Multi-view management of stakeholders' concerns: from architectures back to requirements

@RESG March 2017

Henry Muccini
DISIM Department
University of L' Aquila, L'Aquila, Italy
www.henrymuccini.com - henry.muccini@univaq.it - @muccinihenry

disim

Henry Muccini (academics)

On developing methods and tools for the analysis and design of software architectures

- Interoperable and Multi-view Software Architecture Description
- Architecting Situational Aware Cyber Physical Systems
- Group Decision Making in Software Architecting
- Collaborative Modeling in MDE

Henry Muccini (industrial)

nexpecto
spin-off of the University of L'Aquila

- Crowd management Indoor | outdoor
- Queue forecast and avoidance
- Multi-site crowd management
- Digital Booking and Ticketing

This Talk

- Part I: conceptual reasoning on multi-view multi-stakeholders and concerns
- Part II: concrete industrial case


Multi-view management of stakeholders' concerns: from architectures back to requirements
@RESG March 2017
Henry Muccini
DISIM Department
University of L' Aquila, L'Aquila, Italy
www.henrymuccini.com - henry.muccini@univaq.it - @muccinihenry
disim

Architecture as a set of Views

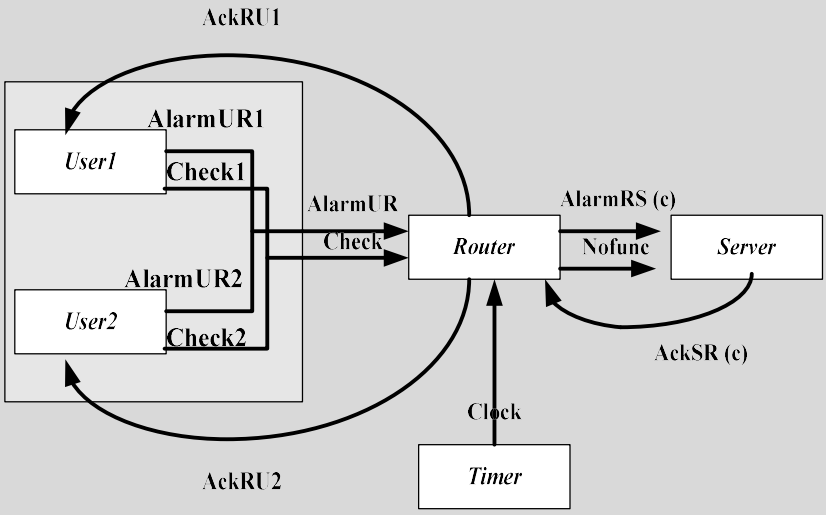
History – Architecture as structure

25+ years back

(maybe more…)

Software Architect

Software Architecture as a collection of computational components interacting through connectors



History – Architecture as behavior

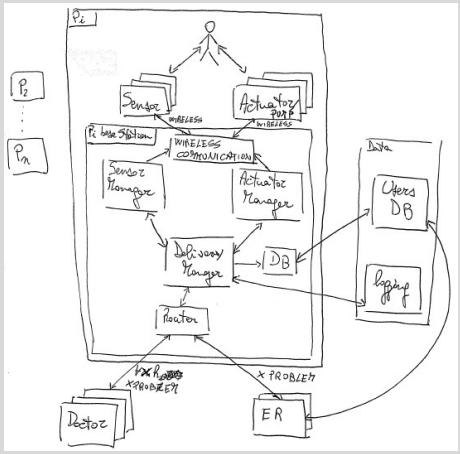
Academia

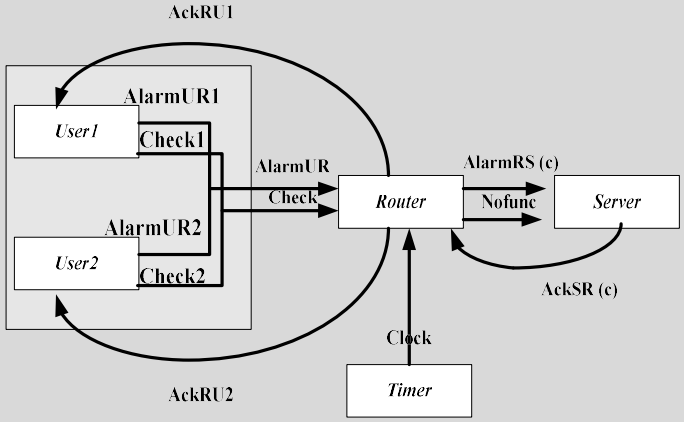
But architecting makes sense if we can run some automated analysis (and more)!

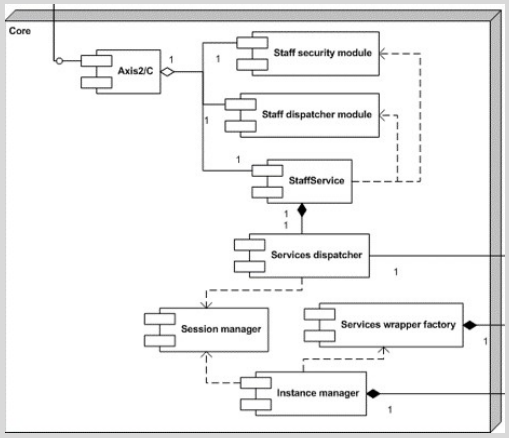
"Aside from providing clear and precise documentation, the primary purpose of specifications is to provide automated analysis of the document and to expose various kinds of problems that would otherwise go undetected" (PW1992)

"Fourth, an architectural system representation is often essential to the analysis and description of the highl-level properties of a complex system" (GS1994)

