused Software Process Improvement (PROFES), November 22nd-24th, Trondheim, Norway.*

For more information please contact helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

FEATURE LIFECYCLE MODEL

Software Center

The Feature Lifecycle model identifies the phases that a feature moves through during its lifetime, and how the value of a feature changes over time. The model helps companies continuously track the value of a feature throughout the feature lifecycle.



- Companies find it difficult to track feature value over time and identify what actions to take when a feature no longer adds value.
- The model details five phases that a feature move through during its lifetime.
- The model helps companies determine (1) when to add a new feature to a product, (2) how to track the value of a feature over time, and (3) how to identify when a feature is obsolete and should be removed from the product.

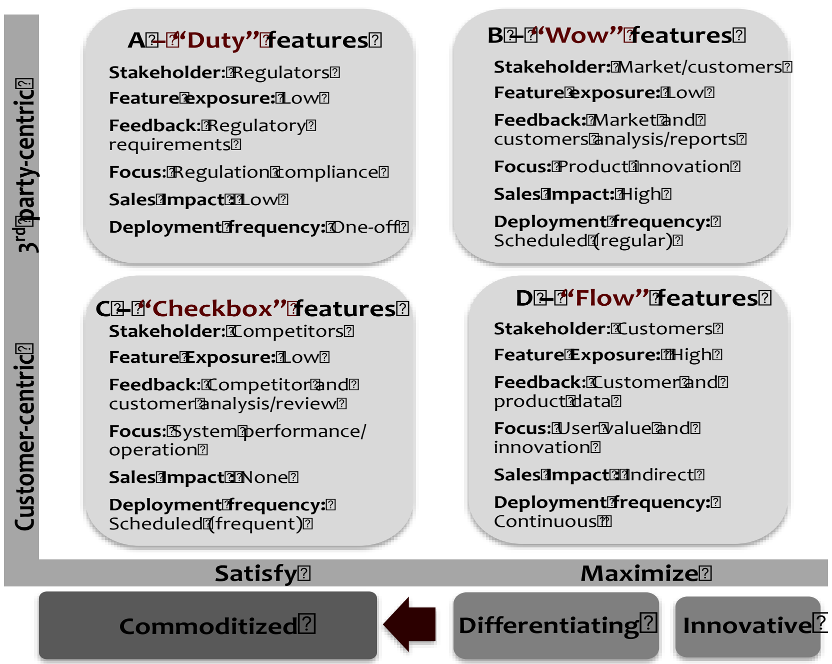
- Fabijan, A., Olsson, H.H., and Bosch, J. (2016). Time to Say 'Good Bye': Feature Lifecycl. *In Proceedings of the 42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), August 31 – September 2, Limassol, Cyprus.*

For more information please contact aleksander.fabijan@mah.se, helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

FEATURE TYPES MODEL

Software Center

The Feature Types model identifies four different types of features that companies develop. The model helps companies (1) prioritize among different features to improve R&D allocation and (2) shift R&D efforts to features that bring most value to customers.



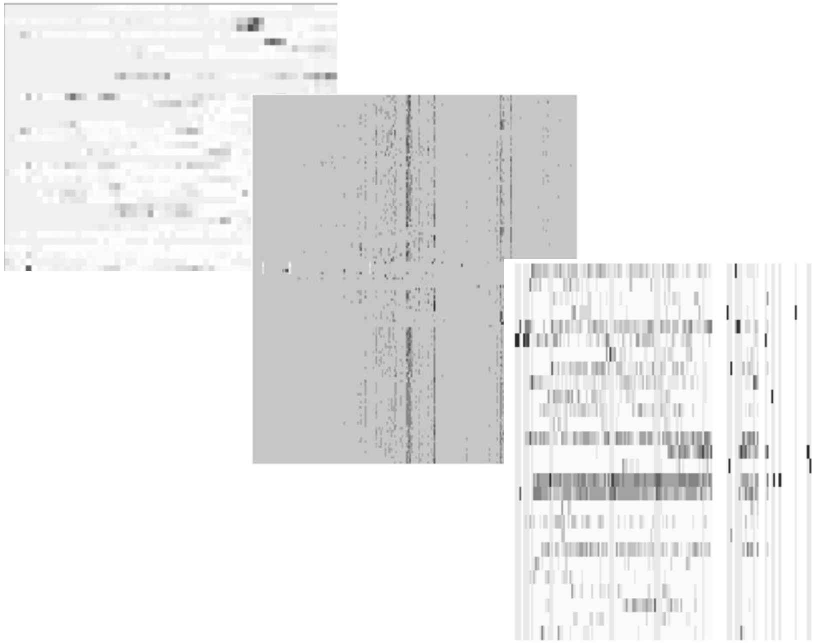
- Companies struggle with distinguishing between different types of features. As a result, all features receive equal R&D efforts and investments.
- The model helps companies identify commodity features, differentiating features and innovative features to avoid heavy investments in commodity and shift resources to differentiating functionality.
- The model helps companies improve resource allocation by identifying the focus, impact and value of features.

- Fabijan, A., Olsson, H.H., and Bosch, J. (submitted). Commodity Eats Innovation for Breakfast: A Model for Differentiating Feature Realization. *Submitted to the 17th International Conference on Product-focused Software Process Improvement (PROFES), November 22-24th, Trondheim, Norway.*

For more information please contact aleksander.fabijan@mah.se, helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

HEAT MAP

The heatmap model quantifies and visualizes large quantities of source code change to show the stability of a software development product.



- Long vertical lines indicate release focus
- Agile software development often results in less stable code base
- Platforms' stability is significantly different than application stability

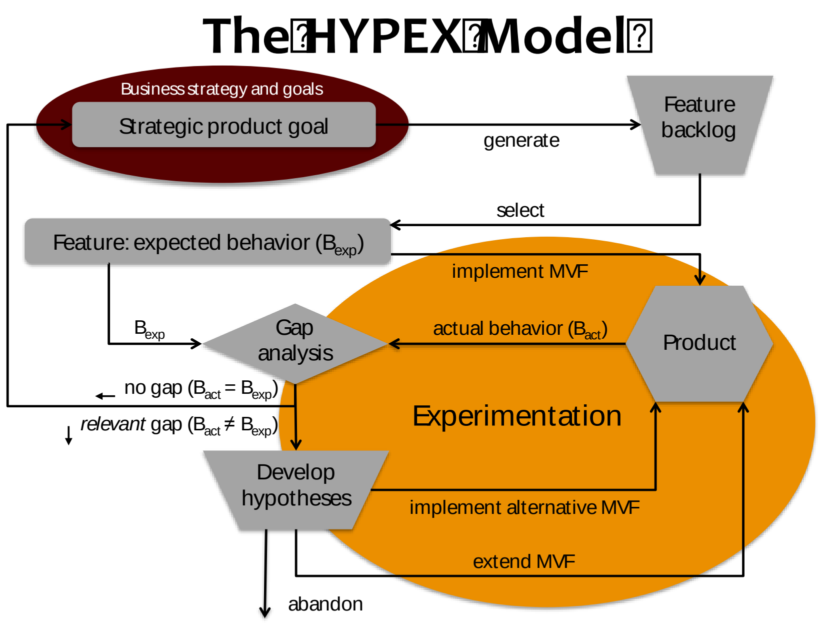
- Staron, Miroslaw, Jorgen Hansson, Robert Feldt, Wilhelm Meding, Aron Henriksson, Sven Nilsson, and Christoffer Hoglund. "Measuring and visualizing code stability--a case study at three companies." In *Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on*, pp. 191-200. IEEE, 2013
- Feldt, Robert, Miroslaw Staron, Erika Hult, and Thomas Liljegren. "Supporting software decision meetings: Heatmaps for visualising test and code measurements." In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pp. 62-69. IEEE, 2013

For more information please contact miroslaw.staron@cse.gu.se

HYPEX

Software Center

The HYPEX model helps companies run feature experiments during development to continuously validate customer value. The model helps companies shorten the feedback loop to customers and adopt data-driven development practices.



