Challenges In Building Applications And Services For Smart Devices

Moderator
Krishna Kavi, University of North Texas, USA

Panelists
Michael Gebhart, iteratec GmbH, Germany
Mira Kajko-Mattsson, KTH Royal Institute of Technology, Sweden
Sylvain Vauttier, Ecole Nationale Supérieure des Mines d'Alès, France
Faouzi Moussa, CRISTAL, Tunisia
Petre Dini, IARIA, USA

Challenges In Building Applications And Services For Smart Devices

A General Framework for Discussion

a). Availability of Tools (and ease of use)

- Tools for Development
- Tools for Testing
- Tools for Performance/Energy evaluation
- Security/Reliability/Safety
- Designing for Ergonomics and Usability

b). Security Related Issues

- Testing for security
- Integrating 3rd party codes
- Updating against new security vulnerabilities

c). Creating A Market

Challenges In Building Applications And Services For Smart Devices

A General Framework for Discussion

Other Issues

- Interoperability
- Programming languages and frameworks
- Ever changing hardware capabilities

And Smart Devices span beyond Smart phones

- Wearable and Implantable devices
- IoT

Challenges In Building Applications And Services For Smart Devices

My two cents worth (based on discussions with students and colleagues)

Per developing smart phone aps

- Keeping up with the ever changing capabilities of devices and updating apps
  - Control over hardware capabilities (e.g., power management)
- Cross platform development tools are becoming available, but often cumbersome
- Interoperability is also an issue, if not in terms of functionality,
  - but in terms of performance
- Philosophy behind different manufacturers (Android vs IOS)
- Need more standardization

Challenges In Building Applications And Services For Smart Devices

My two cents worth (based on discussions with students and colleagues)

Per developing smart phone aps

Testing for functionality is reasonably addressed

but not for security or performance/energy management

Xcode (IOS) is better for integrating with 3rd party libraries

Better support with Objective C (than Java)

Android is better with Java

Android has more relaxed attitude and thus may not be as secure

Challenges In Building Applications And Services For Smart Devices

My two cents worth (based on discussions with students and colleagues)

”Making applications is easy, but securing them is very difficult” Immunio.inc

- Create layers of protection around applications
- Control access
- Log activity (and monitor)
- Sanitize inputs
- Report vulnerabilities appropriately
- Assess risks associated with third party and legacy applications

Challenges In Building Applications And Services For Smart Devices

Michael Gebhart: Choosing the right paradigm: Native vs. Hybrid vs. Web apps.
Is it necessary to write native apps? Or is it sufficient to use web technologies and frameworks?

Mira Kajko-Mattsson: Organizational, educational challenges: Method and competency perspective

Sylvain Vauttier: User empowerment for building smart environments with IoT technologies: Privacy, Ethics and Interoperability

Faouzi Moussa: Designing context-aware User Interfaces while integrating ergonomic/usability rules

Petre Dini: Challenges in developing apps for wearable/implantable devices: Computation vs Sensing processing requirement