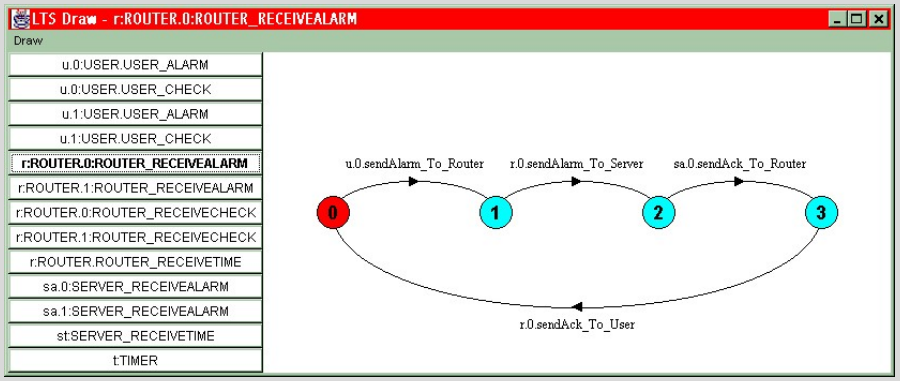
Architecture as Structure and Behavior

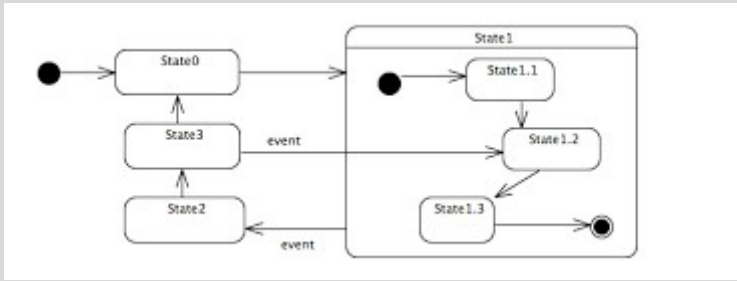






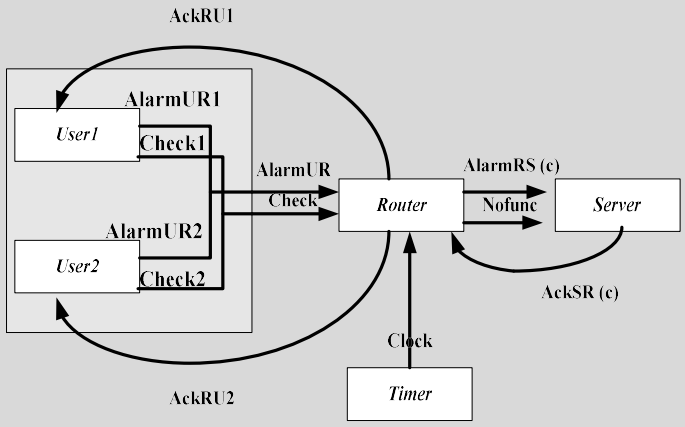
Structural Component and Connector View

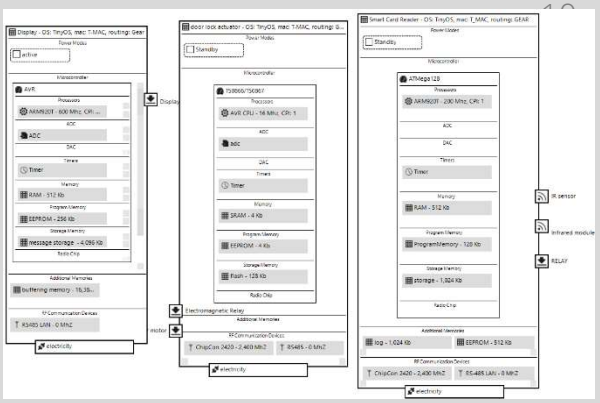


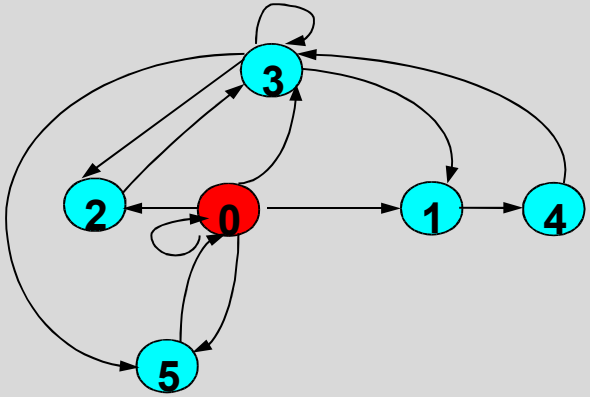


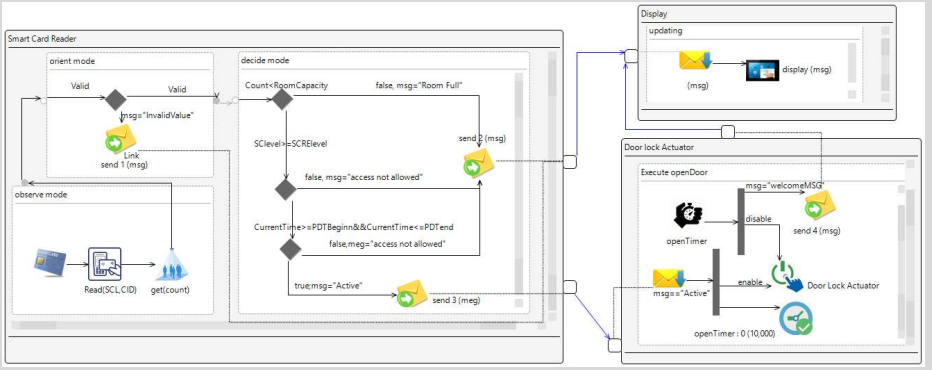
Behavioral View

Mono- vs Multi-Views



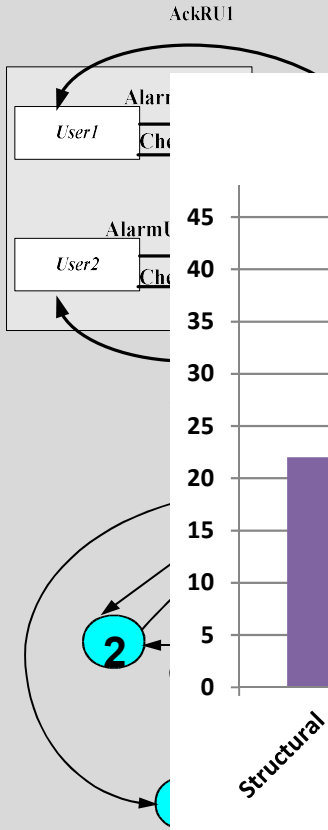


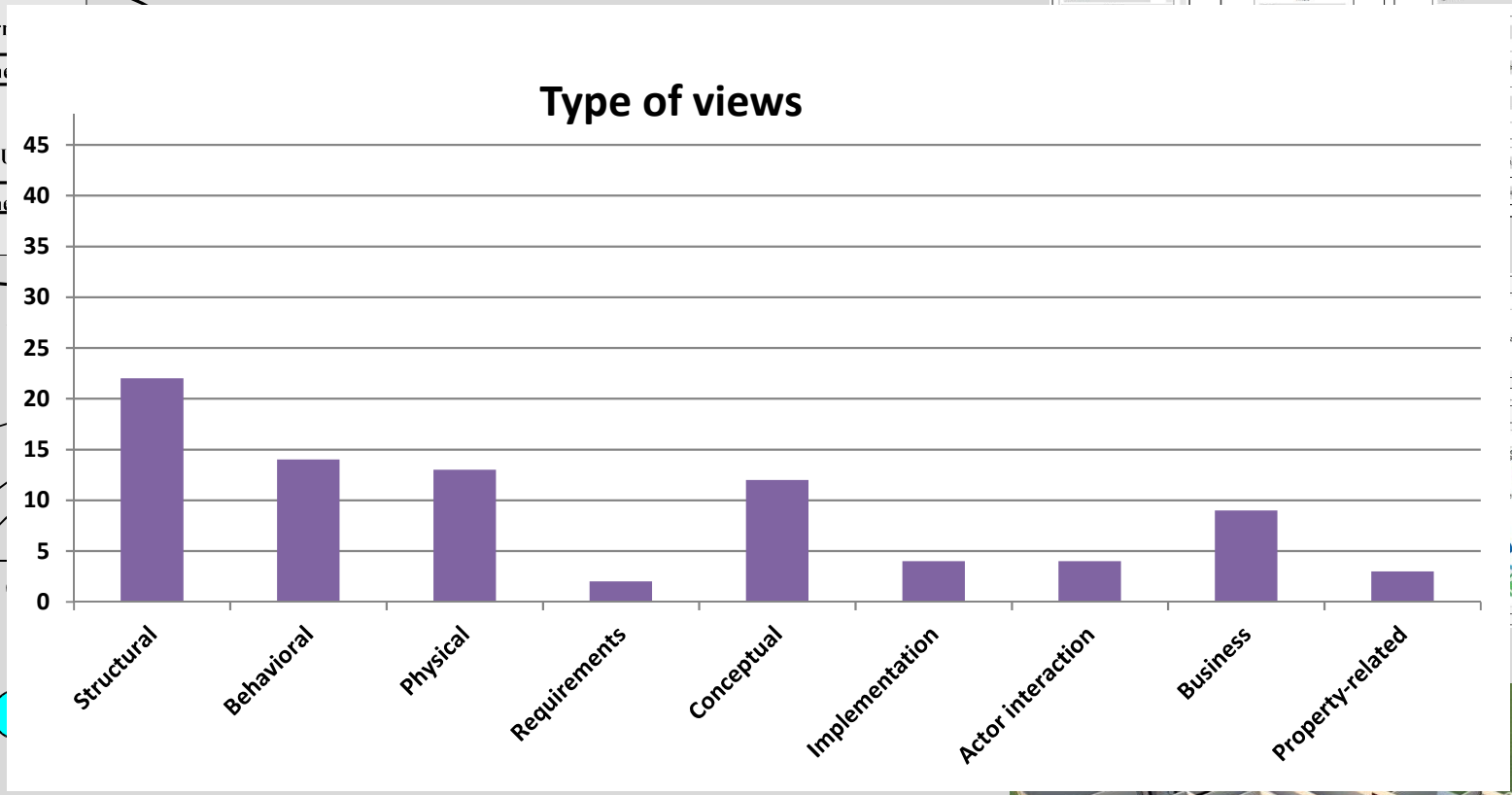


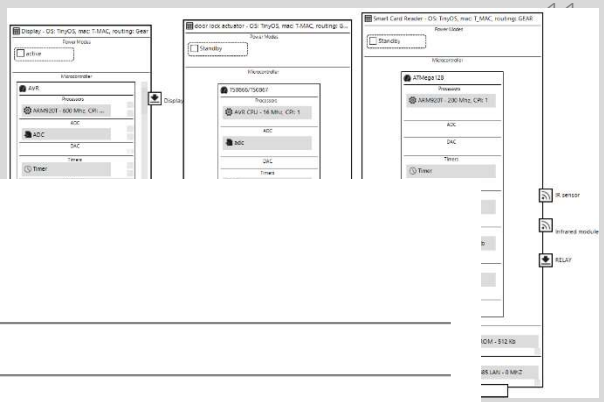


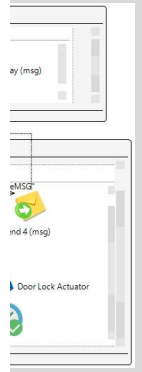
Year	2010	2011	2012	2013	2014	2015	2016	2017	2018
Income	0	2	4	6	8	12	14	16	18
Expenses	(13)	(1)	(1)	(1)	(2)	(15)	(3)	(4)	(5)
Profit	(13)	1	3	5	6	(3)	11	12	13
ROI	(13)	(12)	(9)	(4)	2	(1)	10	22	35

Mono- vs Multi-Views









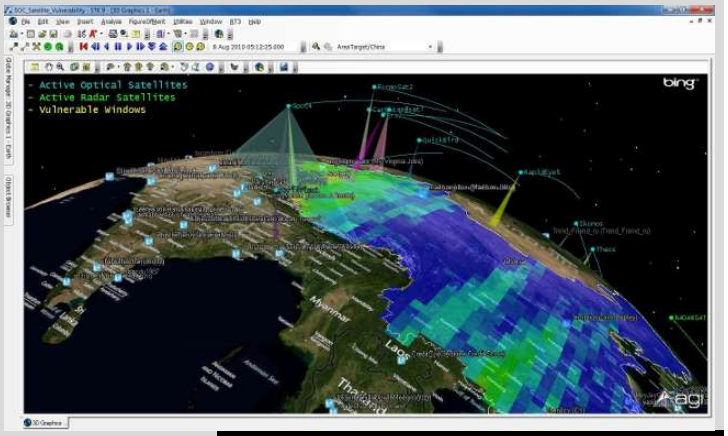
Year	2010	2011	2012	2013	2014	2015	2016	2017	2018
Income	0	2	4	6	8	12	14	16	18
Expenses	(13)	(1)	(1)	(1)	(2)	(15)	(3)	(4)	(5)
Profit	(13)	1	3	5	6	(3)	11	12	13
ROI	(13)	(12)	(9)	(4)	2	(1)	10	22	35

Why all those different viewpoints?

Multi-Views for multi-stakeholders

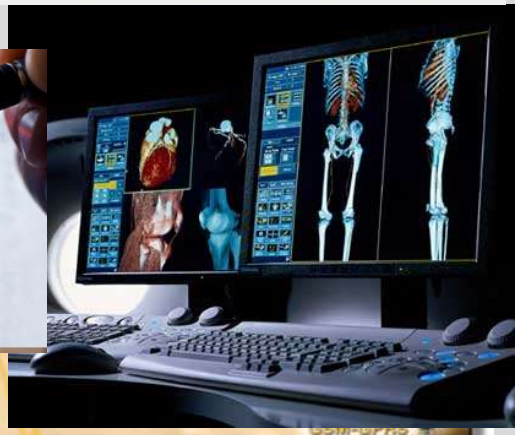
It is common practice to use multiple views and viewpoints to deal with multiple **stakeholders** and their different **concerns**

Stakeholder concerns



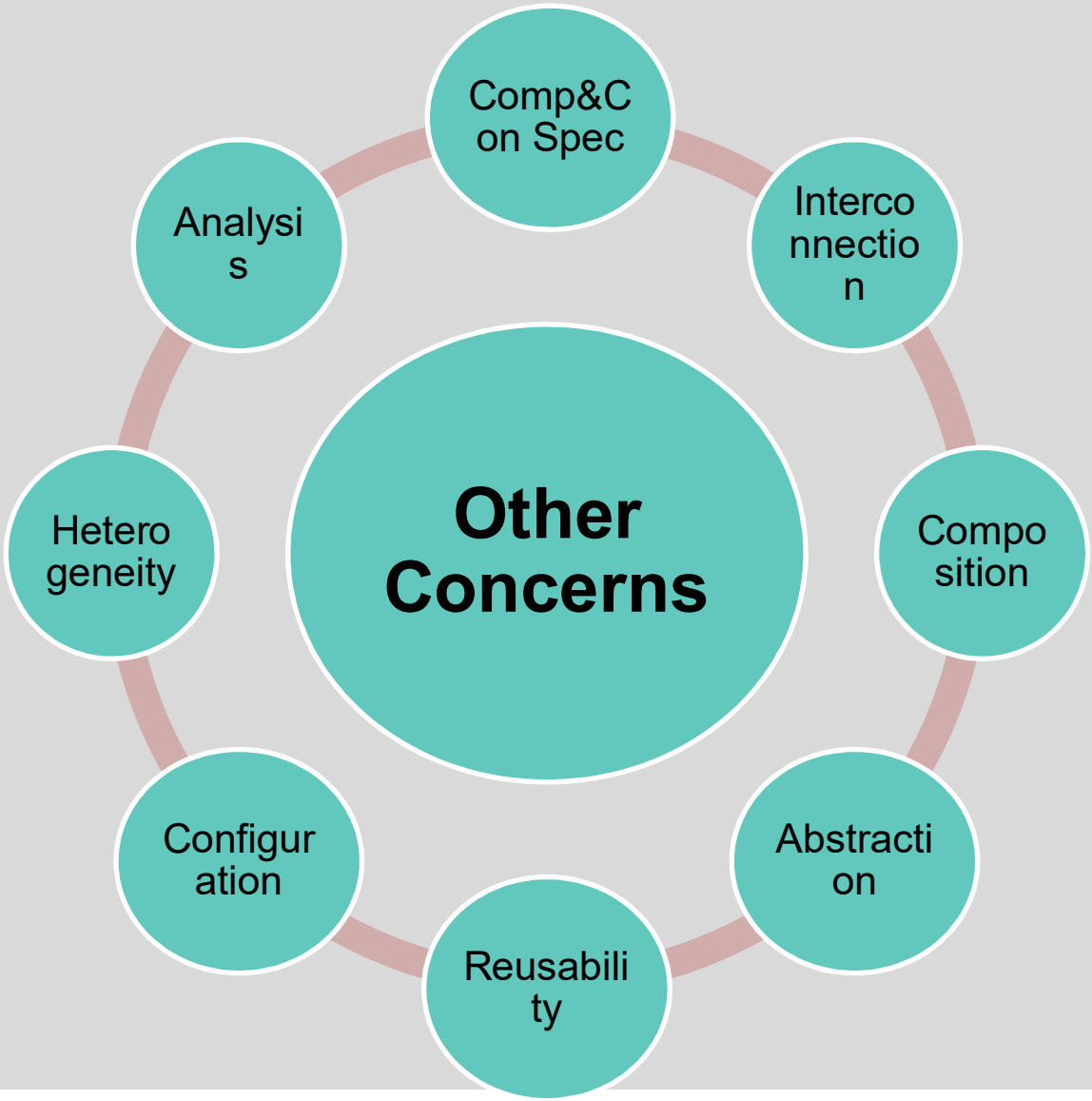
Stakeholders concerns can vary tremendously, depending on:

- the nature of the system
- project-specific constraints
- organizational constraints
- the application domain
- …



disim

Stakeholder architectural concerns



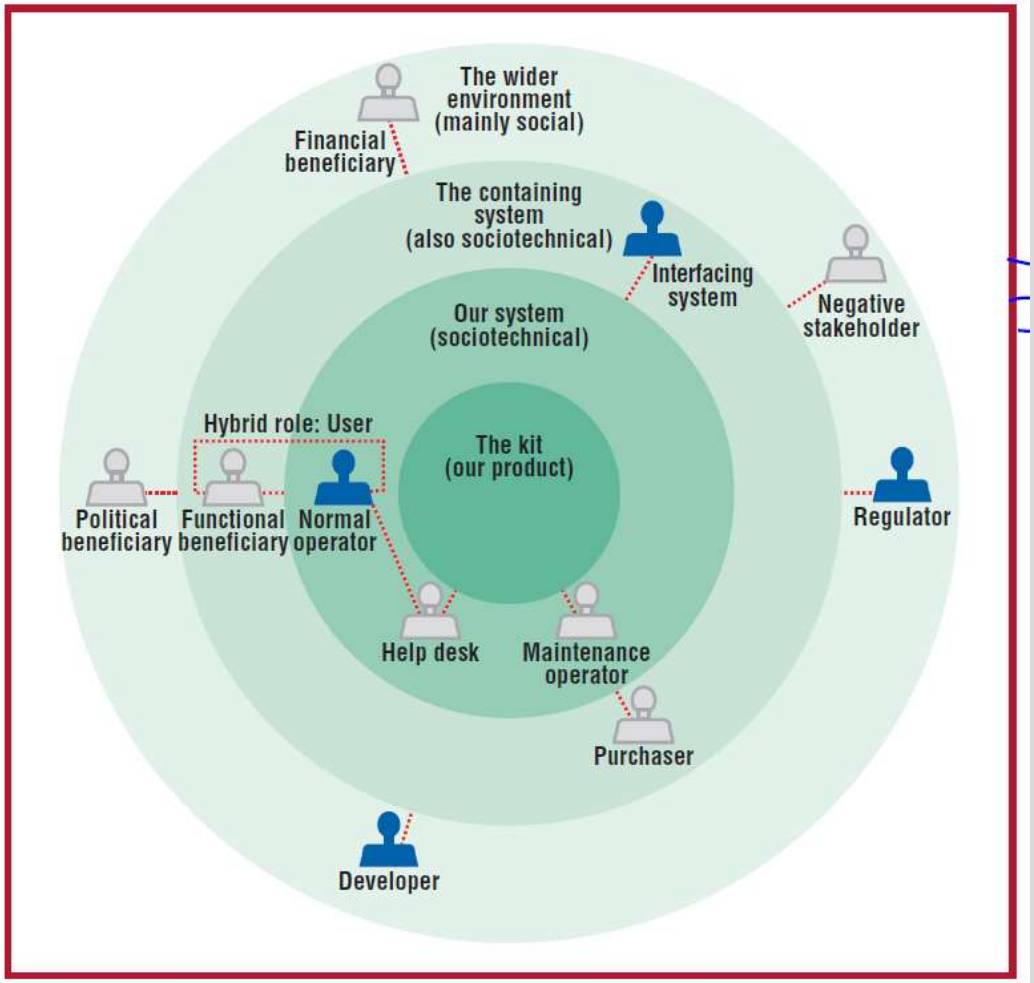


Figure 1. An onion model of stakeholder relationships. Each circle represents a different stakeholder zone.

Understanding Project Sociology by Modeling Stakeholders

Ian Alexander and Suzanne Robertson (IEEE Software 2004)

IEEE Software

ABOUT HISTORY BACK ISSUES WRITE FOR US SUBSCRIBE SE-RADIO BLOG JOIN

Managing Software Platforms and Ecosystems: Call for Papers

FEBRUARY 10, 2017

CALLS FOR PAPERS ♡ 0

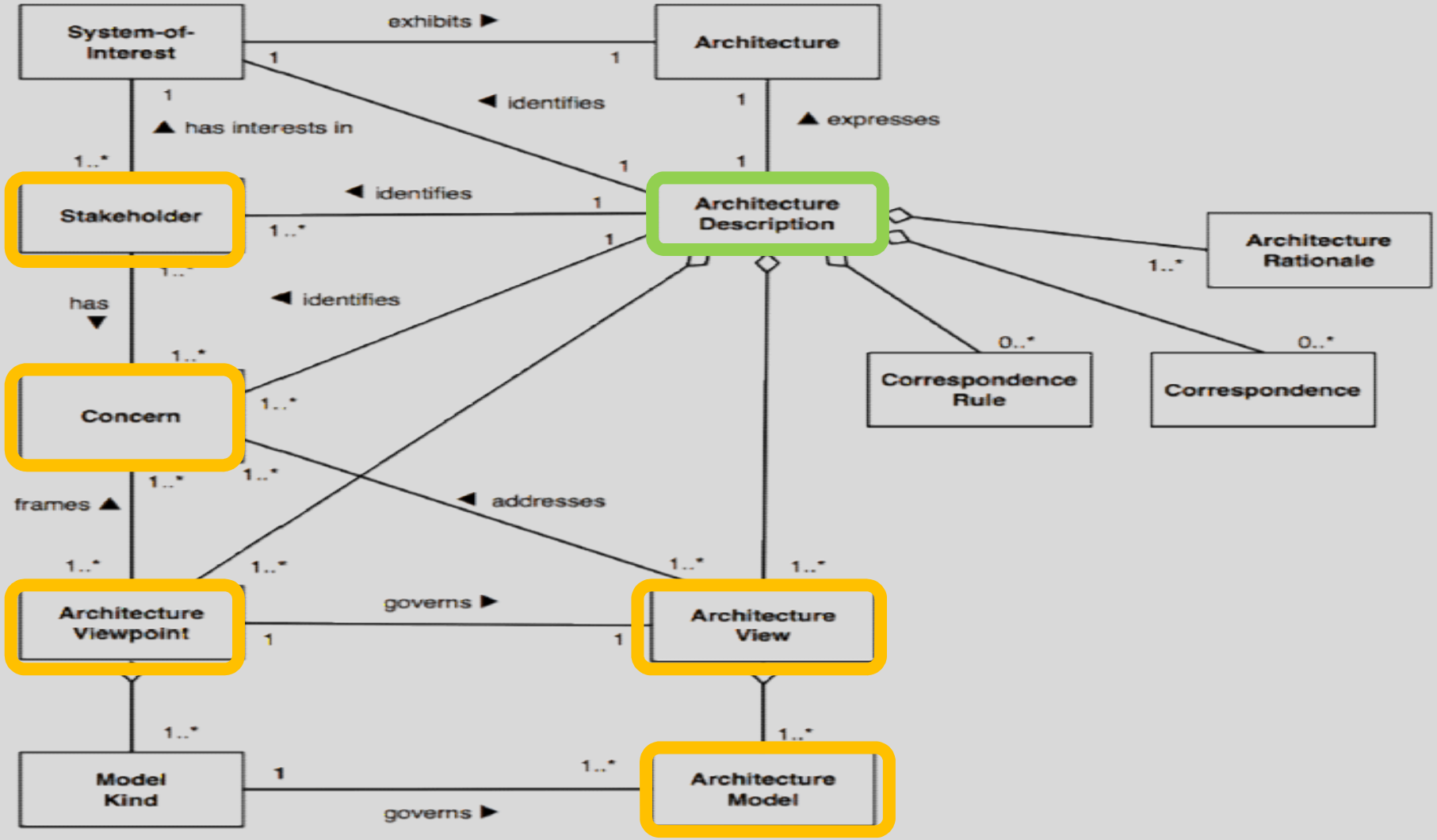
Submission deadline: 1 Dec. 2017

Publication: July/Aug. 2018

A software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. Software ecosystems are pervasive, and software-producing organizations are increasingly realizing that their ecosystem is what makes them and their technologies successful. Decisions to join an ecosystem are made on a strategic level but also typically on an operational level by senior software engineers. These engineers have been coined "kingmakers" because their decisions might lead to long-lasting relationships with the technical platforms they choose to produce technology for.

"One kind of software ecosystem deserves special attention: **developer ecosystems**. The coordination of such ecosystems is challenging…"

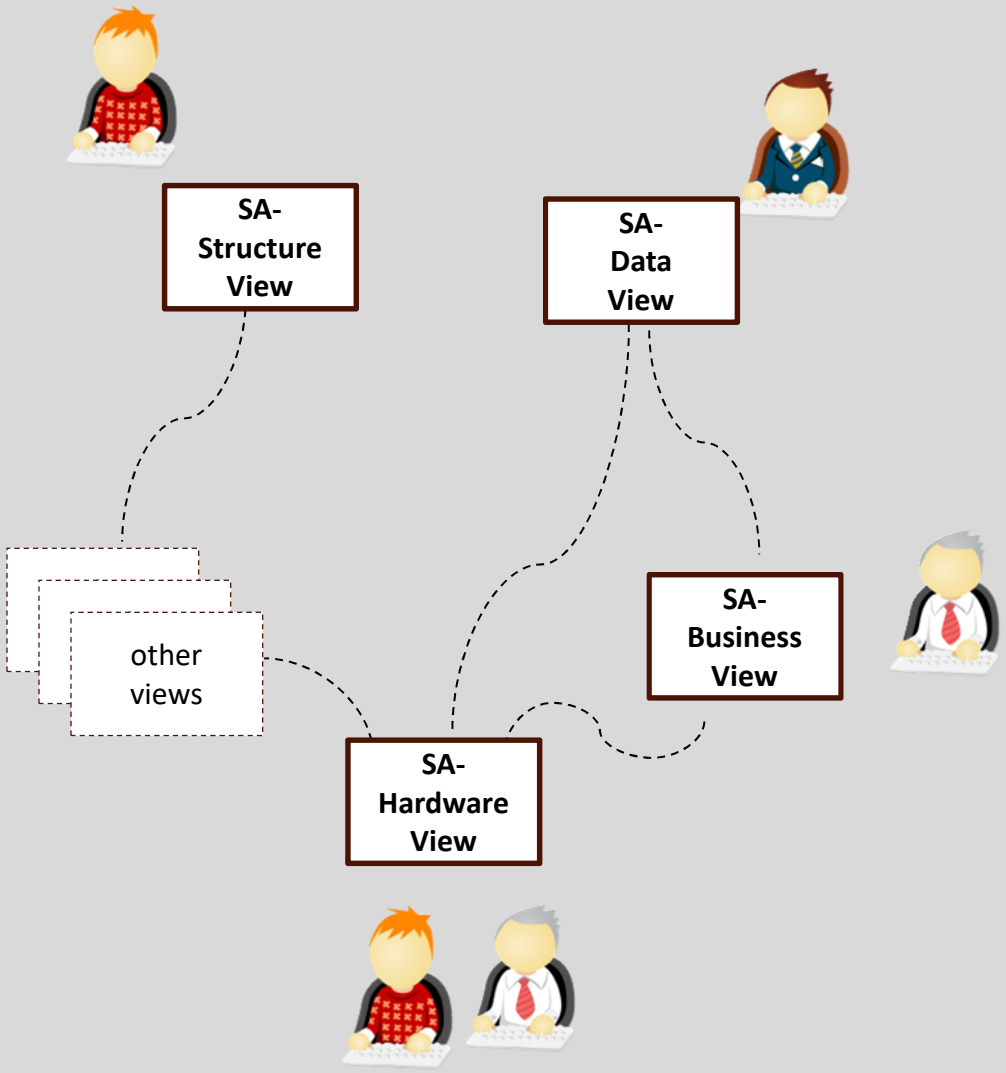
ISO/IEC/IEEE 42010: 2011



ISO/IEC/IEEE 42010 - *International Standard for Systems and Software Engineering Architectural Description*, 2011

So far…

- Multiple SA views
- Produced and managed by different SA stakeholders
- Each with its own architectural concern



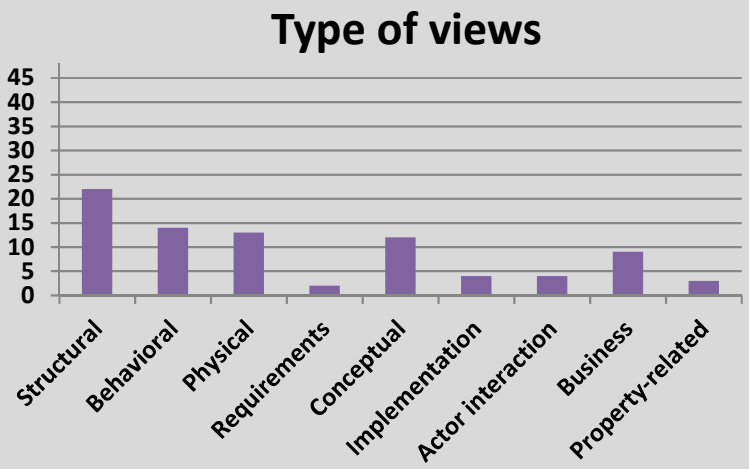
Multi-Views: Who cares?

M-V in practice

Participants: 48 practitioners
From: 15 countries

Use of multiple views for architectural description.

85% use them!