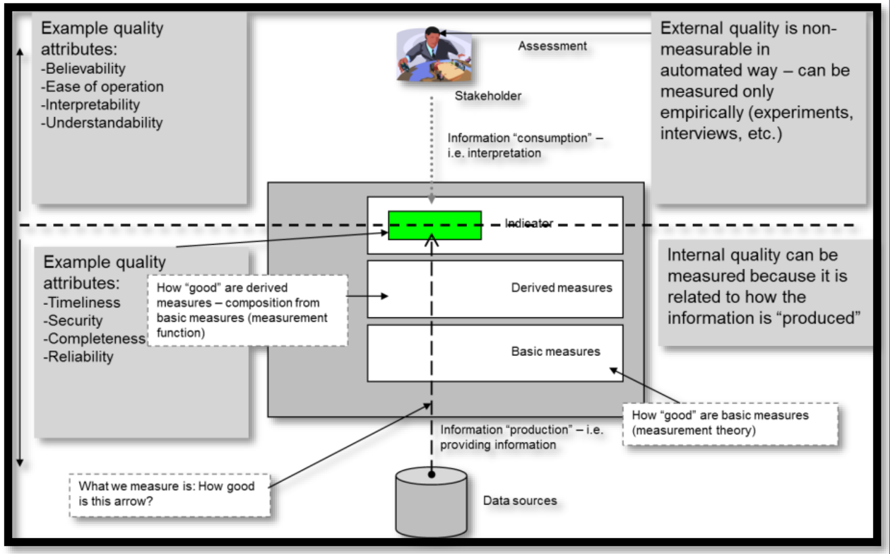
- By continuously validating customer value, the HYPEX model helps companies in the feature ***road-mapping and prioritization*** process
- By continuous experimentation and collection of customer data, the HYPEX model helps companies transition from opinions-based towards ***data-driven development***
- By enabling access to accurate customer data, the HYPEX model closes the ***'open loop'*** between PdM and customers

- Olsson H.H., and Bosch J. (2014). From Opinions to Data-Driven Software R&D: A Multi-Case Study On How To Close The 'Open Loop' Problem. In Proceedings of EUROMICRO, Software Engineering and Advanced Applications (SEAA), August 27-29, Verona, Italy

For more information please contact helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

INFORMATION QUALITY

Monitoring information quality of measurement systems assures that the decisions are taken based on the right data at the right moment



- ISO/IEC 15939 compatible IQ model
- Prevents measurement errors from propagating in the organization
- Visualizes the problems to facilitate fast troubleshooting

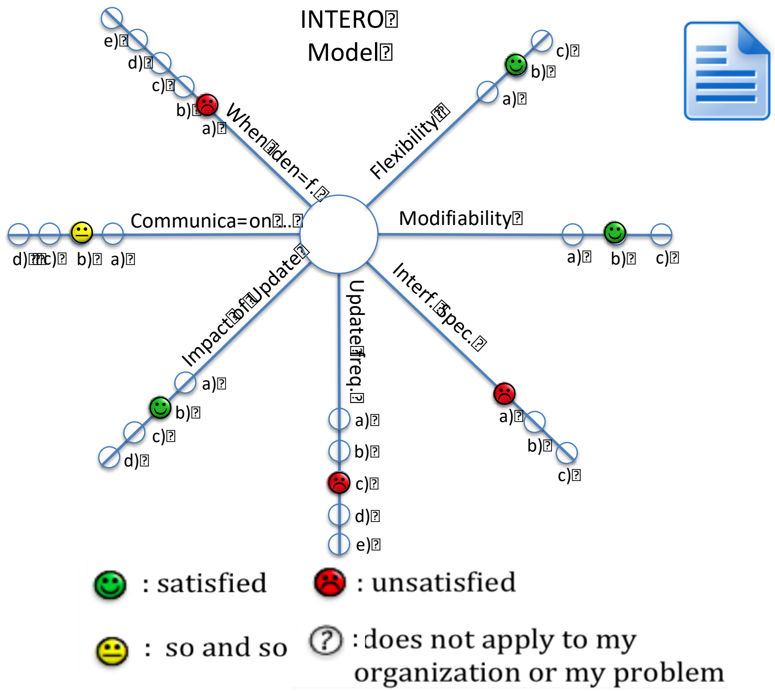
- Staron, M. and Meding, W., 2009. Ensuring reliability of information provided by measurement systems. In *Software Process and Product Measurement*(pp. 1-16). Springer Berlin Heidelberg

For more information please contact miroslaw.staron@gu.se

INTERO

Software Center

(i) To better understand interoperability problems (ii) To identify interoperability dimensions on which to focus on for improvement (iii) To identify interoperability goals as well as conceive steps to reach such goals (iv) To reassess the interoperability of the modified systems



- Identified relevant interoperability dimensions, measures, and satisfaction values
- Put into practice the INTERO model through:
  - a) two experiences, one within Jeppesen-Boeing and one at Volvo GTT (master theses)
  - b) an experience within Axis (validation workshop)
- Provided initial structured guidelines/process about how to use INTERO

- R. Spalazzese, P. Pelliccione, U. Eklund. INTERO: an Interoperability Model for Large Systems. IEEE Software, 2017, to appear.

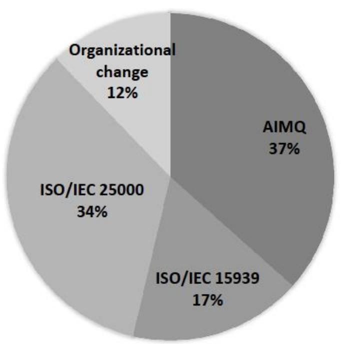
- Additional material about INTERO is available at: http://www.rominaspalazzese.com/INTERO-guidelines.pdf

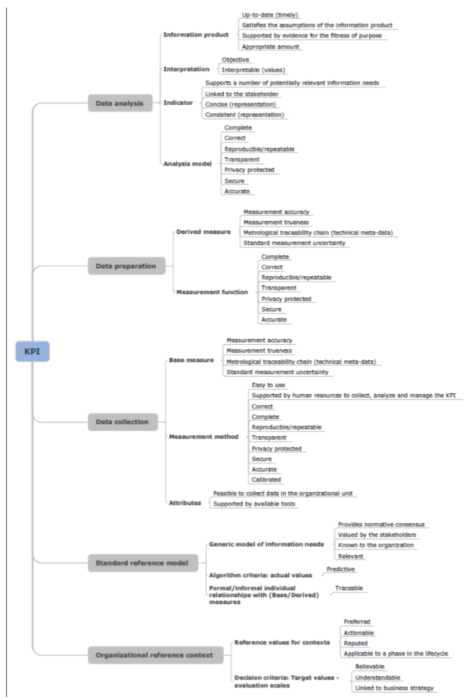
For more information please contact romina.spalazzese@mah.se

KPI QUALITY TOOL

Software Center

The KPI quality tool provides the organization with the possibility to assess whether a KPI is going to be useful, driving the right behavior and results.





- Quantifying the quality of KPIs leads to visual assessment of the quality
- The tool is based on ISO/IEC 25000 and 15939
- Low scores indicate KPIs which should be removed or reworked

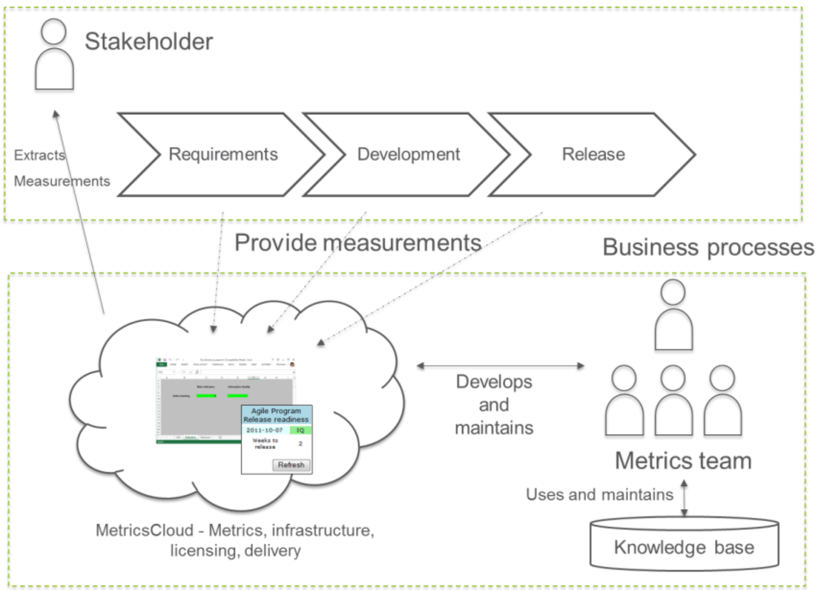
- Miroslaw Staron, Wilhelm Meding, Kent Niesel, Alain Abran, 'KPI quality model', in International Conference on Software Measurement (Mensura), 2016, in submission

For more information please contact miroslaw.staron@cse.gu.se

MaaS

Software Center

The Measurement-as-a-Service model optimizes the organization of measurement programs to on-demand deliver metrics maintaining the long-term competence.