A three-level versioning model for component-based software architectures

Abderrahman MOKNI*, Marianne HUCHARD, Christelle URTADO* and Sylvain VAUTTIER***

***Ecole des Mines d'Alès, Nîmes, France**

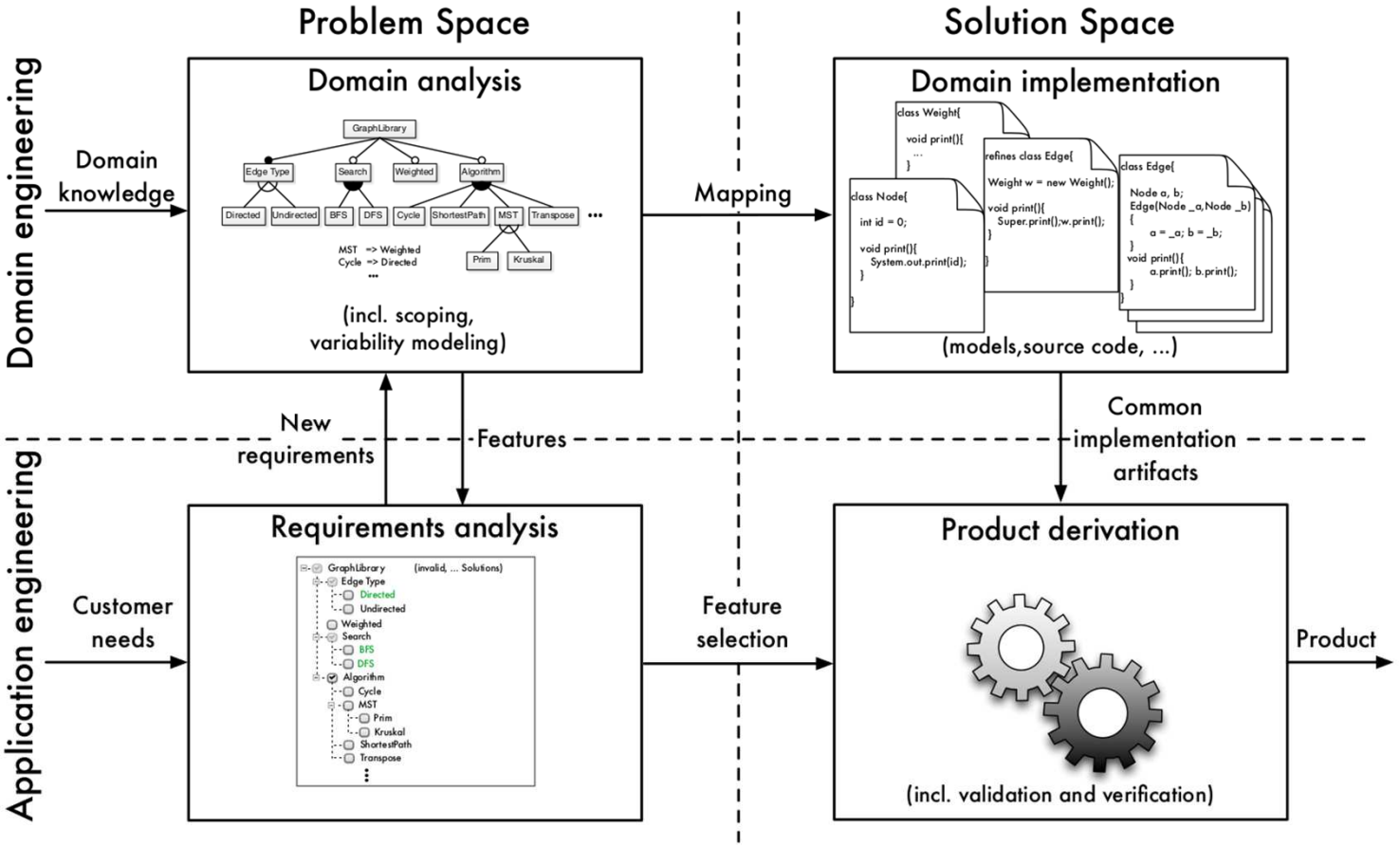
****LIRMM, Montpellier, France**

Outline

- Context: Dedal, an architecture description language for reuse intensive development processes
- Managing the evolution of three-leveled architecture descriptions in Dedal
- Three-level versioning model for tracing the evolutions of architecture descriptions in Dedal
- Conclusion and perspectives