50% of multi view users apply consistency check

What Industry Needs from Architectural Languages: A Survey

Ivano Malavolta, Patricia Lago, *Senior Member, IEEE*, Henry Muccini, Patrizio Pelliccione, and Antony Tang, *Member, IEEE*

Abstract—Many times we are faced with the proliferation of definitions, concepts, languages, and tools in certain (research) topics. But often there is a gap between what is provided by existing technologies and what is needed by their users. The strengths, limitations, and needs of the available technologies can be dubious. The same applies to software architectures, and specifically to languages designed to represent architectural models. Tens of different architectural languages have been introduced by the research and industrial communities in the last two decades. However, it is unclear if they fulfill the user's perceived needs in architectural description. As a way to plan for next generation languages for architectural description, this study analyzes practitioners' perceived strengths, limitations, and needs associated with existing languages for software architecture modeling in industry. We run a survey by interviewing 48 practitioners from 40 different IT companies in 15 countries. Each participant is asked to fill in a questionnaire of 51 questions. By analyzing the data collected through this study, we have concluded that 1) while practitioners are generally satisfied with the design capabilities provided by the languages they use, they are dissatisfied with the architectural language analysis features and their abilities to define extra-functional properties; 2) architectural languages used in practice mostly originate from industrial development instead of from academic research; 3) more formality and better usability are required of an architectural language.

Index Terms—Software architecture, architecture description languages, ADL, survey, empirical study

1 Introduction

1.1 The Problem

In their seminal paper on software architecture [1], Dewayne Perry and Alexander Wolf foresee the 1990s as the decade of software architecture, and justify the need for a (software) architecting discipline on the conjecture that the slow progress in the development of software systems is due to the excessive focus on development and the limited focus on architecting. Twenty years later, we recognize the impact that software architecture has been having on both the academic and industrial worlds. Software architectures are nowadays used for different purposes, including documenting and communicating design decisions and architectural solutions [2], driving analysis techniques (like testing, model and consistency checking, performance analysis [3], [4], [5], [6]), for code generation purposes in model-driven engineering [7], [8], for product line engineering [9], for risks and cost estimation [10], [11], and many more.

- *I. Malavolta, H. Muccini, and P. Pelliccione are with the Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università dell'Aquila, Italy. E-mail: {ivano.malavolta, henry.muccini, patrizio.pelliccione}@univaq.it.*
- *P. Lago is with the Software and Services Group (S4), Department of Computer Science, Faculty of Sciences, VU University Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands. E-mail: p.lago@vu.nl.*
- *A. Tang is with the Centre for Software Analysis and Testing, Faculty of Information and Communication Technologies, Swinburne University of Technology, PO Box 218, Hawthorn VIC 3122, Australia. E-mail: atang@swin.edu.au.*

Manuscript received 5 Apr. 2012; revised 2 Aug. 2012; accepted 22 Oct. 2012; published online 31 Oct. 2012.
Recommended for acceptance by N. Medvidovic.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2012-04-0088.
Digital Object Identifier no. 10.1109/TSE.2012.74.

One principal issue is the proliferation of languages for software architectures (SA) description without a clear understanding of their merits and limitations. Tens of architectural languages (ALs)[1] can be found today, each characterized by slightly different conceptual architectural elements, different syntax, or semantics. They focus on a generic or a specific operational domain; some do not provide design analysis while others support different analysis techniques. As observed in [12], one of the reasons for the accumulation of so many architectural languages is the need to satisfy different *stakeholder concerns*: A language has to adequately capture design decisions judged fundamental by the system's stakeholders. Stakeholder concerns are various, ever evolving, and adapting to changing system requirements. Hence, it is difficult to capture all such concerns with a single, narrowly focused notation. Instead of defining a unique language for software architecture specification, we must accept the existence of domain specific languages for SA modeling, allowing an AL to address specific types of stakeholder concerns. The adoption of UML for modeling architectures (e.g., [13], [14]) did not converge into a standard and uniquely identified notation for SA modeling: A number of UML profiles and extensions have been proposed to enhance the modeling for different concerns, thus further proliferating new architectural languages.

In summary, it is clear that an ideal and general purpose AL is not likely to exist [15], [14], [16], [12]. Rather, architectural languages must be able to focus on "**what is**

1. Hereafter, we use the term architectural language, or AL, to refer to any form of expression used for architecture description. We use the AL term for the sake of clarit, since in the last decades several different definitions of the ADL term have been proposed.

M-V: in academia

Exploring the Temporal Aspects of Software Architecture

Henry Muccini
DISIM, University of L'Aquila, Italy

disim

Keynote at ICSOFT 2016

[ICSOFT2016]

19

Goal

to scrutinize the evolution of the Software Architecture domain over time

17

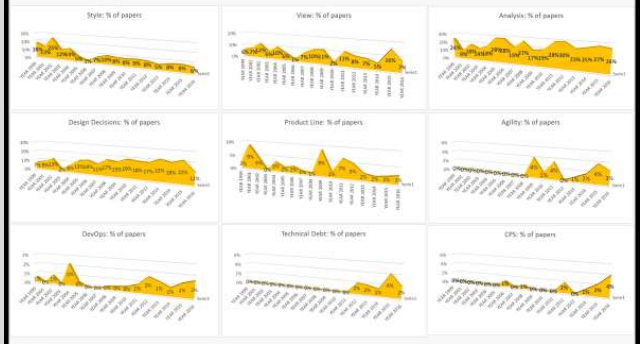
?How to travel over time?

Mixed method used for this study:

1. Topics extraction:
   - Personal knowledge + Seminal papers
2. Data mining
   - From the WICSA, CBSE, ECSA, and QoSA conferences
   - From 1999 to 2016
3. Reasoning on the results

34

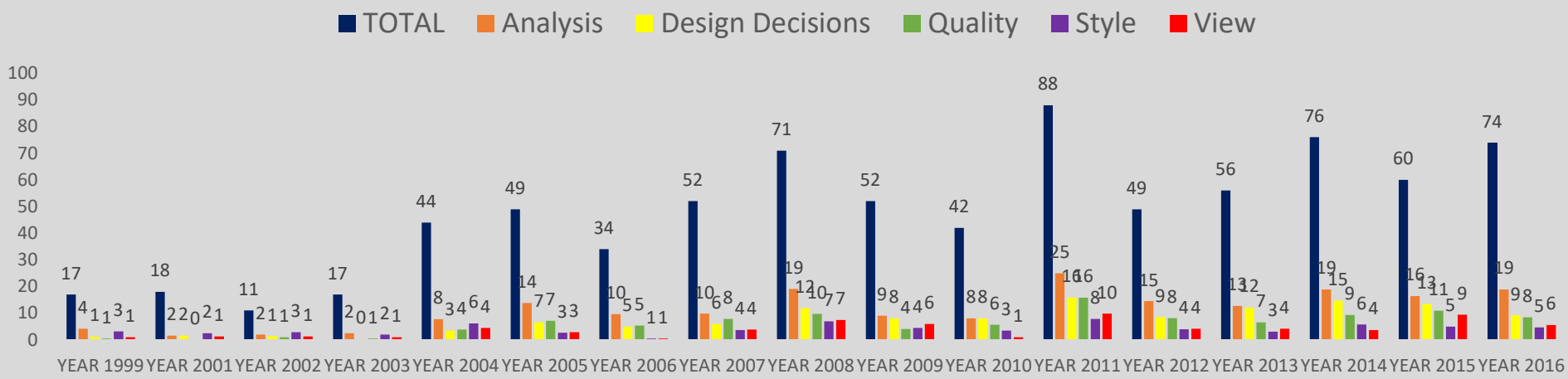
3 Results and Reasoning



M-V: in academia

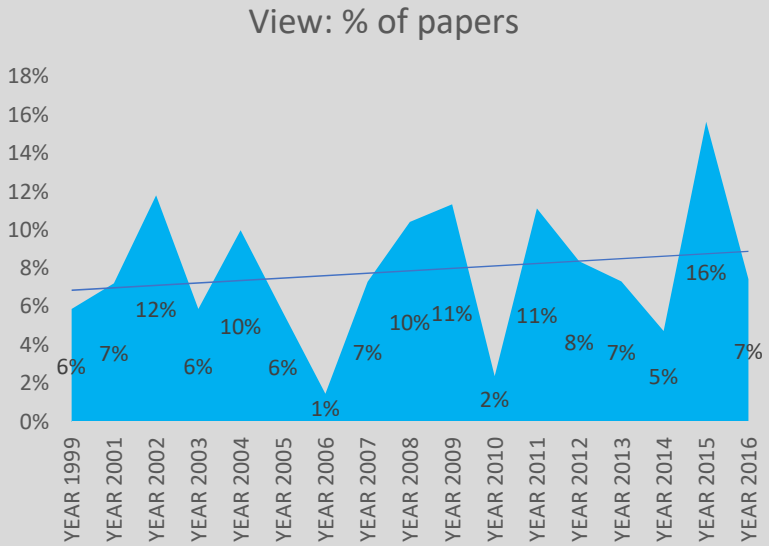
[ICSOFT2016]

Topic Analysis

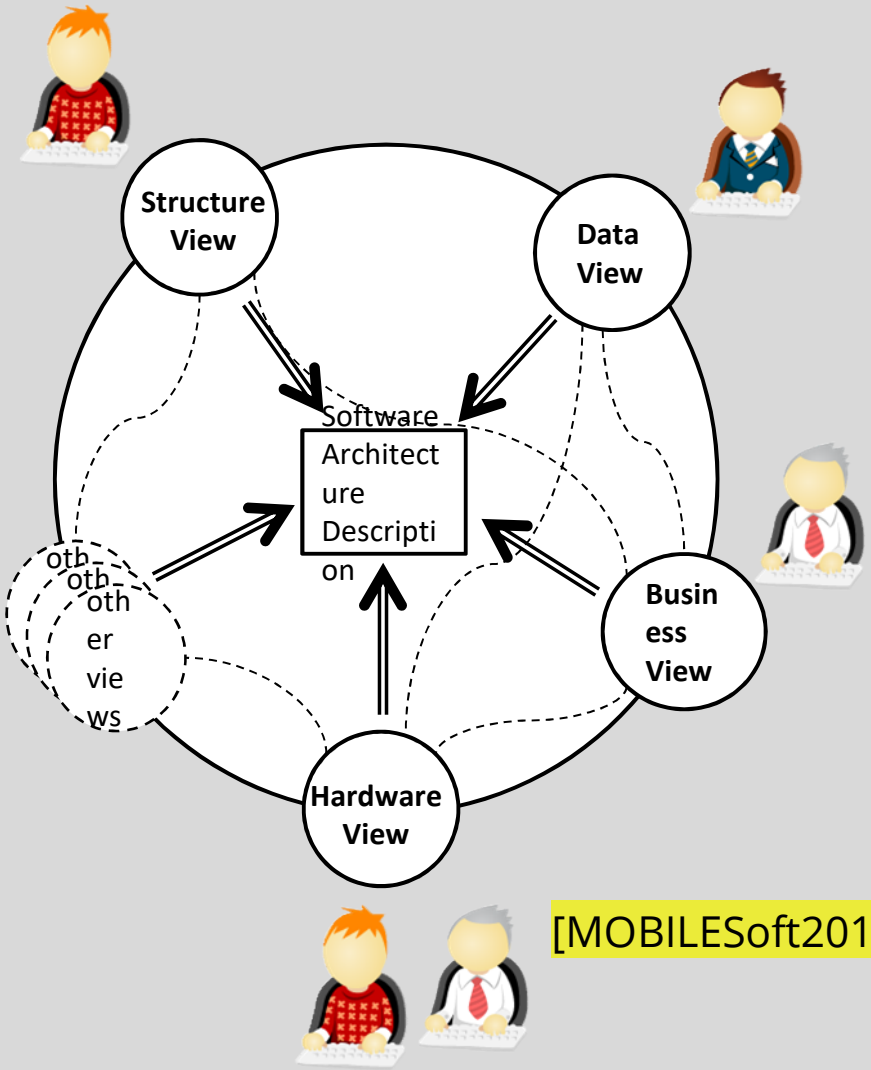


Top five:

1. Analysis (194)
   - Performance (96)
   - Security (27)
   - Consistency (24)
2. Design Decisions (127)
3. Quality (104)
4. Style (68)
5. Views (67)