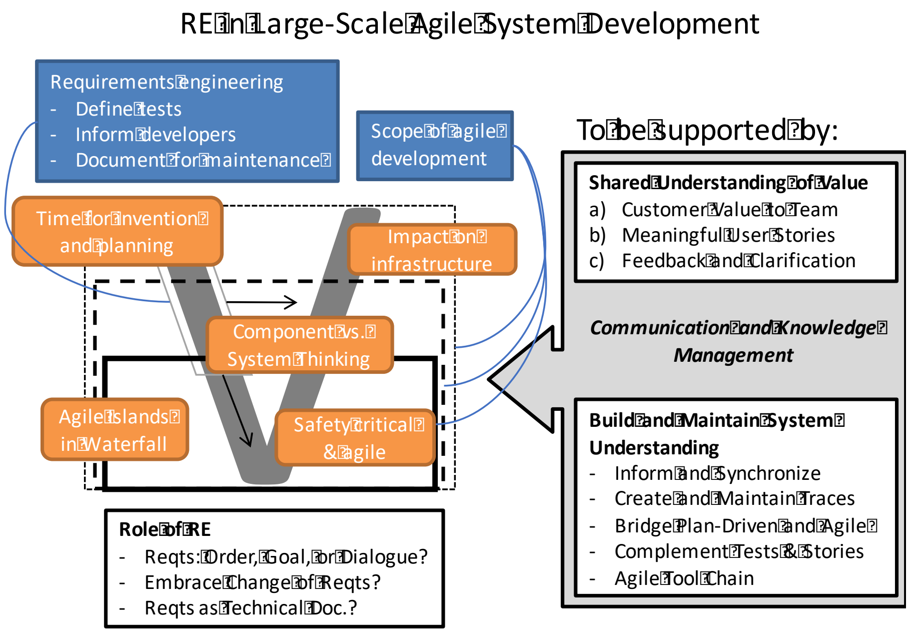
- Dynamically changing information needs are supported by long-term competence
- Technology and business competence are combined at one place
- MetricCloud supports the company-wide dissemination of metrics

- Miroslaw Staron, and Wilhelm Meding, 'Measurement-as-a-Service–a New Way of Organizing Measurement Programs in Large Software Development Companies', in International Conference on Software Measurement (Mensura), 2015

For more information please contact miroslaw.staron@cse.gu.se

MaRK-C

MaRK-C describes how to Manage Requirements Knowledge Continuously to support Large-Scale Agile System Development and supports balancing RE activities to support system engineering needs as well as agile development approaches.



- Requirements critical in agile systems engineering
- Depending on the scope of agile development, critical knowledge needs surface (orange boxes)
- A MaRK-C approach reinvestigates the role of RE and focusses on Communication and Knowledge management to facilitate shared understanding of value and system

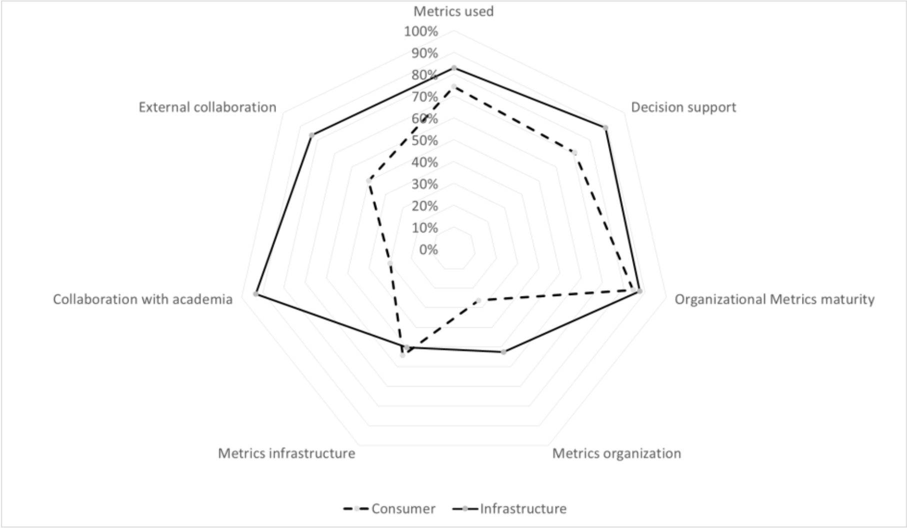
- Kasauli, R.; Liebel, G.; Knauss, E.; Gopakumar, S.; Kanagwa, B.: Requirements Engineering Challenges in Large-Scale Agile System Development. Submitted to RE conference, 2017
- Kasauli, R.; Knauss, E.; Nilsson, A. & Klug, S.: Adding Value Every Sprint: A Case Study on Large-Scale Continuous Requirements Engineering. In: Proc. of 3rd WS on Continuous Requirements Engineering, Essen, Germany, 2017

For more information please contact Eric Knauss <knauss@chalmers.se>

MESRAM

Software Center

Measurement program robustness assessment model (MeSRAM) lets the companies *stress-test* their metrics portfolio and identify weak-spots – measurement areas to improve.



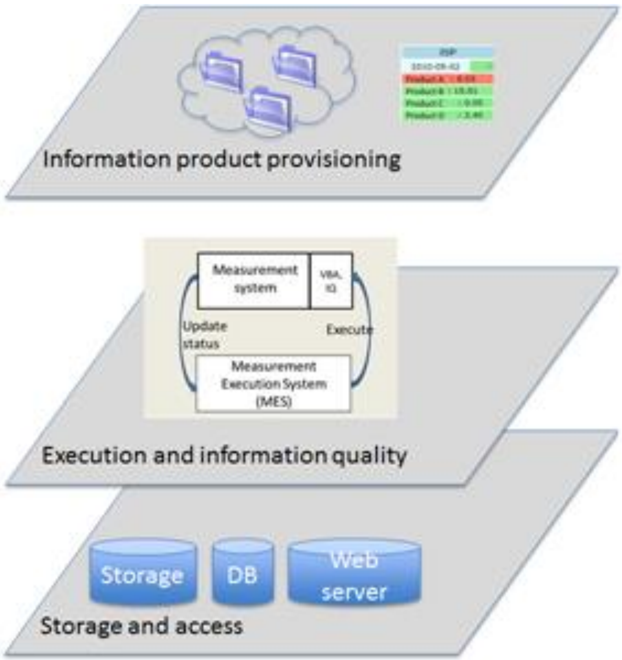
- History has a big impact on the measurement program waterfall -> Agile makes the program wider
- Agility in companies leads to deeper measurement programs (deeper adoption)
- Supplier-client relations lead to more metric-orientation

- M. Staron, W. Meding, "MeSRAM - A Method for Assessing Robustness of Measurement Programs in Large Software Development Organizations and Its Industrial Evaluation", Journal of Systems and Software, 2016

For more information please contact miroslaw.staron@cse.gu.se

MetricCloud

MetricCloud enables "always access" to information products without relaying on a single-point of failure in the organization.



- MetricCloud supports the company-wide dissemination of metrics
- Information accessible offline
- Simple, limited code needed to realize the MetricCloud concept

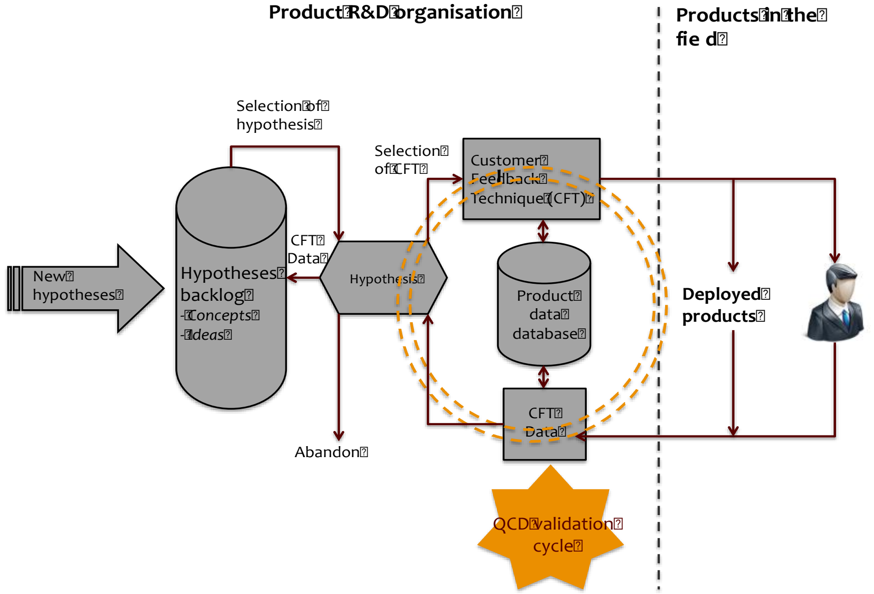
- Staron, M. and Meding, W., 2014. Metricscloud: Scaling-up metrics dissemination in large organizations. Advances in Software Engineering, 2014, p.8

For more information please contact miroslaw.staron@cse.gu.se

QCD

Software Center

The QCD model identifies qualitative and quantitative customer feedback techniques and helps companies select among these. The model helps companies continuously validate hypotheses and re-prioritize feature content pre-during and post development.