Context : reuse-intensive development processes

- **Software product-line engineering**



Context : reuse intensive development processes

- **Component-based software engineering**

Functional requirements

Component repositories

Architecture Design

Architecture Specification

Architecture Implementation

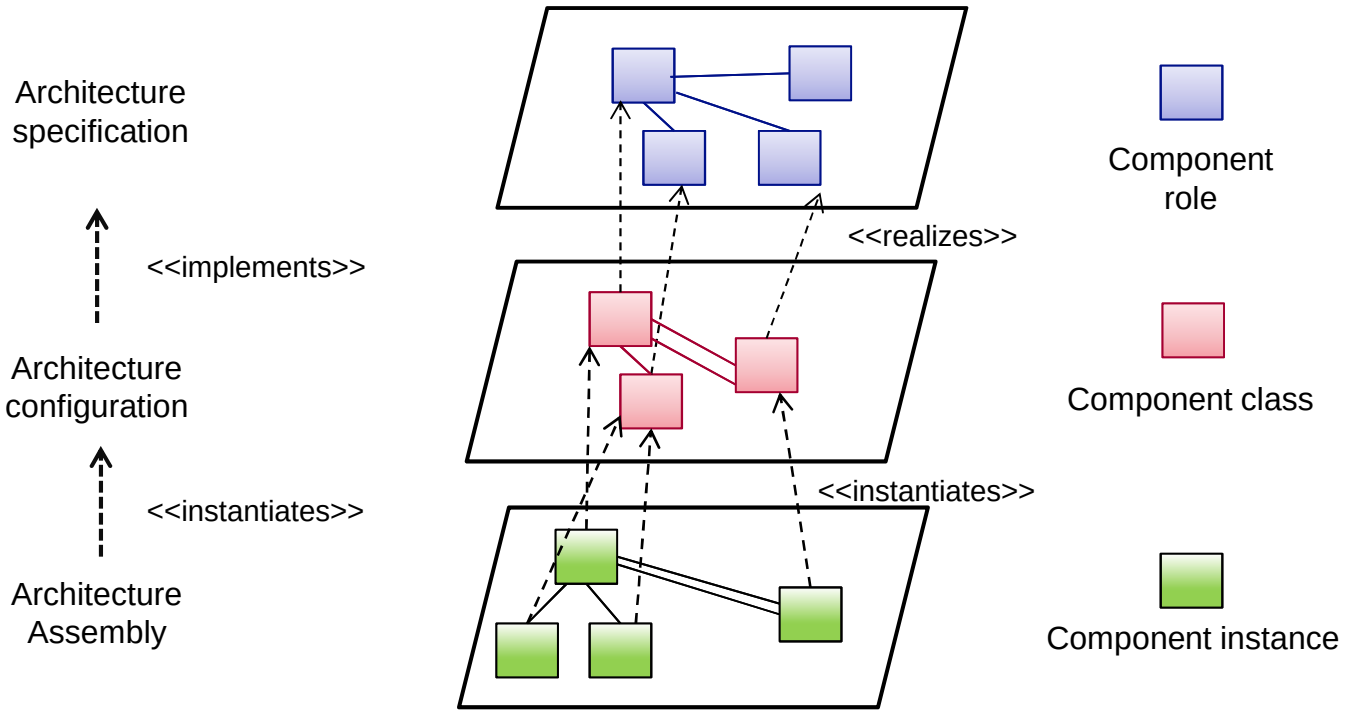
Architecture Configuration

Architecture Deployment

Architecture Assembly

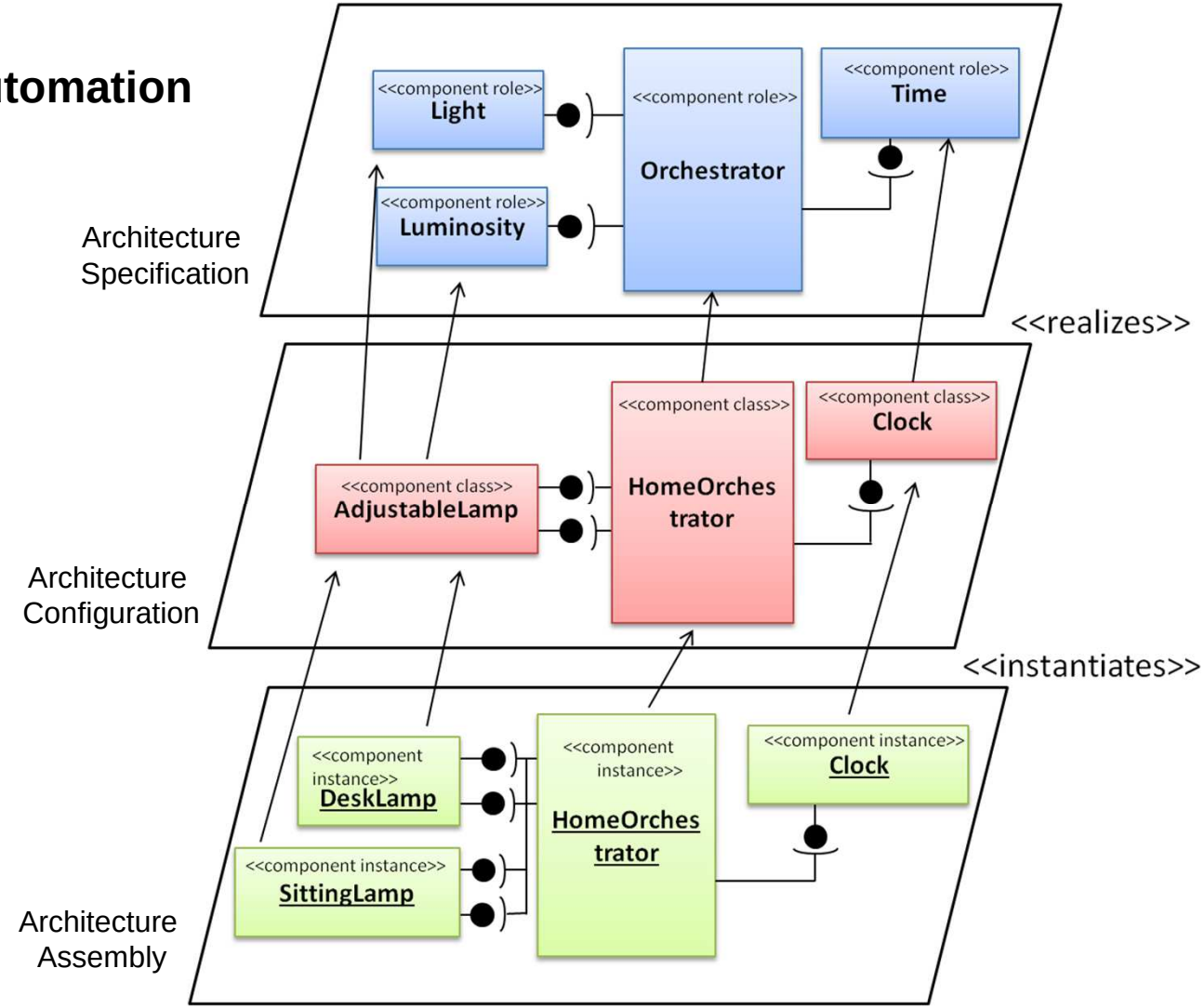
Context : reuse intensive development processes

- **Dedal: a three-level architecture description language**
  - capture architectural decisions
  - foster architecture description reuse



Context : reuse intensive development processes

- **Example: Home Automation System**

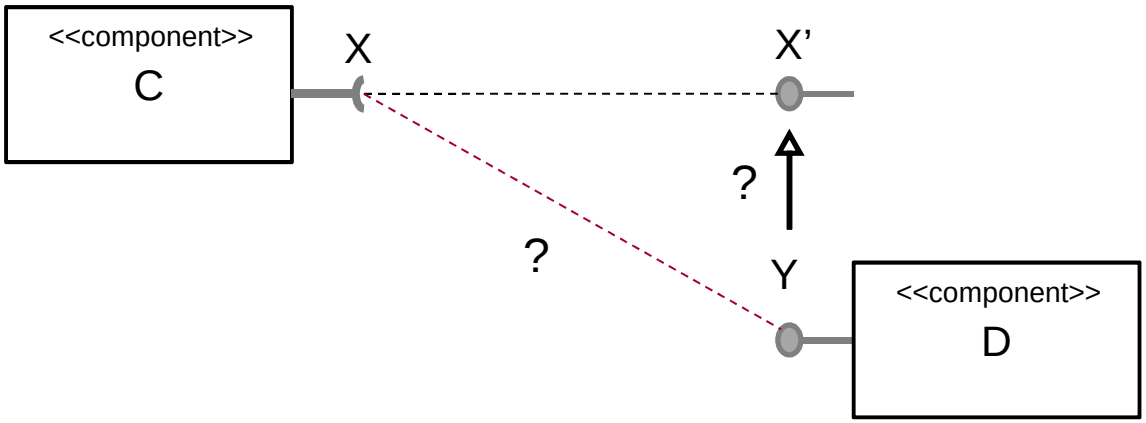


Evolution of three-leveled architecture models in Dedal

- **Architecture maintenance**
  - prevent obsolescence
- **Derive new architectures from existing ones**
  - agile/incremental development
- **Problematics: inconsistencies, loss of architectural decisions**
  - Drift: architectural decision that does not violate higher level design decisions
  - Erosion: architectural decision that does not violate higher level design decisions
- **Solution: a disciplined evolution process…**

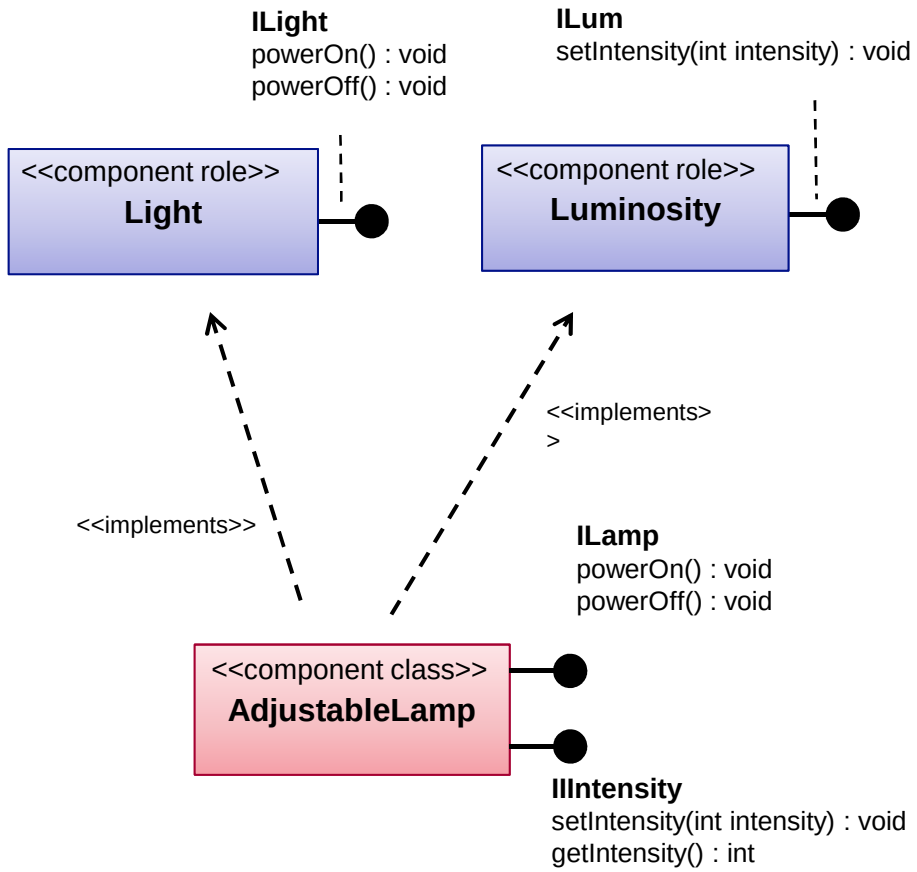
Evolution of three-leveled architecture models in Dedal