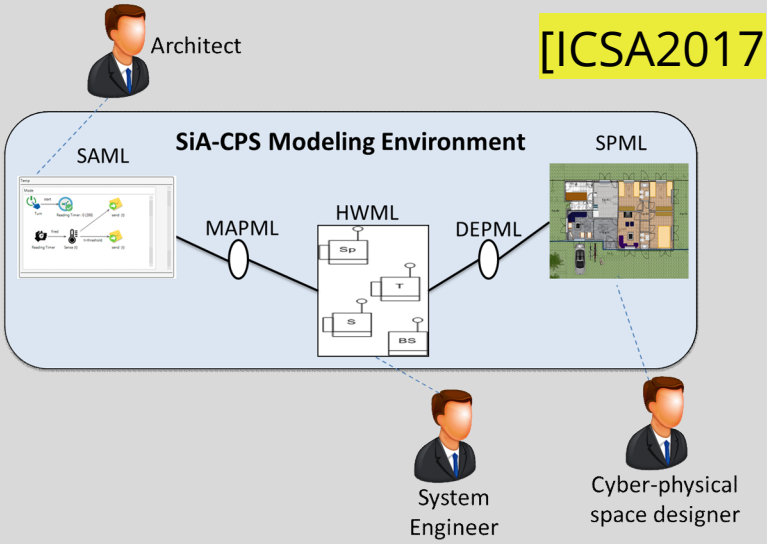


M-V: myself



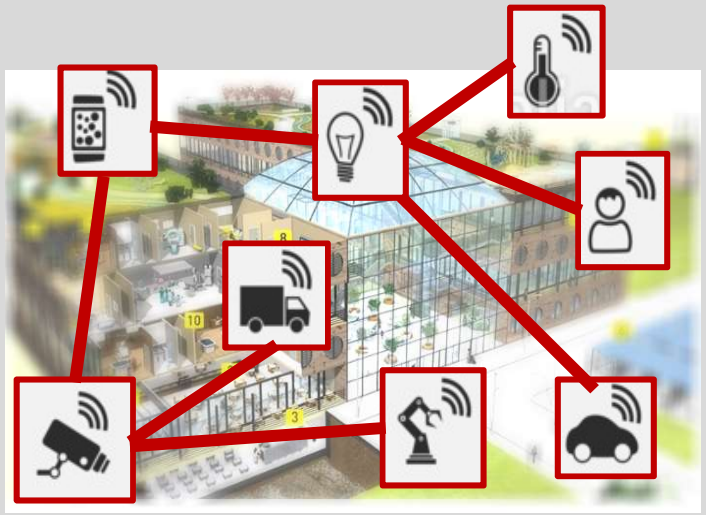
[MOBILESoft2014]

Engineering Mobile Applications



[ICSA2017

Engineering Situational-Aware CPS




Multi-view management of stakeholders' concerns: from architectures back to requirements
@RESG March 2017
Henry Muccini
DISIM Department
University of L' Aquila, L'Aquila, Italy
www.henrymuccini.com - henry.muccini@univaq.it - @muccinihenry
disim

But, what about Requirements Engineering?

Viewpoint-oriented RE

Annals of Software Engineering 3 (1997) 101–130 101

Viewpoints: principles, problems and a practical approach to requirements engineering

Ian Sommerville and Pete Sawyer

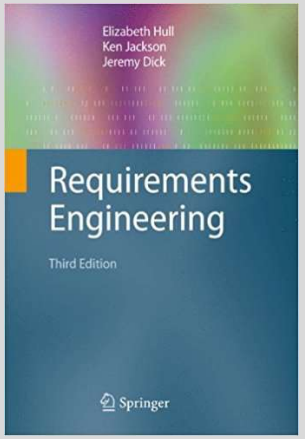
Computing Department, Lancaster University, Lancaster LA1 4YR, UK

2. Viewpoints

A viewpoint-based approach to requirements engineering recognises that all information about the system requirements cannot be discovered by considering the system from a single perspective. Rather, we need to collect and organise requirements from a number of different *viewpoints*. A viewpoint is an encapsulation of partial information about a system's requirements. Information from different viewpoints must be integrated to form the final system specification.

The principal arguments in favour of a viewpoint-based approach to requirements engineering are:

1. Systems usage is heterogeneous – there is no such thing a Viewpoints may organise system requirements from different end-user and other system stakeholders.
2. Different types of information are needed to specify systems i tion about the application domain, information about the syst and engineering information about the system's development. be used to collect and classify this information.
3. Viewpoints may be used as a means of structuring the proces elicitation.
4. Viewpoints may be used to encapsulate different models of t which provides some specification information.
5. Viewpoints may be used to structure the requirements descri conflicts between different requirements.



VORD
Preview
CORE
SADT
…

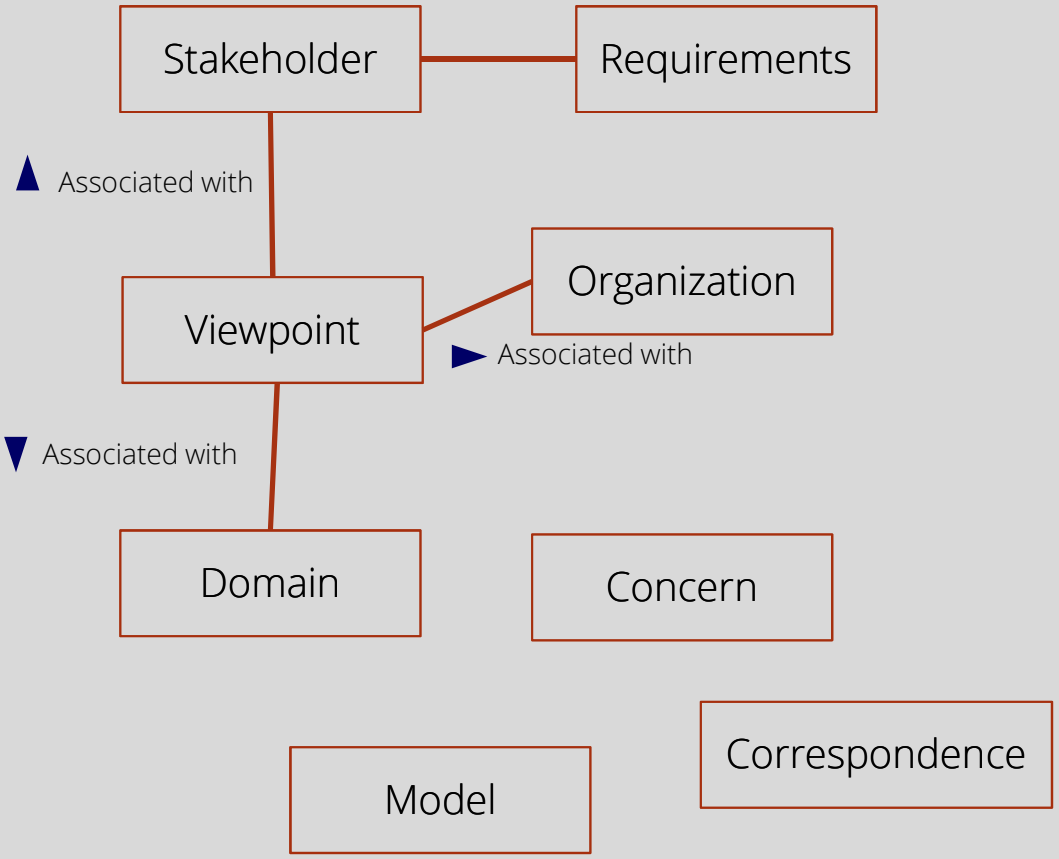
Requirements Engineering: A Roadmap

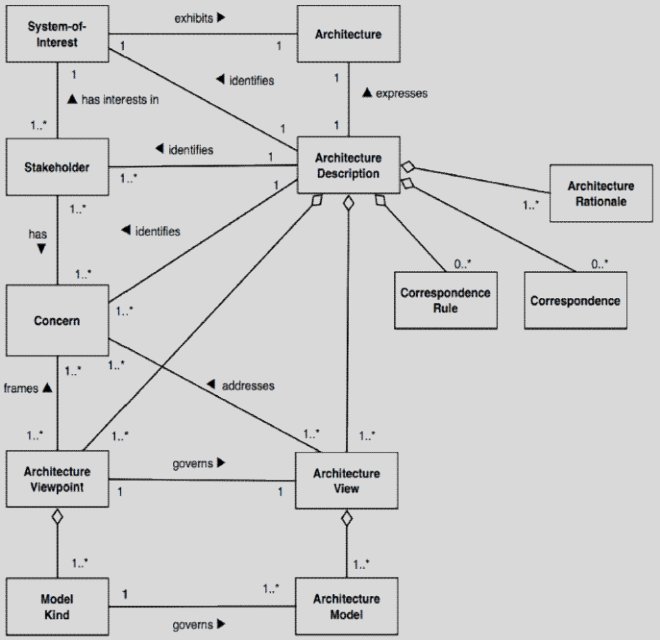
Bashar Nuseibeh & Steve Easterbrook

Key Research Pointers

- Better modelling and analysis of problem domains, as opposed to the behaviour of software.
- Development of richer models for capturing and analysing non-functional requirements.
- Bridging the gap between requirements elicitation approaches based on contextual enquiry and more formal specification and analysis techniques.
- Better understanding of the impact of software architectural choices on the prioritisation and evolution of requirements.
- Reuse of requirements models to facilitate the development of system families and the selection of COTS.
- Multi-disciplinary training for requirements practitioners.

A «partial» RE model for Viewpoints



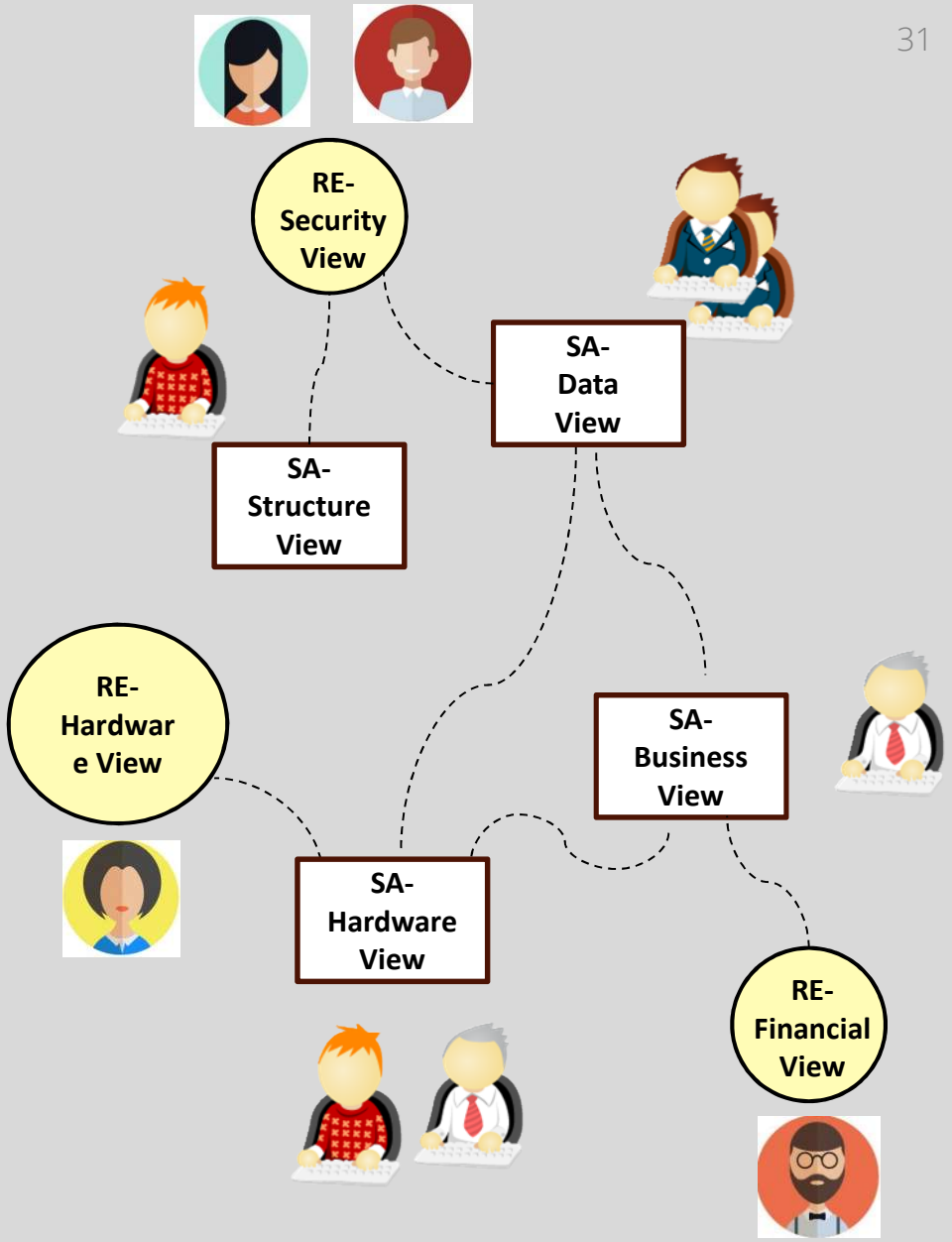


42010 Standard


Multi-view management of stakeholders' concerns: from architectures back to requirements
@RESG March 2017
Henry Muccini
DISIM Department
University of L' Aquila, L'Aquila, Italy
www.henrymuccini.com - henry.muccini@univaq.it - @muccinihenry
disim