- By treating requirements as hypotheses, the QCD model helps companies ***continuously validate*** customer value
- By continuous validation of hypotheses, the QCD model enables ***re-prioritization of features*** also after development has started
- By identifying qualitative and quantitative customer feedback techniques (CFT:s), the QCD model helps companies answer both ***'what' and 'how/why'*** is customer value

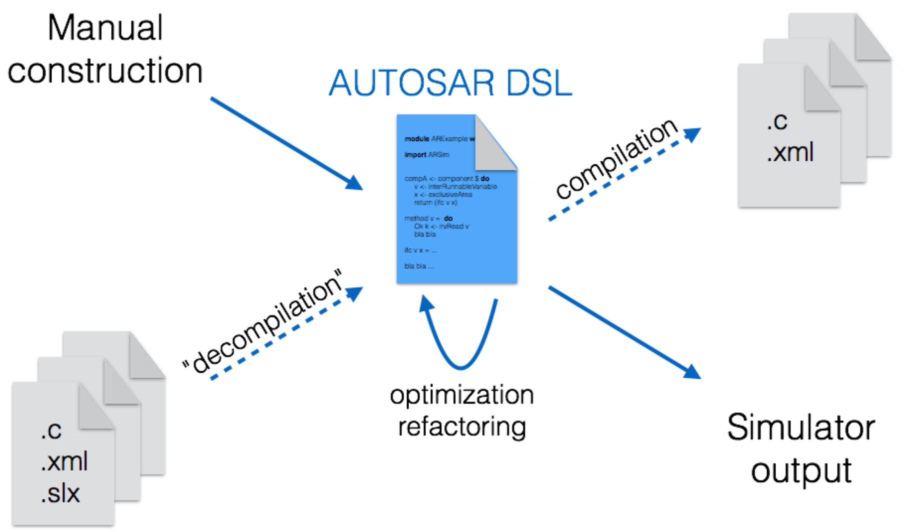
- Olsson, H.H., and Bosch, J. (2015). Towards Continuous Customer Validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation. In Proceedings of the 6th International Conference on Software Business (ICSOB). June 10-12, Braga, Portugal

For more information please contact helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

RAW FP

Software Center

Exploring Resource-Aware Functional Programming and embedded Domain-Specific Languages in a tool for platform-independent construction and simulation of AUTOSAR systems.



- The AUTOSAR standard (autosar.org) is intertwined with platform dependencies and implementation language concerns.
- Based on a formalized semantics, our AUTOSAR DSL allows software components to be developed and tested without prior commitments to a particular platform.
- Central to our simulation tool is a random scheduler with Simulink and QuickCheck intergration.

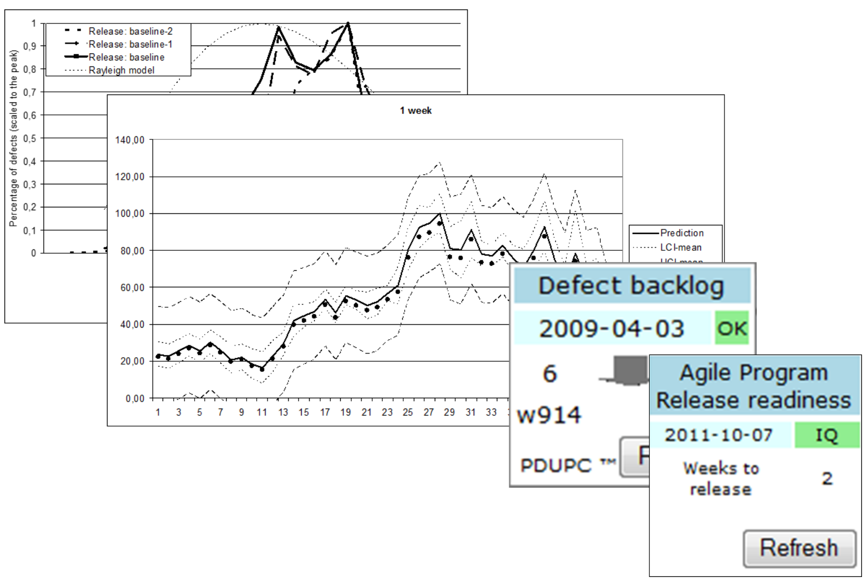
- Josef Svenningsson and Emil Axelsson: "Combining deep and shallow embedding of domain-specific languages." In *Computer Languages, Systems & Structures*, vol 44, pp 143-165, 2015.
- Johan Nordlander and Patrik Jansson: "A semantics of core AUTOSAR." Preprint and source code available at GitHub, 2016.

For more information please contact johan.nordlander@dataductus.se or patrik.jansson@chalmers.se

RELEASE READINESS

Software Center

Software reliability growth modelling supports the companies in optimizing the test allocation.



- Defects are discovered in patterns
- Understanding the right pattern makes the predictions more correct
- Combining short- and long term predictions provide the ability to make better release readiness decisions

- Staron, M., Meding, W. and Palm, K., 2012. Release readiness indicator for mature agile and lean software development projects. In *Agile Processes in Software Engineering and Extreme Programming* (pp. 93-107). Springer Berlin Heidelberg
- Staron, M. and Meding, W., 2008. Predicting weekly defect inflow in large software projects based on project planning and test status. Information and Software Technology, 50(7), pp.782-796

For more information please contact miroslaw.staron@gu.se

RENDEX

Software Center

Rendex is a measurement-based method for automated quality assessment of textual software requirements. The method can detect about 70-80% of such requirements that need improvements before the software design.

Requirement quality metric | Functional safet

Name	Quality index	Item type
Safety concept	1	Functional Requirement
In-signal deactivated mode	13	Functional Requirement
ECU Deactivated mode	3	Functional Requirement
AEBS deactivated mode	9	Functional Requirement
Emergency brake inhibited mode	8	Functional Requirement
Sensor installation and alignment	5	Functional Requirement
Sensor safety concept	13	Functional Requirement
Sensor output information at failure	3	Functional Requirement
Sensor data confidence	1	Functional Requirement
Sensor message counters	3	Functional Requirement
Sensor returning vehicle data	1	Functional Requirement
Sensor monitoring of vehicle input data	2	Functional Requirement
sensor information confidence and redundancy	5	Functional Requirement
AEBS message counters	4	Functional Requirement
Experience based development	3	Functional Requirement
Design pattern	0	Functional Requirement
Securing AEBS on/off	8	Functional Requirement
Monitor layer 2	14	Functional Requirement
Fully redundant calculations by Monitor	15	Functional Requirement
Plausibility check	4	Functional Requirement
Monitor layer 3	2	Functional Requirement

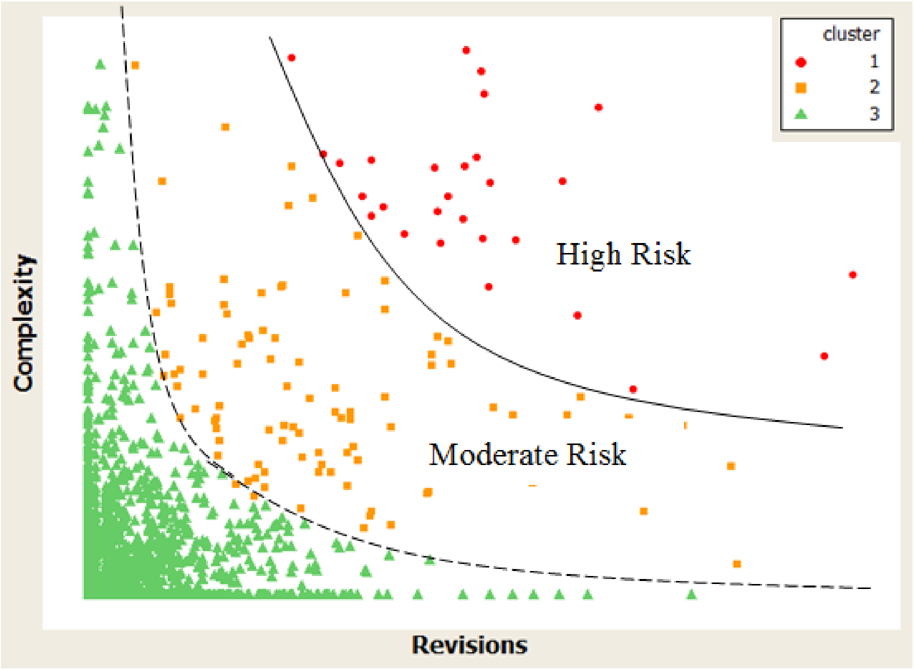
- Rendex decrease the requirement review time from several weeks to several minutes
- Rendex detects needed improvements by 70-80% accuracy
- Rendex permits proactive requirements quality control