- **Solution: … based on a formal metamodel**
  - written in B (first-order logic, set theory based formal language)
  - formal definition of the relations between components on each architecture description level (intra-level relations)
    - connection, specialization (substitution)
  - formal definition of the relations between the different architecture description levels (inter-level relations)
    - implementation, instantiation
  - Derived from object type theory (*Liskov* 1993)



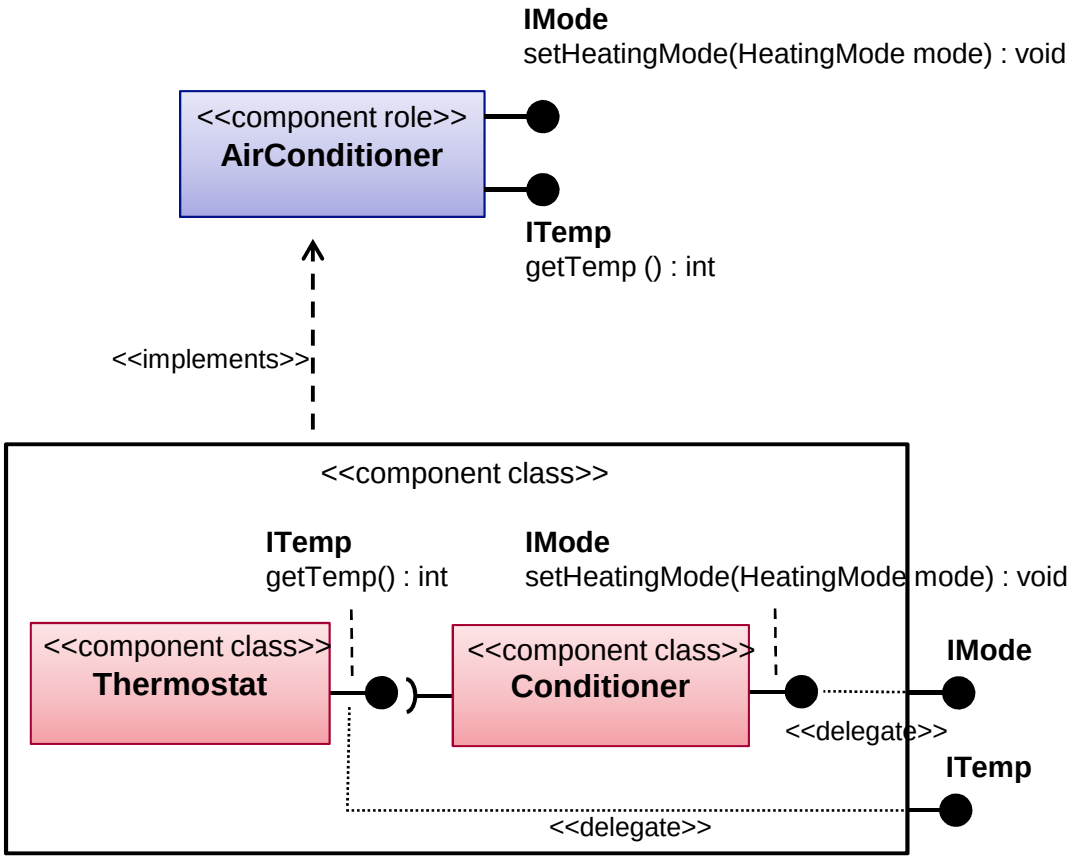
Evolution of three-leveled architecture models in Dedal

- **Example: implementation relations**
  - N-M relations between component roles from the specification level and component classes from the implementation level



Evolution of three-leveled architecture models in Dedal

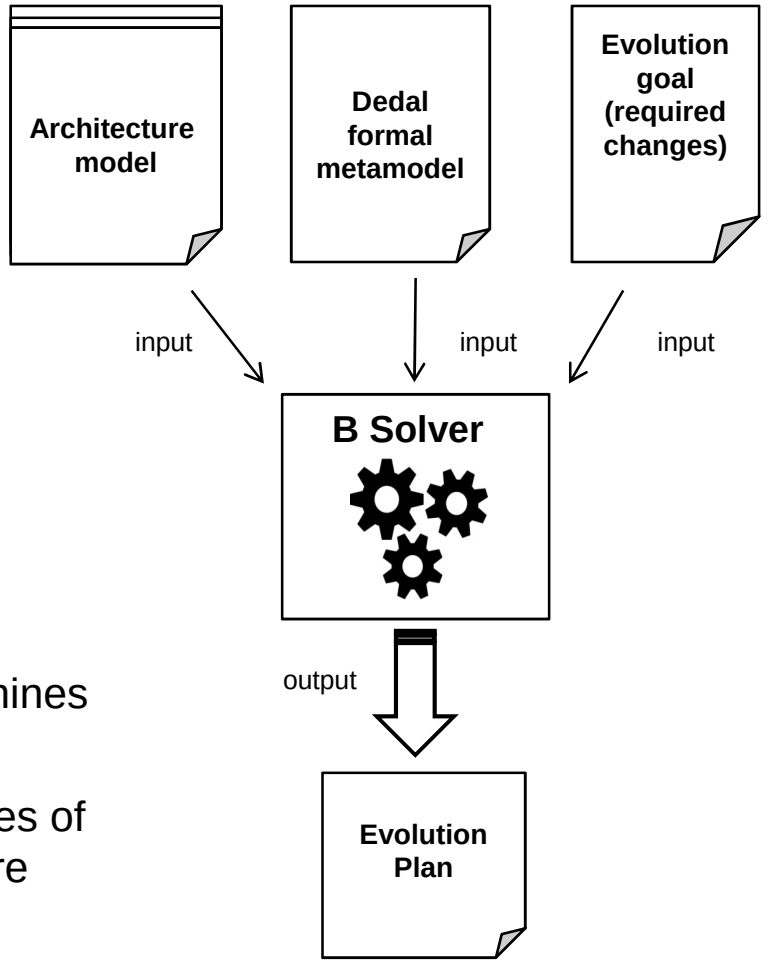
- **Example : implementation relation**
  - a N-M relation between component roles from the specification level and the component classes from the implementation level