So far…

- Multiple views and artifacts
- With different links
- Produced and managed by different stakeholders
- Each with its own concern



Many advantages, but challenges

1. To organise system requirements from different classes of system end-user and other system stakeholders.
2. to collect and classify info about the application domain, the system's environment, the system's development.
3. To structure the process of requirements elicitation.
4. To encapsulate different models of the system each of which provides some specification information.
5. To structure the requirements description and expose conflicts between different requirements

[Sommerville+97]

Concrete industrial case

Change Management in multi-view multi-stakeholder systems

Concrete industrial case

Context:

digital financial advice for mortgage applications company

Goal:
mortgage advice
100% digital

Challenge:

requirements change

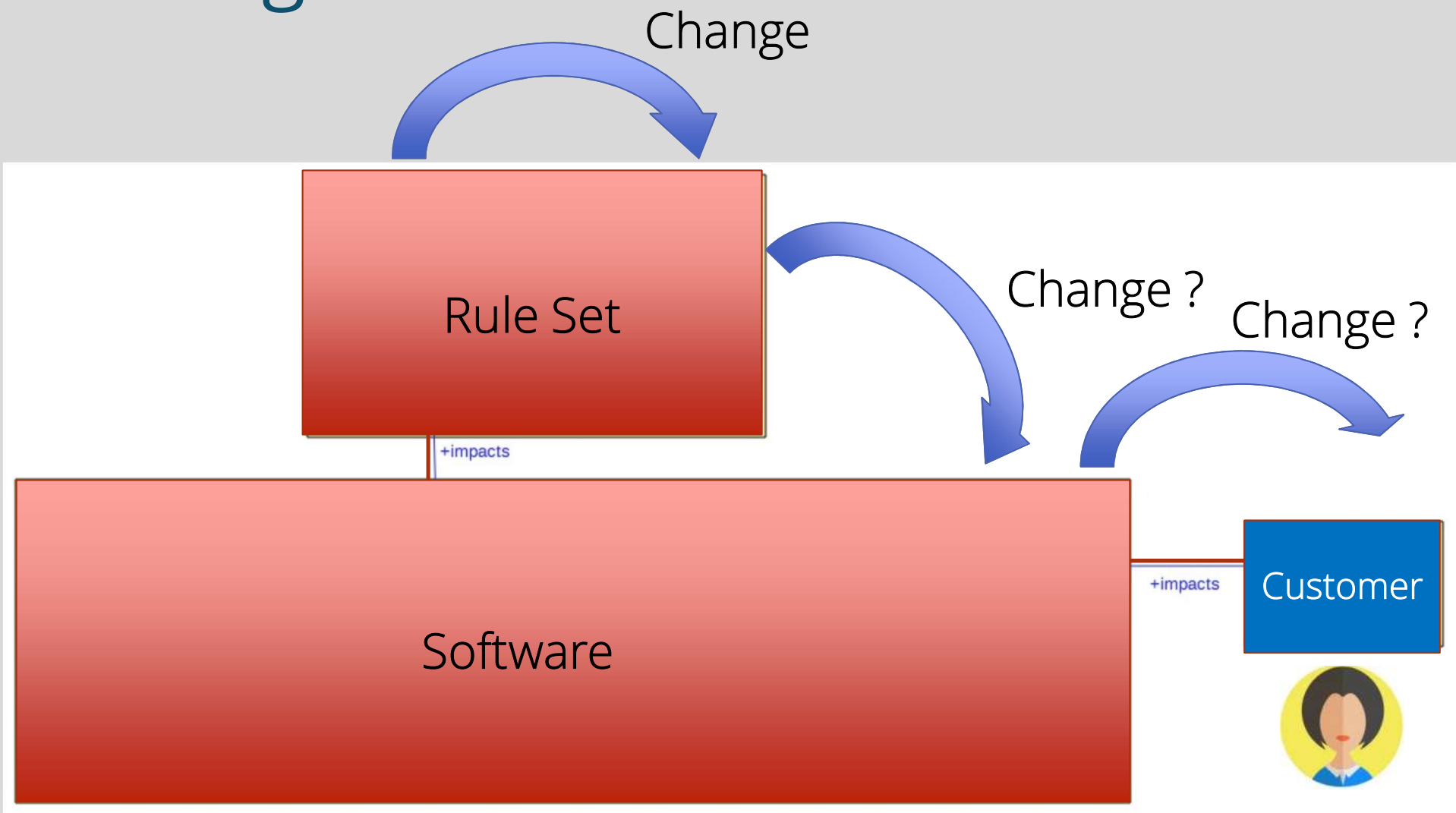
@Knab Advies & Bemiddeling
(formerly eyeOpen.nl)

Issues:

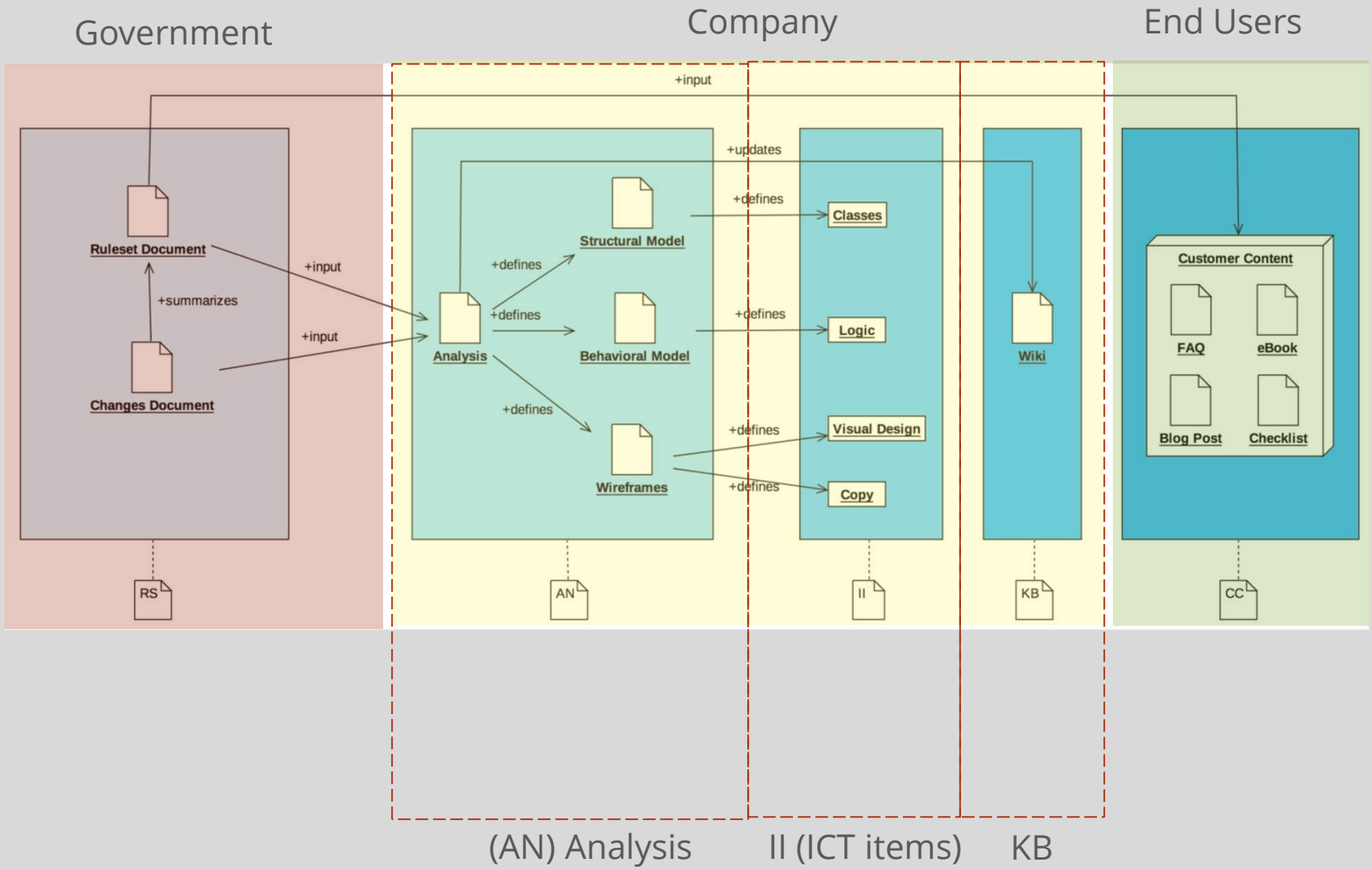
i1) How to make stakeholders aware of the changes?

i2) Impact change measurement

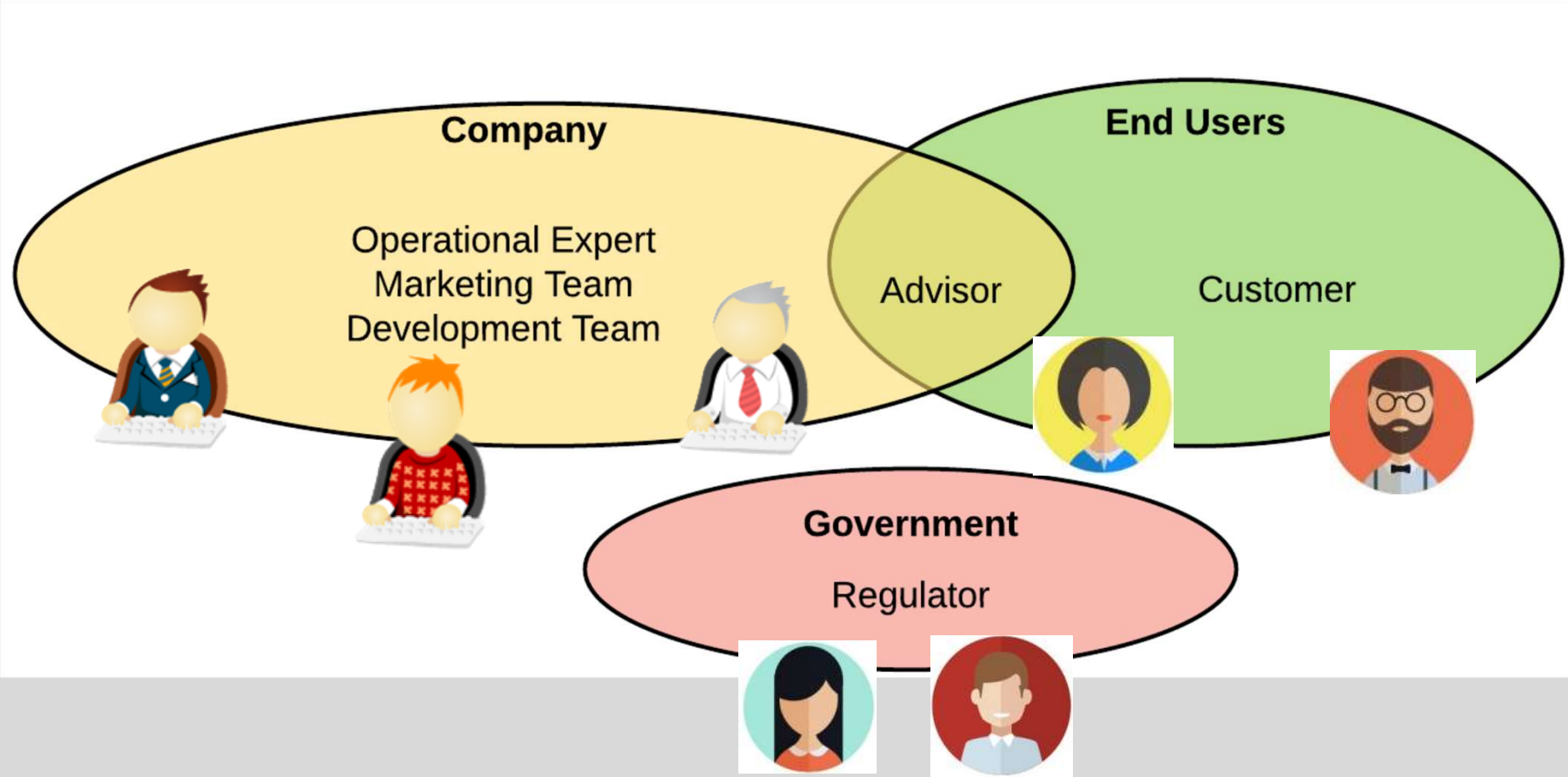
Change links



Artifacts


Government
Company
End Users
+input
Ruleset Document
+summarizes
Changes Document
+input
+input
+updates
Structural Model
+defines
Classes
+defines
Analysis
+defines
Behavioral Model
+defines
Logic
+defines
Wiki
Customer Content
FAQ
eBook
Blog Post
Checklist
+defines
Visual Design
Wireframes
+defines
Copy
RS
AN
II
KB
CC
(AN) Analysis
II (ICT items)
KB