- Rendex: A Method for Automated Reviews of Textual Requirements (under revision in TSE)
- A Complexity Measure for Textual Requirements, *International Conference on Software Process and Product Measurement (IWSM-MENSURA), IEEE, 2016*

For more information please contact vard.antinyan@gu.se or Miroslaw.Staron@cse.gu.se

RISKY FILES

Risky Files is a measurement-based method for automated identification of source files that are error-prone and difficult-to-maintain. The method can detect about 70-80% of such files that need attention before merging them to the main code branch of the product.



Model Findings/Arguments:

- Files that are complex and change frequently are error-prone and difficult-to-maintain
- There are only few files out of thousands, that are risky at a given point of development time
- Those files can be found proactively by Risky Files method

- Antinyan, Vard, et al. "Identifying risky areas of software code in Agile/Lean software development: An industrial experience report." *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE),* IEEE, 2014
- Antinyan, Vard, et al. "Monitoring Evolution of Code Complexity and Magnitude of Changes." *Acta Cybern.* 21.3 (2014): 367-382.

For more information please contact vard@chalmers.se, wilhelm.meding@ericsson.com, Miroslaw.Staron@cse.gu.se

RI-Speed model

Software Center

RI-Speed model provides a guide on how to balance speed of reviews with the speed/quality of integration

Speed

Phase	Slow	Fast
Reviews	• Selected roles can do +2 • Review comes as a low priority • Internal + external review • All code commits get the full review	• Reminders about overdue reviews • Small commits • Close proximity of reviewers/submitters • Review is prioritized • Frequent commits • Trust / review culture • Automated code checks used in the review • Selected commits get the full review
Integration	• Difficult to divide features in small commits • Done only in selected hours/time frame	• Multiple code repositories • Thorough code reviews – all commits get full review • Small commits • *Integrate often, integrate little* • Done 24/7

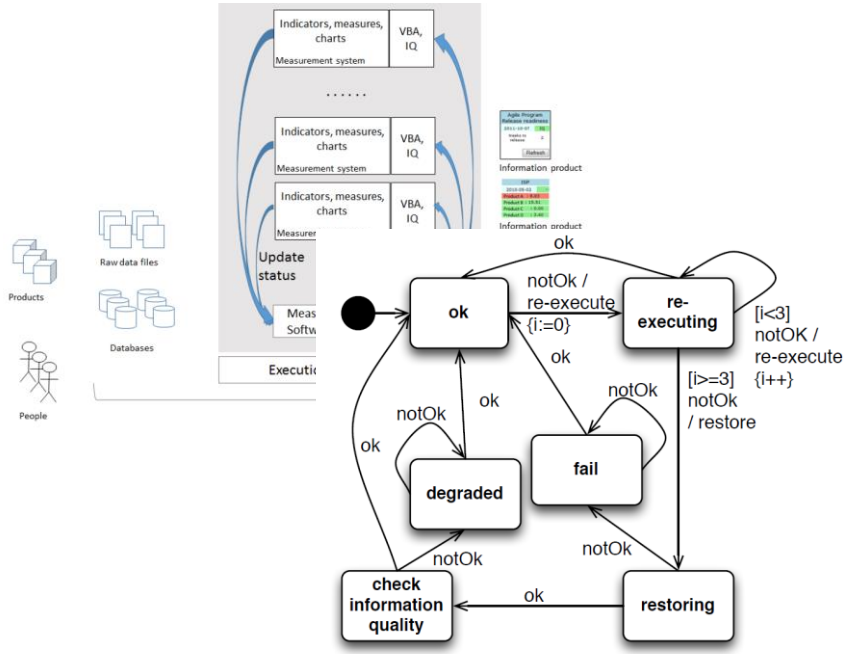
- Reviews and integration can be balanced to find the optimal speed development of software
- In the model we developed the measurement instruments for measuring speed
- Location of the code, size of the commit and organizational closeness have the highest influence on the speed

- In praparation

For more information please contact miroslaw.staron@gu.se

SELF-HEALING

The self-healing model automatically repairs measurement systems when these crash due to infrastructure changes, file aging and low information quality to reduce the maintenance effort of the measurement program.



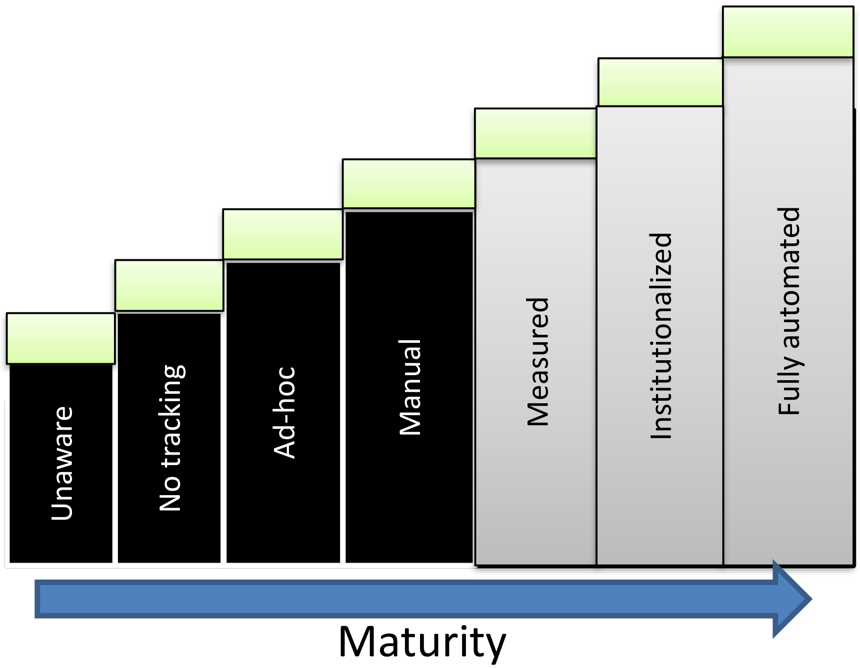
- Self-healing reduces the weekly maintenance effort from hours to minutes
- A simplistic MAPE-K implementation allows to stay on transparent technology much longer
- Information quality supports repairing of semantic errors in measurement systems

- Staron, Miroslaw; Meding, Wilhelm, “Industrial Self-healing Measurement Systems”, Continuous Software Engineering, edited by Jan Bosch, Springer-Verlag, 2014

For more information please contact miroslaw.staron@cse.gu.se

SAMTTD

SAMTTD (Strategic Adoption Model for Tracking Technical Debt) is a maturity model for the introduction of Technical Debt Management in large companies. We studied several companies, with a survey in 15 organizations (226 answers) and 3 in-depth case studies.