Evolution of three-leveled architecture models in Dedal

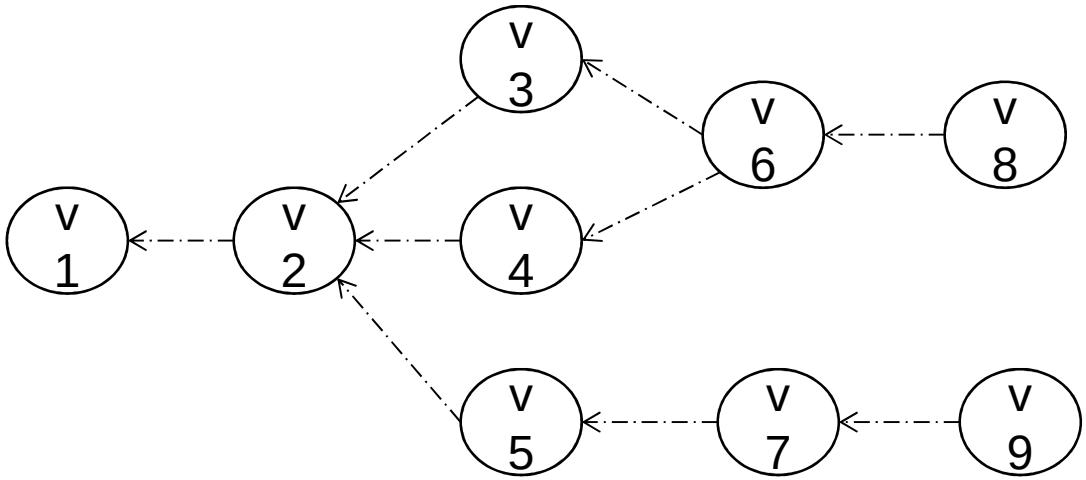
- **A complex evolution process…**
  - Change initiation
  - Local impact analysis (intra-level consistency checking)
  - Local consistency restoration (intra-level change propagation)
  - Global impact analysis (inter-level consistency checking)
  - Global consistency restoration (inter-level change propagation)

- **… hopefully assisted by a solver**
  - architectures definitions considered as state machines
  - changes considered as state transitions
  - automatic generation of evolution plans (sequences of changes) that realize required changes and restore local and global consistency



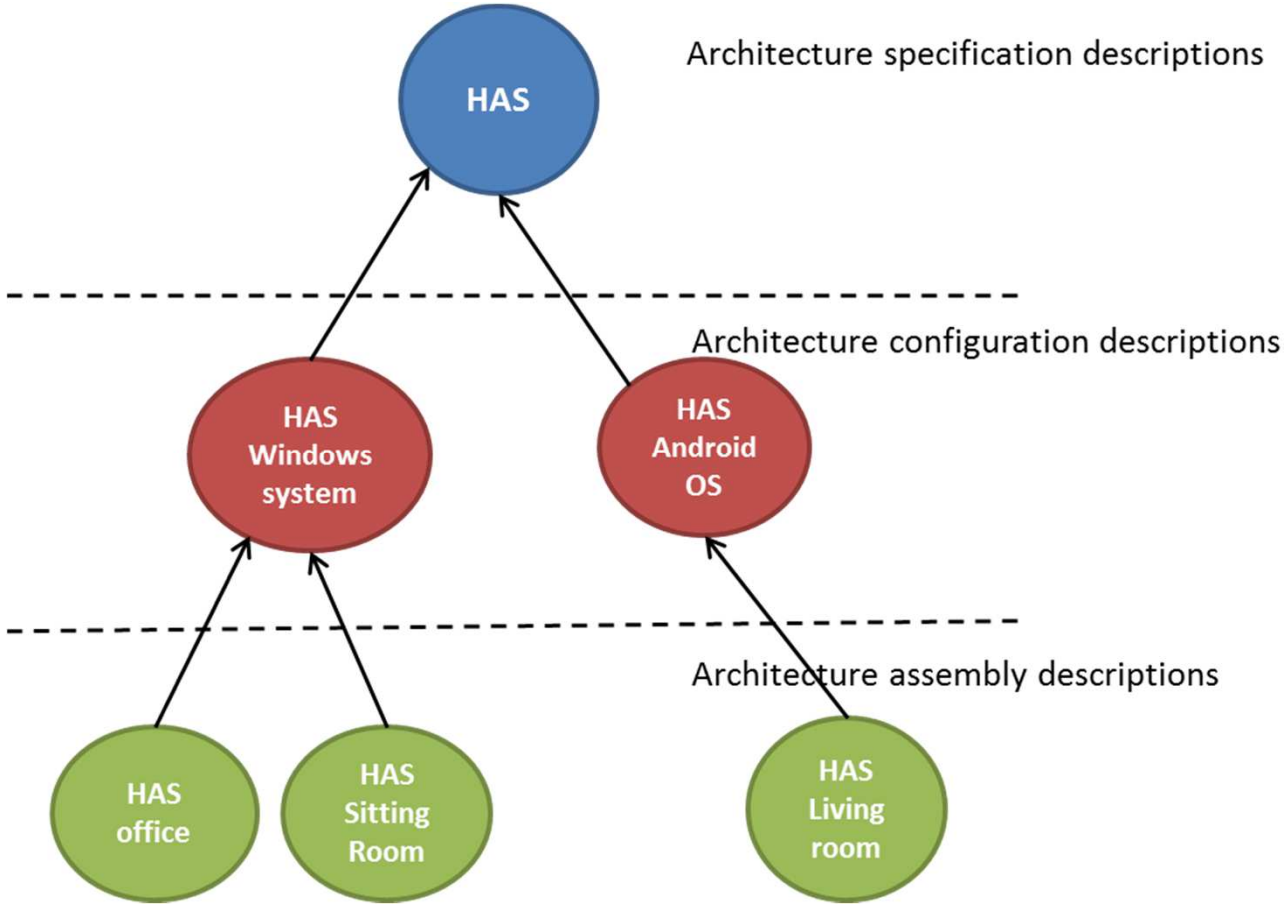
Three-level versioning model for architecture descriptions in Dedal