Eco-system of Stakeholders



Multi-stakeholders on multi artifacst

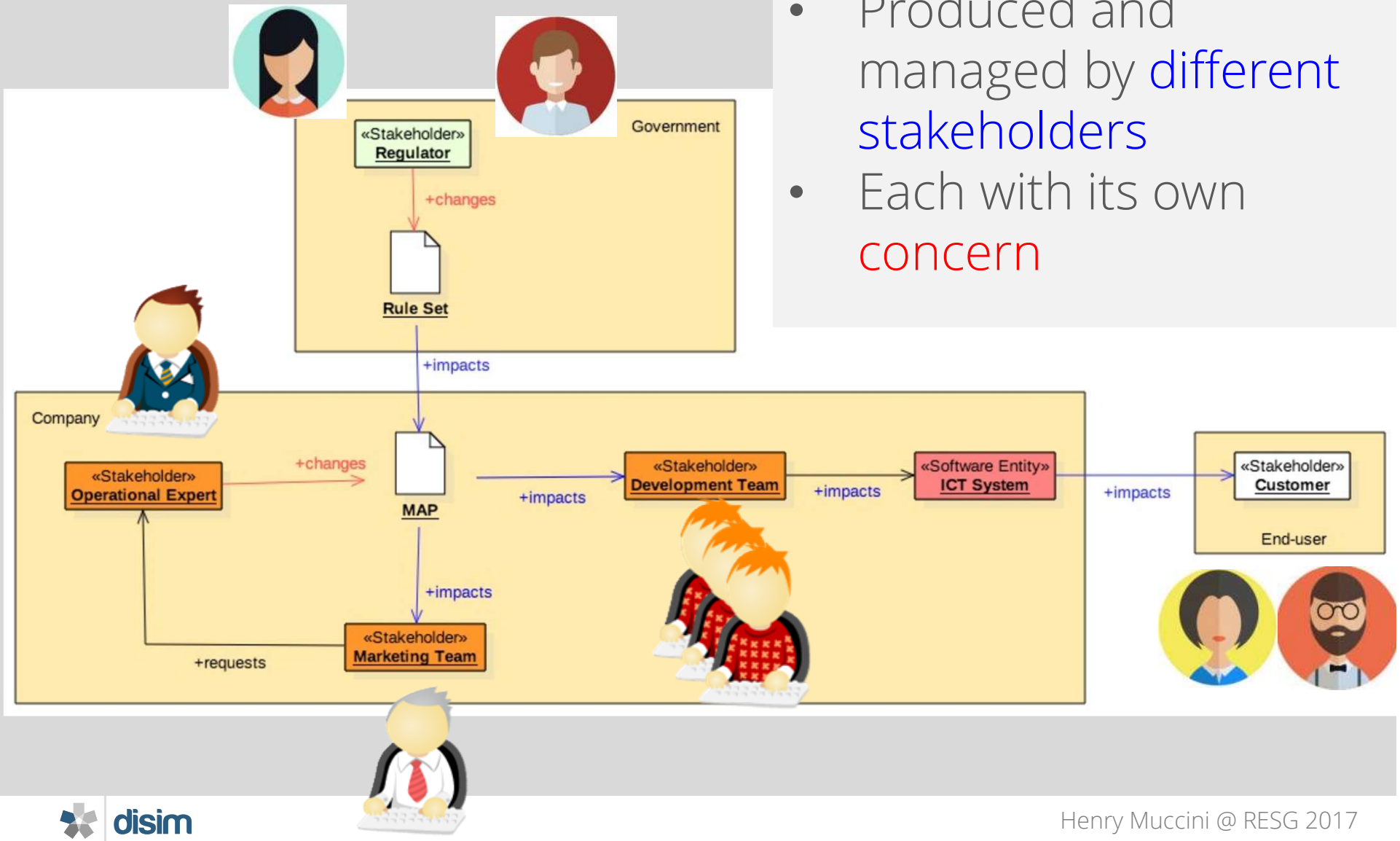
	R	OE	DT	MT	A	C
RS						
AN						
II						
KB						
CC						

Stakeholders

Classes of Artifacts

Change links

- Multiple views and artifacts
- With different links
- Produced and managed by different stakeholders
- Each with its own concern



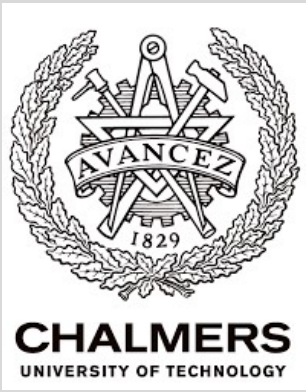
Issues

- Requirement dependency
- Communicating changes to stakeholders
- Impact change measurement (both globally and per stakeholder)

Goals

- Improve analysis process
- Improve change awareness among stakeholders

Our ongoing Solution



knab

Capra Traceability Framework

It is a configurable and extendable traceability management tool based on EMF

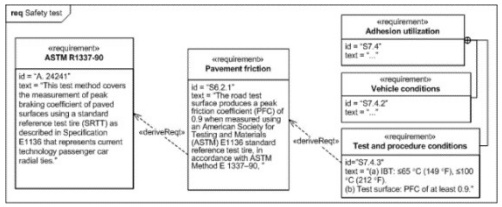
How it works:

- Based on EMF, it stores the traceability model as an EMF model.
- provides a number of artifact handlers that allow using, as sources and targets for traceability links.
- traceability metamodel is not fixed and can be defined by the user
- it relies on the Eclipse Extension mechanism and provides an extension point for artifacts types to be supported

Capra Traceability
Tool Integration

ITEA 3

Requirements

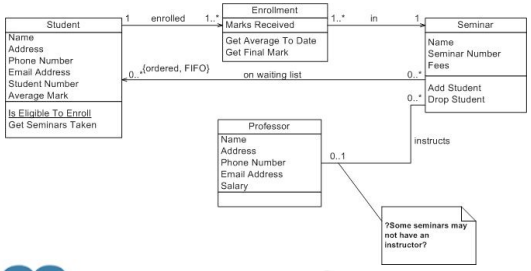


SysML™ Office

Tests

Hudson Jenkins

Design



Mechatronic UML APP4MC

Capella Papyrus

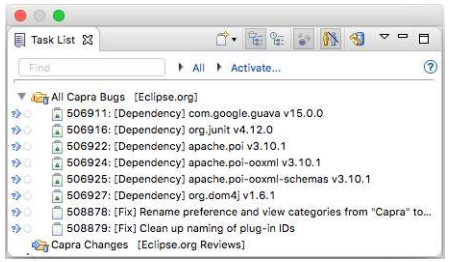
4public AMALTHEA

Source Code

```
public void setLocaleToCookie(HttpServletResponse request, String lokale) {
    cookieGenerator.addCookie(request, lokale);
}

public String getLocaleFromCookie(HttpServletRequest request) {
    Cookie cookie = WebUtils.getCookie(request, REQUEST_ATTRIBUTE_LANGUAGE);
    return cookie != null ? cookie.getValue() : null;
}

@Override
public void doInit() throws ServletException {
    cookieGenerator.setCookieName(REQUEST_ATTRIBUTE_LANGUAGE);
}
```

C++ Java™ php

Tickets and Tasks



Bugzilla JIRA

GitHub

Capra Traceability
Technical Overview

Capra

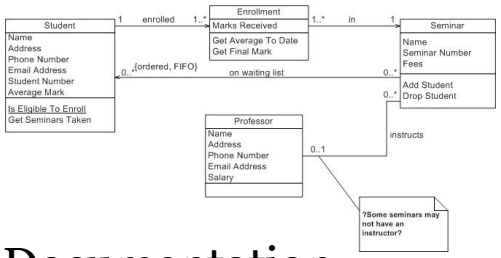
ITEA 3

Requirements
DOORS, ReqIF, MS Excel

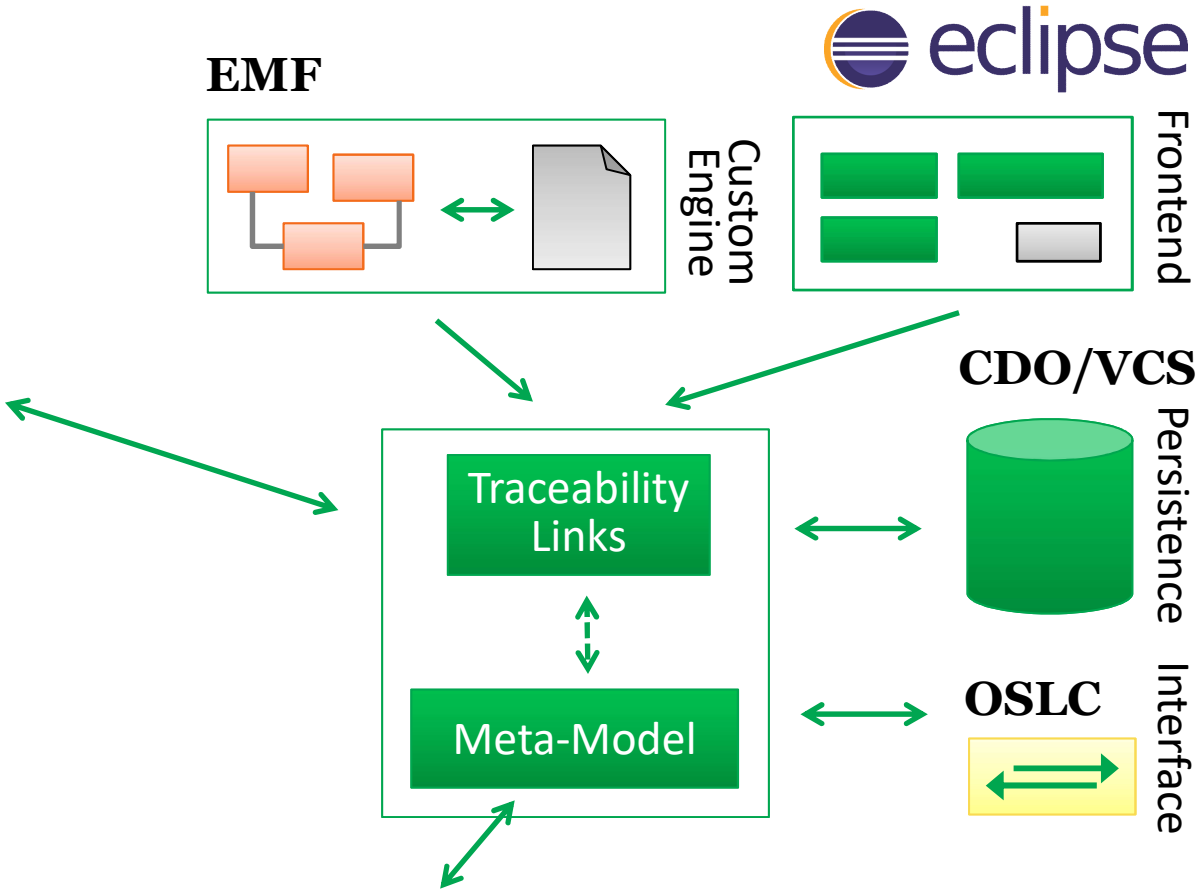
Source Code

```
public void setLocaleToCookie(HttpServletResponse request, String lokale) {
    cookieGenerator.addCookie(request, lokale);
}

public String getLocaleFromCookie(HttpServletRequest request) {
    Cookie cookie = WebUtils.getCookie(request, REQUEST_ATTRIBUTE_LANGUAGE);
    return cookie != null ? cookie.getValue() : null;
}

@Override
public void doInit() throws ServletException {
    cookieGenerator.setCookieName(REQUEST_ATTRIBUTE_LANGUAGE);
}
```