- On average, **25 % of development is spent** on Managing TD
- Many companies are not mature in tracking TD**: 65 %** are in the "**no tracking**" spot, only **7** % are in "**Manual**"
- Managing TD needs some initial **funds and activities** (preparation), continuous **budget**, and clear **responsible** in the organization
- Tools such as **static analyzers** and TD **backlogs reduce** management overhead

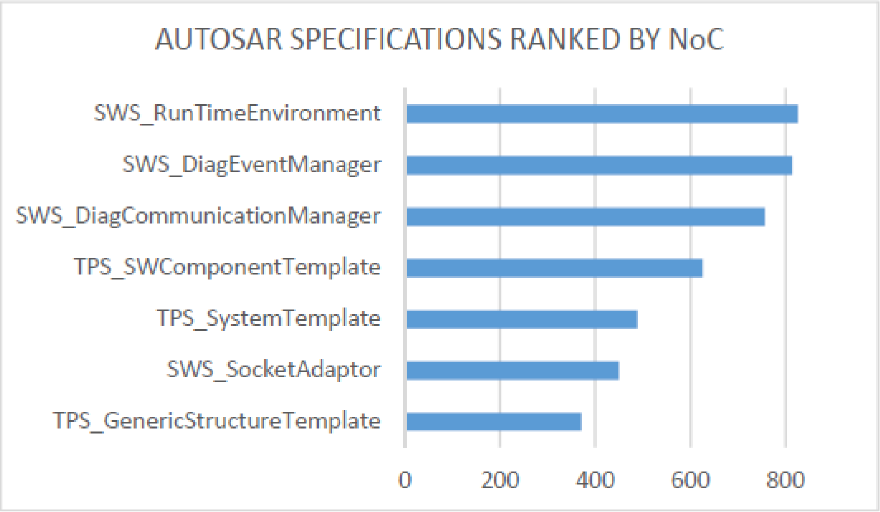
- A. Martini, T. Besker, and J. Bosch, "The Introduction of Technical Debt Tracking in Large Companies," in *accepted at APSEC 2016,*

For more information please contact Antonio Martini: antonio.martini.am@gmail.com

SREA

Software Center

Standardized requirements evolution assessment method (SREA) supports the companies in analyzing the evolution of system requirements from rapidly changing industrial standards.



- In order to use new standardized features, new releases of the standards need to be adopted together with their requirements.
- This requires thorough analysis of the requirements which can be time-consuming.
- SREA method can facilitate this analysis by identifying the most unstable specifications from the standards and their requirements.

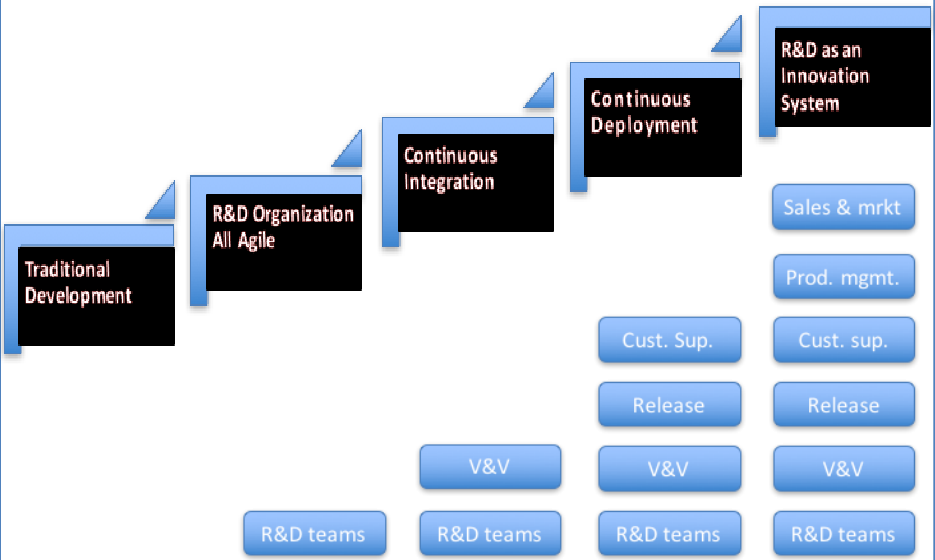
- C. Motta, D. Durisic and M. Staron. "Should We Adopt a New Version of a Standard? – A Method and Its Evaluation on AUTOSAR." In *Proceedings of the 17th International Conference on Product-Focused Software Process Improvement*, pp. 127-143. 2016

For more information please contact Darko Durisic.

STAIRWAY TO HEAVEN

Software Center

The Stairway to Heaven Model describes the stages that companies evolve through when adopting novel approaches to software engineering.



- Companies move through a predictable and repeatable pattern over time when evolving software engineering practices
- Each transition has business, architectural, process and organizational implications
- The higher up the stairway an organization climbs, the more organizational units are affected

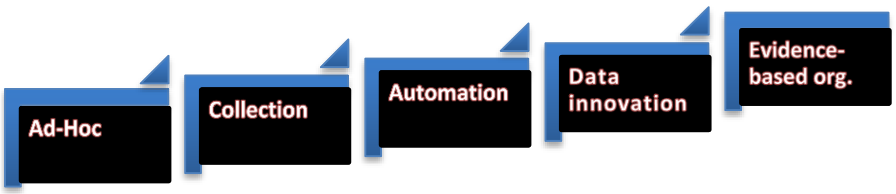
- H.H. Olsson, H. Alahyari, J. Bosch, Climbing the"" Stairway to Heaven --A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software, 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA),, pp. 392-399, IEEE, 2012

For more information please contact jan.bosch@chalmers.se

StH: DATA DIMENSION

Software Center

The 'Stairway to Heaven: Data Dimension' details a predictable set of steps that software-intensive companies move through as they transition towards evidence-based organizations in which data informs processes at all levels in the organization.



- The model outlines the transition towards a ***data-driven company*** characterized by rapid, informed and evidence-based decision-making.
- The model helps companies move away from ***decision-making*** based on opinions towards decision-making based on data.
- The model is concerned with the organizational ***change processes*** that companies evolve through when adopting data-driven development practices.

- Bosch, J., and Olsson, H.H. (2017). Towards Evidence-Based Organizations: Learnings From Embedded Systems, Online Games And Internet of Things. *To appear in IEEE Software (forthcoming).*

For more information please contact jan.bosch@chalmers.se and/or helena.holmstrom.olsson@mah.se.

TBCES

The Benefits of Controlled Experimentation at Scale

Software Center

The value of Controlled Experimentation (CE) extends beyond finding the 'better' feature or a product version. Our model (1) identifies ***where*** companies can benefit from experimentation, and (2) provides guidance on ***how*** to achieve these benefits.

	Benefits	Guidelines to achieve the benefit
Portfolio	Value discovery and validation	(1) Customer and Business value are hypothesized on a portfolio level. (2) Measurement of the value is formalized in terms of leading metrics. (3) Hypotheses are evaluated in multiple experiments across multiple products. (4) Hypotheses that were confirmed indicate value on a portfolio level.
Product	Incremental product improvements	(1) Instrumentation data of a single experiment are collected, (2) Metrics are calculated based on the collected data, (3) Statistical difference between variants is measured, (4) Variants with improvements to key metrics are deployed.
	Optimizing and predicting product infrastructure and capacity needs	(1) Change is deployed on a low % of treatment, (2) Changes in infrastructure are monitored, (3) Treatment group is gradually increased if resources allow.
	Ensuring product quality	(1) Product changes that degrade key metrics are not deployed.
	Stabilizing and lowering product complexity	(1) Product increments with no impact on the key metrics are not deployed. (2) Product features with no impact on the key metrics are removed with reverse experiments.
	Product Instrumentation quality assurance	(1) A/A experiments are conducted to identify noisy instrumentation. (2) Experiments with known outcomes validate instrumentation quality.
Team	Team activity planning	(1) Changes/features that improve key metrics are shared among teams. (2) Teams generalize learnings to identify feature areas that should be prioritized.
	Defining performance goals for teams	(1) By measuring the exact amount of impact that changes of one team had on the leading metrics over a period, a realistic goal can be set for the next period.

We identify benefits on three levels:

- **portfolio level,**
- **product level**,
- **team level**.

CE enables more accurate planning of portfolio, product, and team work.

With CE, companies can identify relationships between metrics, set and measure perf. goals for teams, reduce product complexity, predict infra. needs, and detect quality issues.

- A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch (2017), "The Benefits of Controlled Experimentation at Scale," In Proceedings of the 43th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, Austria. 30 Aug.-1 Sept., 2017.

For more information please contact aleksander.fabijan@mah.se

Team Metrics Portfolio

Software Center

The team metrics portfolio gives the teams:
a) a list of team related measures to choose from, and
b) a list of prioritized measures.

Examples of team measures:

- Team size
- Team member loading
- Workload
- Multidisciplinary teams
- % self-organizing teams
- Rewards of success
- Obstacles
- Creativity
- People turnover
- Awareness of Ops

- Gives a list of team related measures.
- The list comprises both theory and software industry best practices.
- If necessary, the list provides also the top measures that teams should have.