- **Requirements: manage a version space**
  - to store all the designed versions of architectures
  - to trace all the architectural decisions that define architectures (historic derivation relations)
  - to handle the different semantics of derivation
    - revision: the new version of the architecture is intended to replace source versions
    - variant: the new version is intended to co-exist with source versions



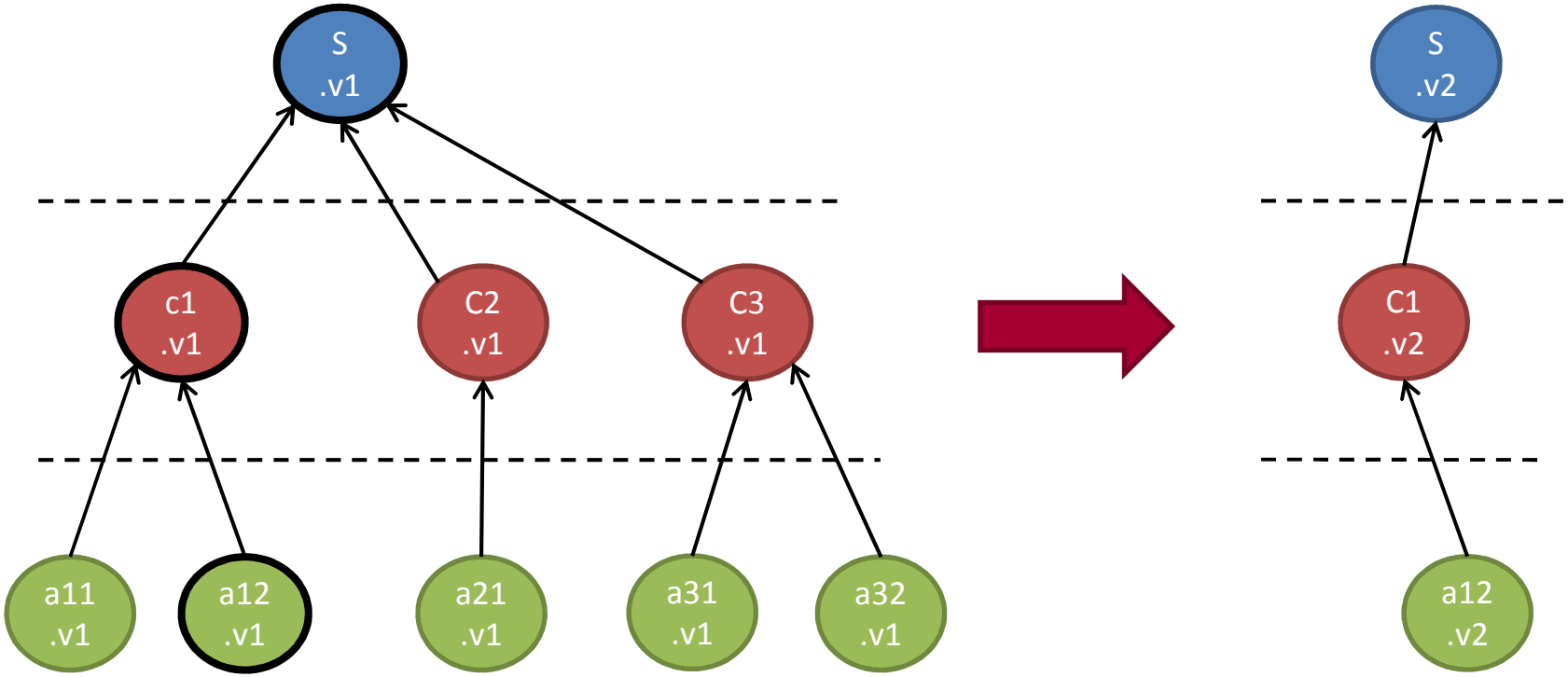
Three-level versioning model for architecture descriptions in Dedal

- **Problematics: combining version space with architectural space**



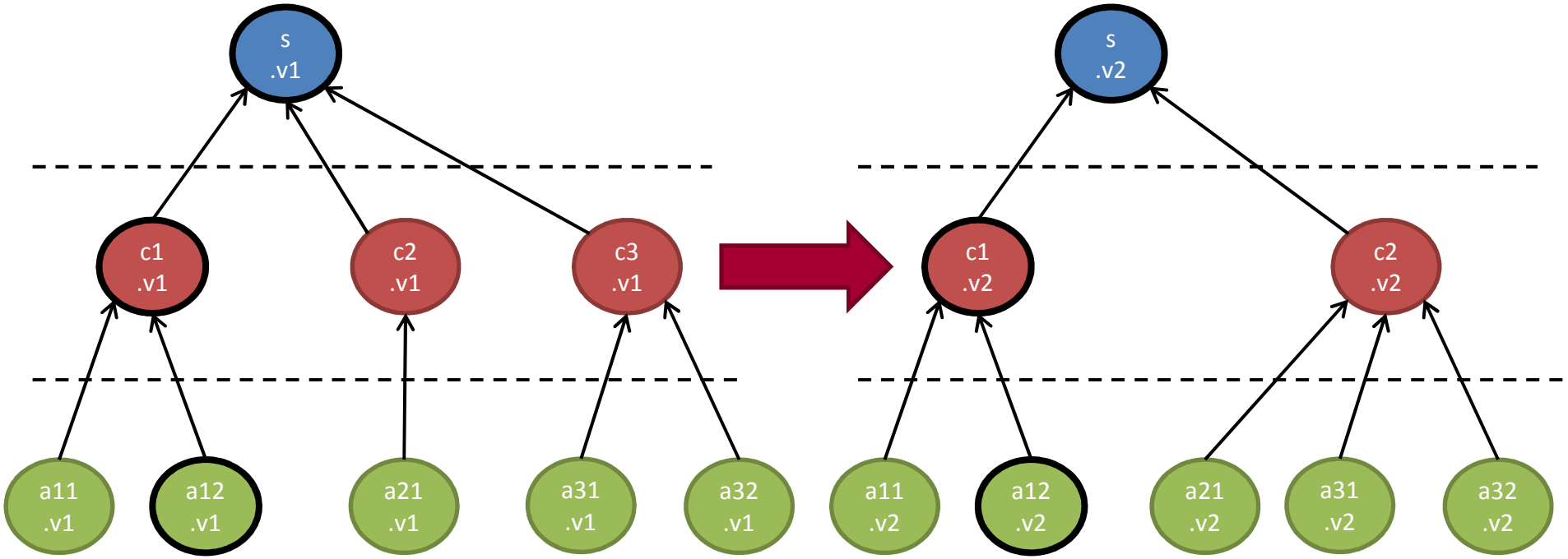
Three-level versioning model for architecture descriptions in Dedal