Design

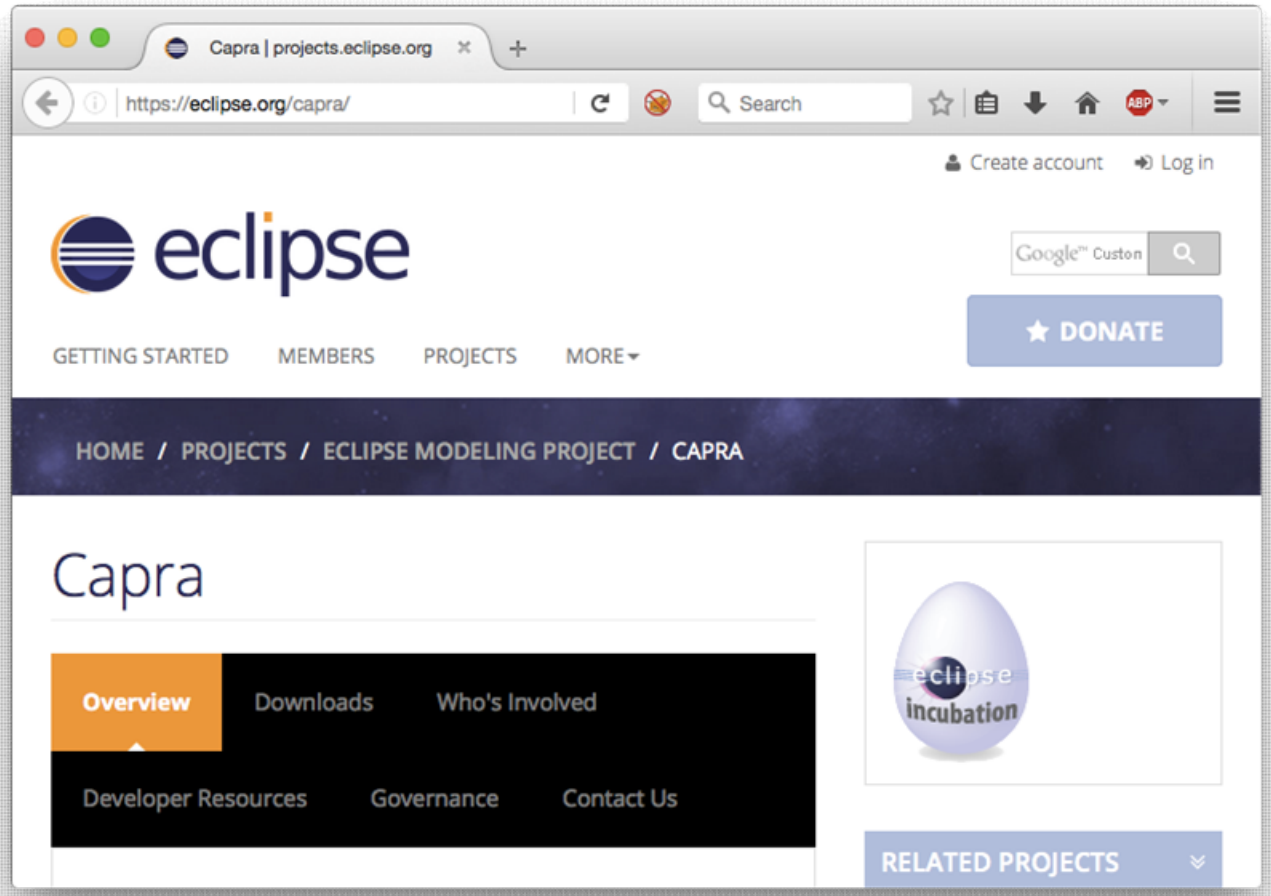


Documentation

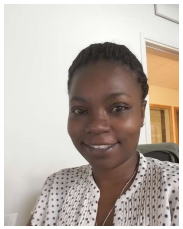


4public AMALTHEA

Capra Traceability
Source Code and Community @ Eclipse



https://eclipse.org/capra



Salome Maro
salome.maro@gu.se

Jan-Philipp Steghöfer
jan-philipp.steghofer@gu.se

Why are we using Capra?

[CapraRE2016]

1) Supports the creation of traceability links to arbitrary artifacts.
   - Other tools are only req-specific (e.g., DOORS)

2) Supports the definition of custom traceability link types for projects.

3) Supports the visualization of artifacts connected by traceability links through matrix or graph view.

Our solution with Capra

- Integration:
  - 1. Analysis of current development and change process
  - 2. Definition of goals (suggested outcomes)
  - 3. Definition of Traceability Analysis outcome and identifcation of related decision-makers that will use it
- Instantiation:
  - 1. Artifacts elicitation and classification
  - 2. Collection of existing Artifacts
  - 3. Definition of Case Scenarios and related examples
  - 4. Traceability metamodel definition (type of links, based on point 2)
  - 5. Creation of a project with Artifacts in Capra
  - 6. Traceability Links creation in Capra

Research method

- Elicitation and definition of artifacts [DONE]
- Creation of traceability metamodel [DONE]
- Creation of traceability links among artifacts [ONGOING]
- Measurement of current analysis method, without Traceability Management [TO DO]
- Measurement of new analysis method, based on Traceability Management [TO DO]
- Comparison of two methods
  - Quantitative analysis (time to spec, time to develop)
  - Qualitative analysis (spec detail, change awareness among stakeholders)

knab

Challenges

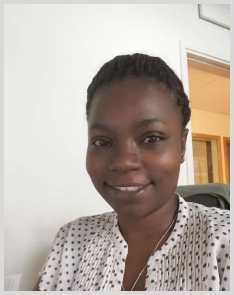
- Addition of new elements in each artifact and related maintenance of traceability links (possible missing updates)
- Measurement of current analysis process (lack of metrics)
- Traceability links granularity (level of the relationship between artifacts)

knab

Capra in practice in Knab A&B

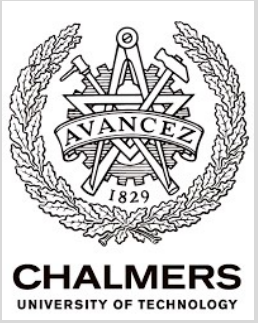
Paolo Bozzelli

Henry Muccini



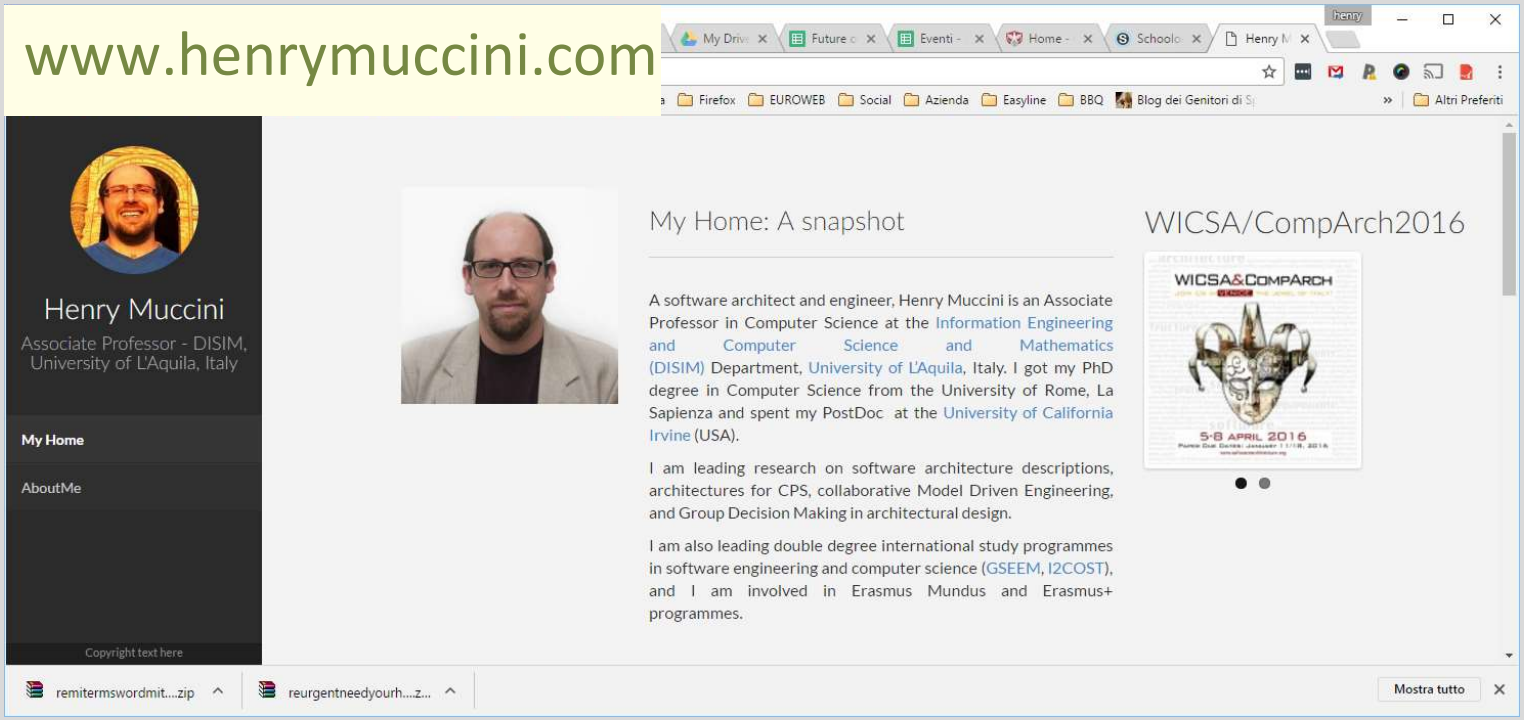
Salome Maro
Jan-Philipp Steghöfer

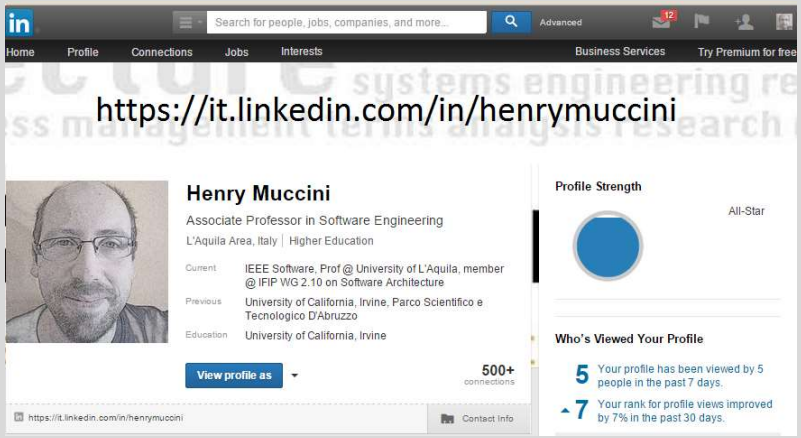
knab



Keep in touch…

www.henrymuccini.com





disim

References

[MLMPT13] Ivano Malavolta, Patricia Lago, Henry Muccini, Patrizio Pelliccione, Antony Tang: What Industry Needs from Architectural Languages: A Survey. IEEE Trans. Software Eng. 39(6): 869-891 (2013), http://doi.ieeecomputersociety.org/10.1109/TSE.2012.74

[ICSOFT2016] Henry Muccini. Exploring the Temporal Aspects of Software Architecture. Keynote talk @ICSOFT 2016. Slides available at: http://www.slideshare.net/henry.muccini/

[MOBILESoft2014] Mirco Franzago, Henry Muccini, Ivano Malavolta: Towards a collaborative framework for the design and development of data-intensive mobile applications. MOBILESoft 2014: 58-61

[ICSA2017] Henry Muccini and Mohammad Sharaf. CAPS: Architecture Description of Situational Aware Cyber Physical Systems. In IEEE Proc. Int. Conference on Software Architecture, ICSA 2017.

[CapraRE2016] Salome Maro, Jan-Philipp Steghöfer: Capra: A Configurable and Extendable Traceability Management Tool. RE 2016: 407-408

[Sommerville+97] Ian Sommerville and Pete Sawyer. Viewpoints: principles, problems and a practical approach to requirements engineering. Annals of Software Engineering 3 (1997) 101–130