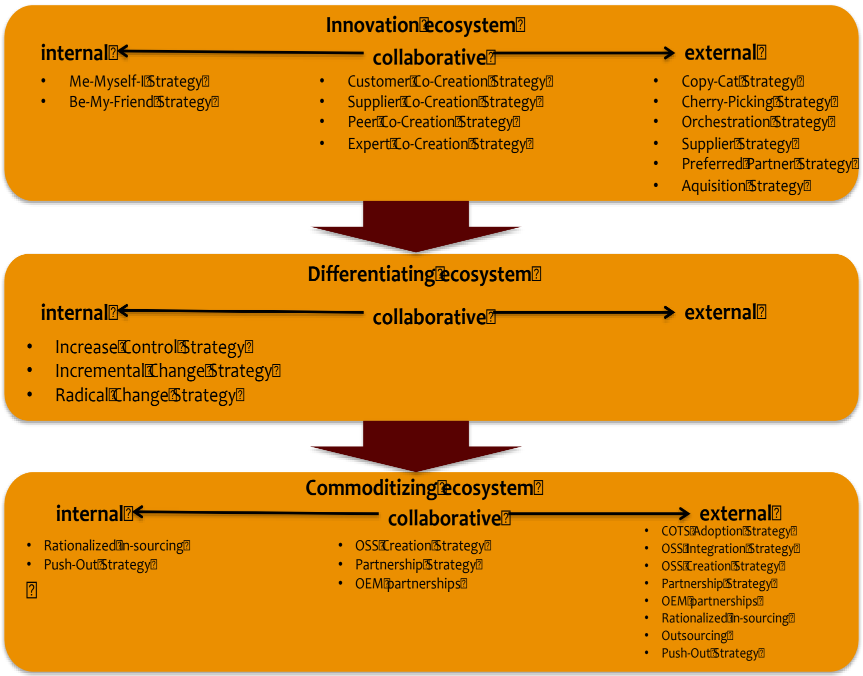
- W. Meding, "Effective monitoring of progress of agile software development teams, in modern software companies – an industrial case study", under revision.

For more information please contact Miroslaw.Staron@cse.gu.se

TeLESM

Software Center

The TeLESM model distinguishes between three types of ecosystems and identifies strategies for how to manage partners within each of these. The model helps companies in moving towards strategic management of their ecosystems.



- **TeLESM** distinguishes between the innovation, the differentiating and the commoditizing ecosystems and identifies strategies for managing each of these.
- **TeLESM** helps companies select the optimal strategies for managing each ecosystem.
- **TeLESM** helps companies identify when to transfer functionality between ecosystems to focus R&D resources on differentiating and innovative functionality.

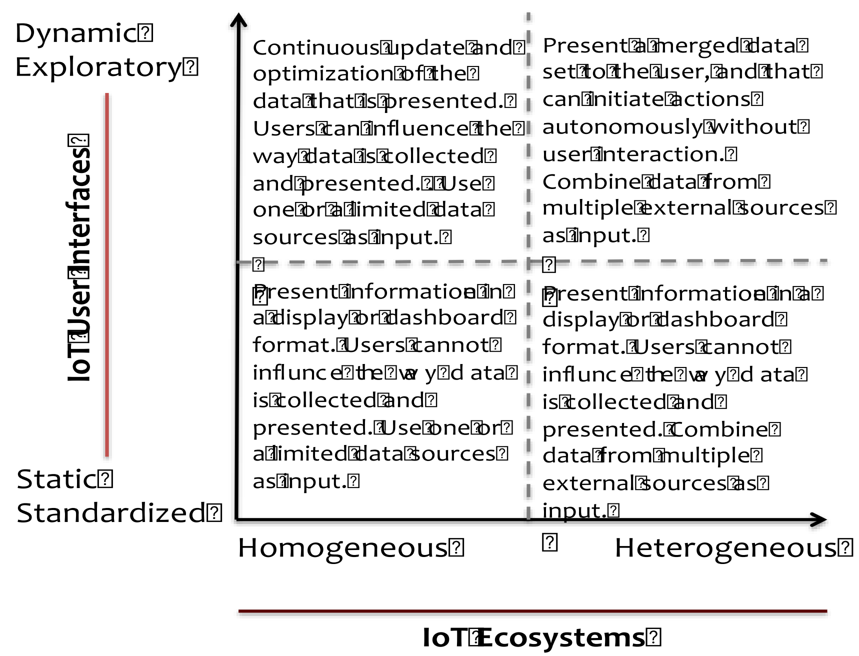
- Olsson, H.H., and Bosch, J. (2015). Strategic Ecosystem Management: A multi-case study on challenges and strategies for different ecosystem types. In Proceedings of the 41st Euromicro Conference series on Software Engineering and Advanced Applications (SEAA), August 26-28th, Madeira, Portugal

For more information please contact helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

UDIT

Software Center

The UDIT model helps companies assess two dimensions of IoT systems. Companies can use the model to: (1) identify current state of their systems, (2) identify desired state and (3) identify the steps necessary to develop more advanced IoT systems.



- The IoT User Interface dimension identifies the format in which data is presented to users and how users interact with IoT systems
- The IoT ecosystem dimension defines the level of which IoT systems interconnect with external systems
- The UDIT model Identifies the desired transition towards multi-source systems that require less interaction from the user

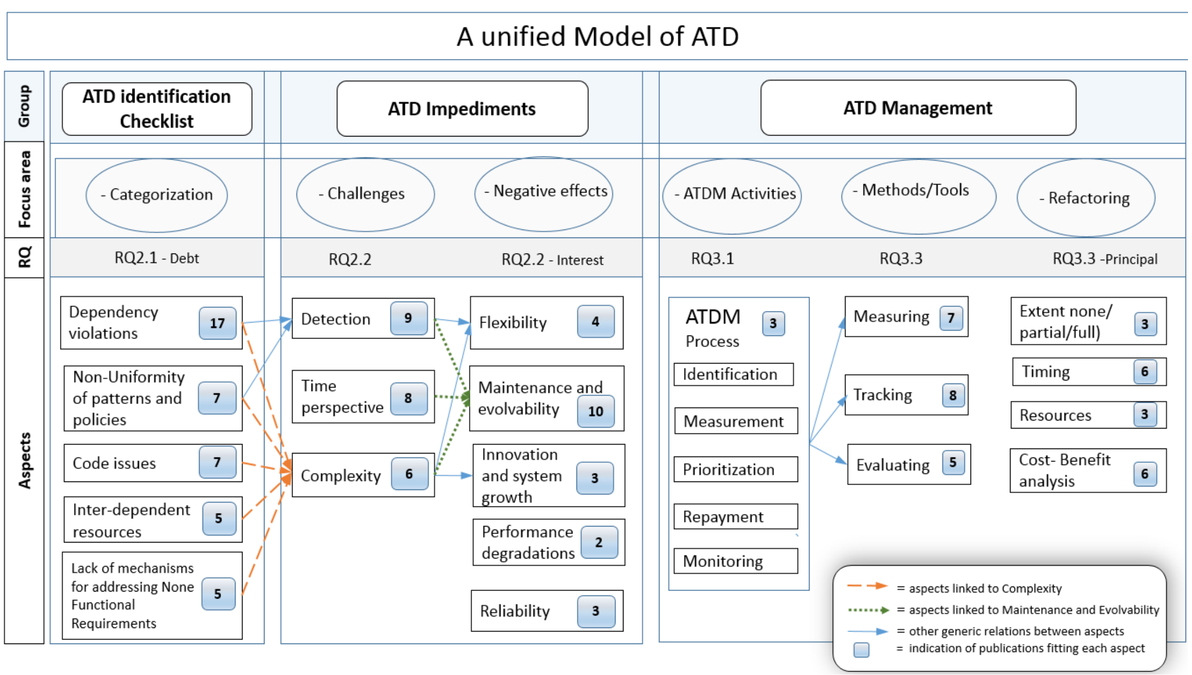
- Olsson, H.H., Bosch, J., and Katumba, B. (2016). User Dimensions In ‘Internet of Things’ Systems: The UDIT Model. In Proceedings of the 7th International Conference on Software Business (ICSOB), June 13-14, Ljubljana, Slovenia

For more information please contact helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

UniMATeD

Software Center

UniMATeD (Unified Model for Architectural Technical Debt) is a descriptive model that provides an overall understanding of Architectural Technical Debt (ATD), both in terms of a checklist, impediments, and different management strategies.