- **Proposal: assisted versioning strategies**
- **Minimum derivation strategy**
  - Derive only the impacted architecture definitions
  - Suitable for architecture variant derivation



Three-level versioning model for architecture descriptions in Dedal

- **Maximum derivation strategy**
  - Derive the whole architectural space
  - Suitable for architecture revision derivation



Conclusion

- **Dedal ADL and tools**
  - Capture architectural decisions
    - a three-level architecture description language
  - Maintain architectural decisions
    - a disciplined and assisted evolution process
  - Reuse architectural decisions
    - a semantic versioning model
- **Future work**
  - formal definition of the derivation relations (variant, revision)
  - formal definition of version space consistency properties
  - automatic management of versioning (automatic consistency checking and derivation)

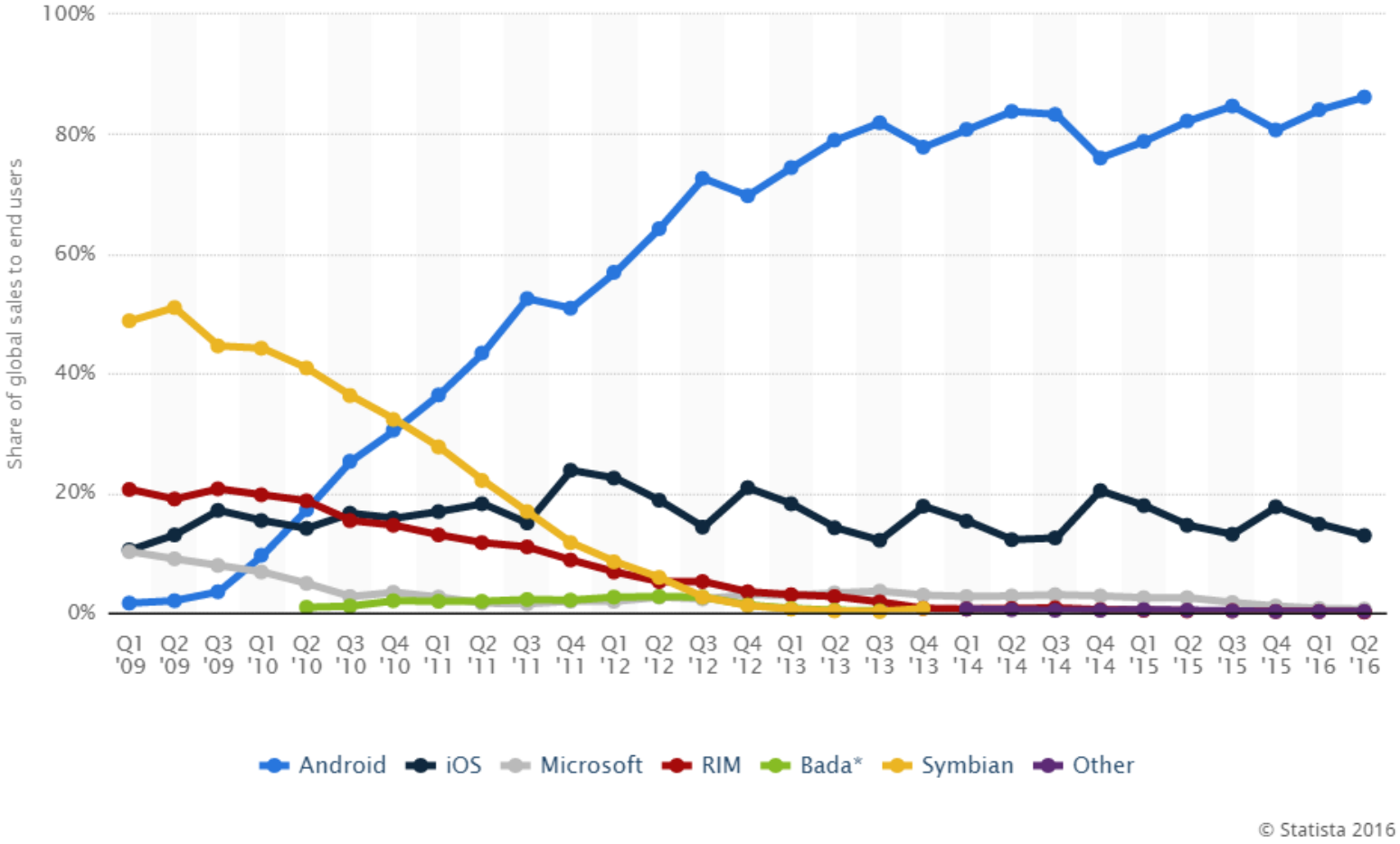
iteratec

KOMPETENZ, DIE ENTLASTET

Hybrid vs. Native Apps

Michael Gebhart

Mobile OS Market Share



- Android: 86.2%
- iOS: 12.9%
- Microsoft Windows: 0.6%
- Others: 0.3%

Native vs. Hybrid Development

Support for Android and iOS

Challenges for Building Applications and Services for Smart Devices

Mira Kajko-Mattsson
KTH Royal Institute of Technology
Sweden

Challenges

- We need smart education
- We need smart organizations
- We need smart employees
- We need smart methods

Smart education

- Education is not smart today.
- Educators cannot imagine what our future will be like in 20 years.
- Educators must choose a portfolio of subjects that prepare students for work for at least 10 years ahead.
- Educators do not have all the competencies required for teaching the subject portfolios.
- Help needs to be acquired from outside.
- The subjects’ needs will only increase.
- Students are not well prepared for developing applications and services for smart devices.