Model Findings:

- ATD can be classified in different categories
- ATD has several challenges and negative effects
- The ATD management includes processes, method/tools and refactoring strategies

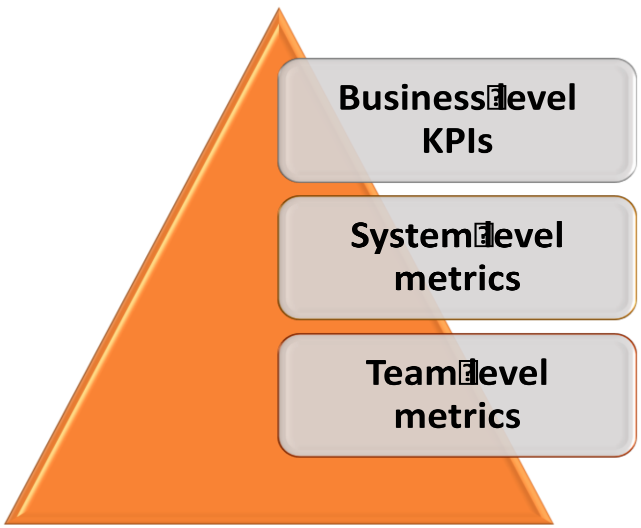
- T. Besker, A. Martini, and J. Bosch, "A Systematic Literature Review and a Unified Model of ATD," in 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2016, pp. 189-197.

For more information please contact besker@chalmers.se

VALUE FACTOR NETWORK

Software Center

The ‘Value Factor Network’ recognizes the challenge with aligning business level KPIs and team level metrics during experimentation. The model helps companies define key metrics to avoid sub-optimization and accelerate the impact of experiments.



- The model increases the awareness of experiments as part of a larger business context where value modeling on all levels of the business is critical
- The model is a systematic approach to value modeling that helps companies identify the values they optimize for
- The model defines ten activities critical for systematic design, execution and evaluation of feature experiments and results in a quantitative equation that enables statistical validation of feature value

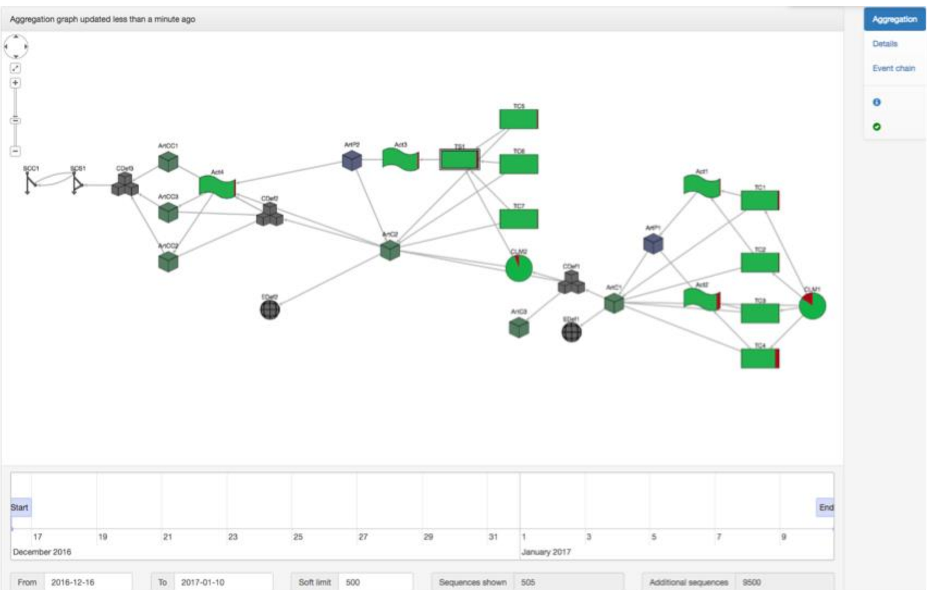
- Olsson, H., and Bosch, J. (2017). So Much Data – So Little Value: A multi-case study on improving the impact of data-driven development practices. *In Proceedings of the Ibero American Conference on Software Engineering (CIbSE), May 22nd – 23rd, Buenos Aires, Argentina.*

For more information please contact helena.holmstrom.olsson@mah.se and/or jan.bosch@chalmers.se

ViCI - Visualization of Continuous Integration

Software Center

Rich realtime, visual representations of aggregate and detailed Eiffel workflows to enable advanced analysis for multiple stakeholders.



The visualisation contains three visualisation levels:

- The aggregate view shows different types of Eiffel events and products, with relationships and status where applicable,
- events can be viewed and filtered by metadata,
- and individual events can be selected to drill down for causes to problems

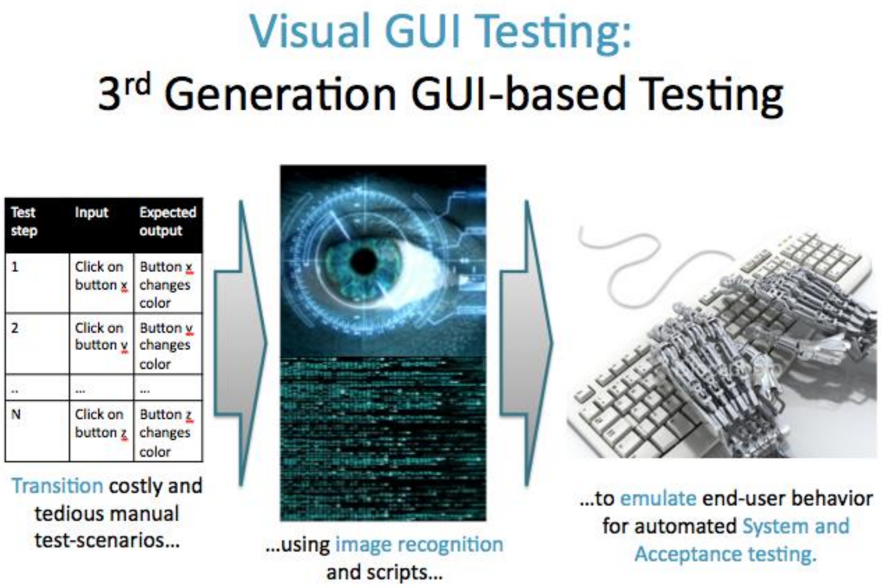
https://gitlab.ida.liu.se/tddd96/visualization
http://pum-2-1.pum-2017.ida.liu.se:3000/

For more information please contact Kristian Sandahl kristian.sandahl@liu.se or Ola Leifler ola.leifler@liu.se

VISUAL GUI TESTING

Software Center

In order to research higher levels of continuous development, automated testing is required on all levels of system abstraction. Visual GUI Testing provides a technical solution for GUI-based testing for automated system and Acceptance testing.



Visual GUI Testing enables:

- Testing of systems that previously lacked automated test support.
- Enables automation of high-level system and acceptance tests
- Can be applied to almost all GUI-based systems
- Lowers cost, tediousness and error-proneness compared to manual GUI-based testing

- E. Alégroth, “Visual GUI Testing: Automating High-level Software Testing in Industrial Practice”, PhD Thesis, Chalmers, 2015
- E. Alégroth, R. Feldt, P. Kolström, “Maintenance of Automated Testing in Industry: An Empirical study on Visual GUI Testing”, Information and Software Technology Journal, vol 73, p66-80, 2016

For more information please contact emil.alegroth@chalmers.se or emil.alegroth@bth.se

X-CODE CLONE (XCC)

Software Center

Cloning of the code can be both positive or negative, depending on the location, type and criticality of the cloned code. The XCC model allows to identify clones which can significantly hinder effective product development.

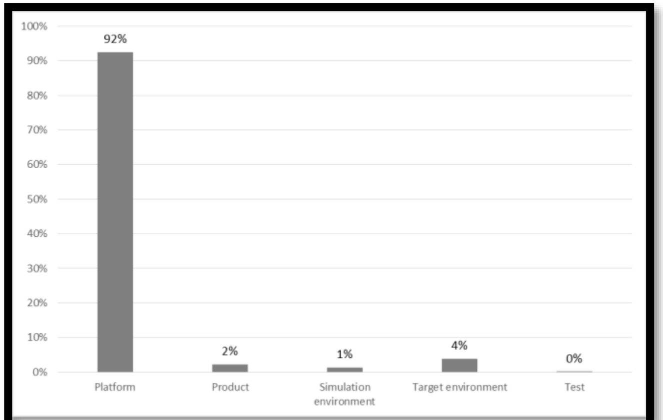


Figure 11. Types of clones in Product A

- Location of the clone is the primary determinant of its significance
- If left unmanaged, cloning can be a hinder of efficient development
- Ca. 4% of the clones in the studied projects could be considered obstructive/significant

- Staron, M., Meding, W., Eriksson, P., Nilsson, J., Lövgren, N. and Österström, P., 2015. Classifying Obstructive and Nonobstructive Code Clones of Type I Using Simplified Classification Scheme: A Case Study.*Advances in Software Engineering, 2015*.

For more information please contact miroslaw.staron@gu.se