Smart Organizations

- Constantly evolving organizations
- Highly innovative and productive mills
- More flexible and more competitive and still have control over what they do and how they act.
- Encourage the development and improvement of new devices and services.

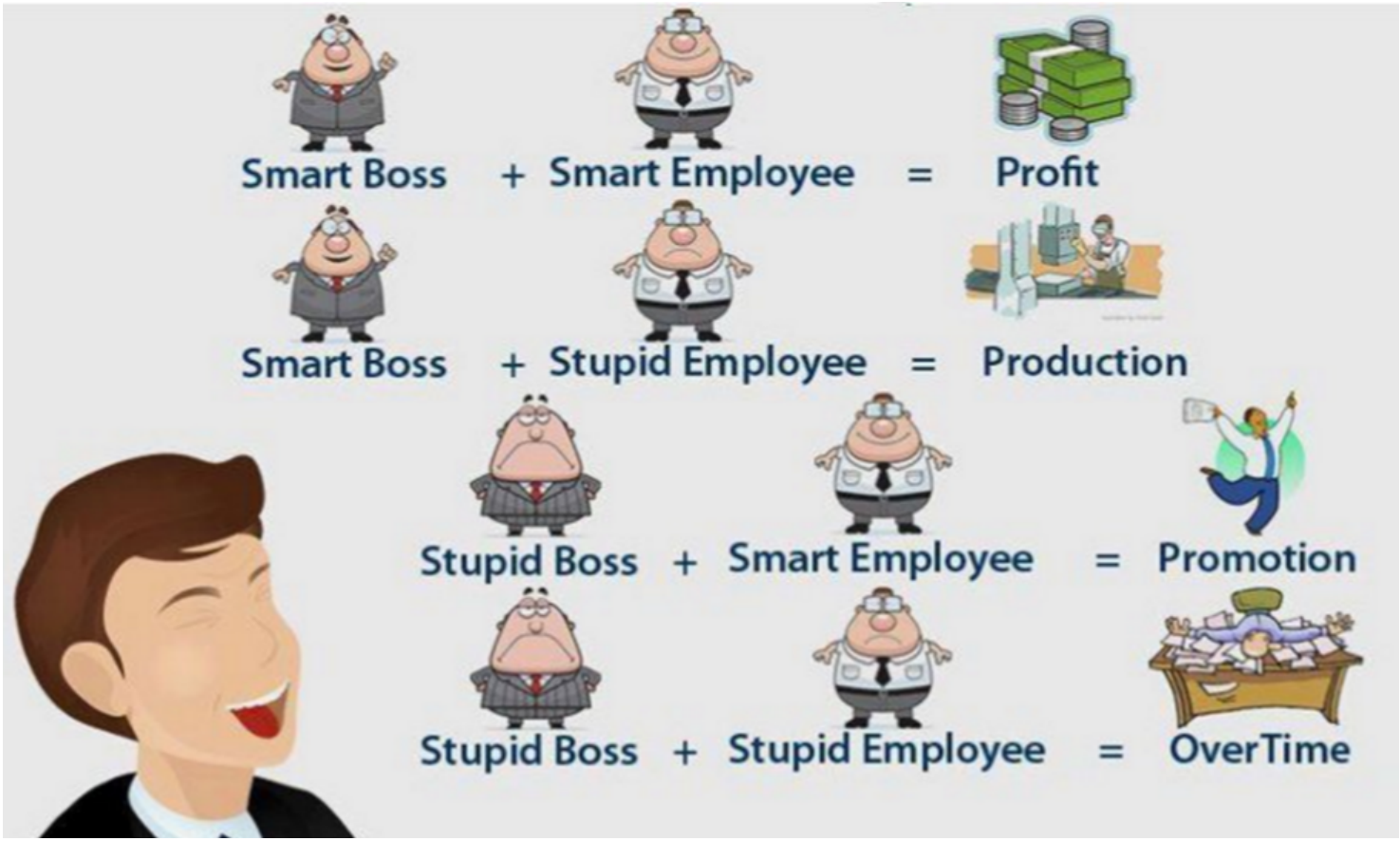
Smart ~~Agile~~ Organizations: Towards Innovative and Highly Productive Mills

Smart ~~Agile~~ methods

- Idea generation and productivization
- Spontaneous order
- Emergent adaptations

- Communication
- Continuous learning envir.
- Fluid roles and dynamic decision making
- Management and co-ordination
- Organizational structures

Smart employee



There’s a new brand type of employee out there.
The Smart Creative.

Smart employee

- Smart creatives causes change.
- They do not hold back whenever they have an idea that can improve the world.
- They always find solutions to major problems.
- They are not afraid to fail or try smth new.
- They make sure that their ideas are foolproof.
- They influence other employees and make them better.
- They bring fast-paced thinking and problem-solving.
- They find ways to work smarter.
- They reinvent the wheel while being original and forward-thinking.
- Smart creatives come in all shapes and sizes. There is no race, gender, sexual orientation, education.

Smart methods

What does cooking have to do with developing applications and services for smart devices?

WWW.IARIA.ORG

PANEL on ICSEA 2016

Title: Challenges for Building Applications and Services for Smart Devices

on

Comfort/Heat Computational Requirements

in

developing apps for wearable/implantable devices

Petre Dini, IARIA

Requirements for Software/Apps

GENERAL CONSIDERATIONS

- **Centralized systems | hardware vs. software**
- **Distributed systems | hardware vs. software**
- **Real-time systems | embedded software**
- **Mobile systems | systems on the chip**
- **Wearable systems | systems on the chip**
- **Implantable systems | systems on the chip**
- **Body systems | cyberman**

- **Requirements → Systems → Testing and Validation**
- **Mobile/Wearable/Implantable → Human Behavior/ Body Features**

Specifics of requirements for Apps

- **Standardization and methodologies**

  Screen Sizes

  API for many OSs

- **Special considerations for Requirements, as Humans are heavily involved**

- **Classical: *functional / non-functional***
- **Specific for Apps: *functional / non-functional / comfort-requirements***

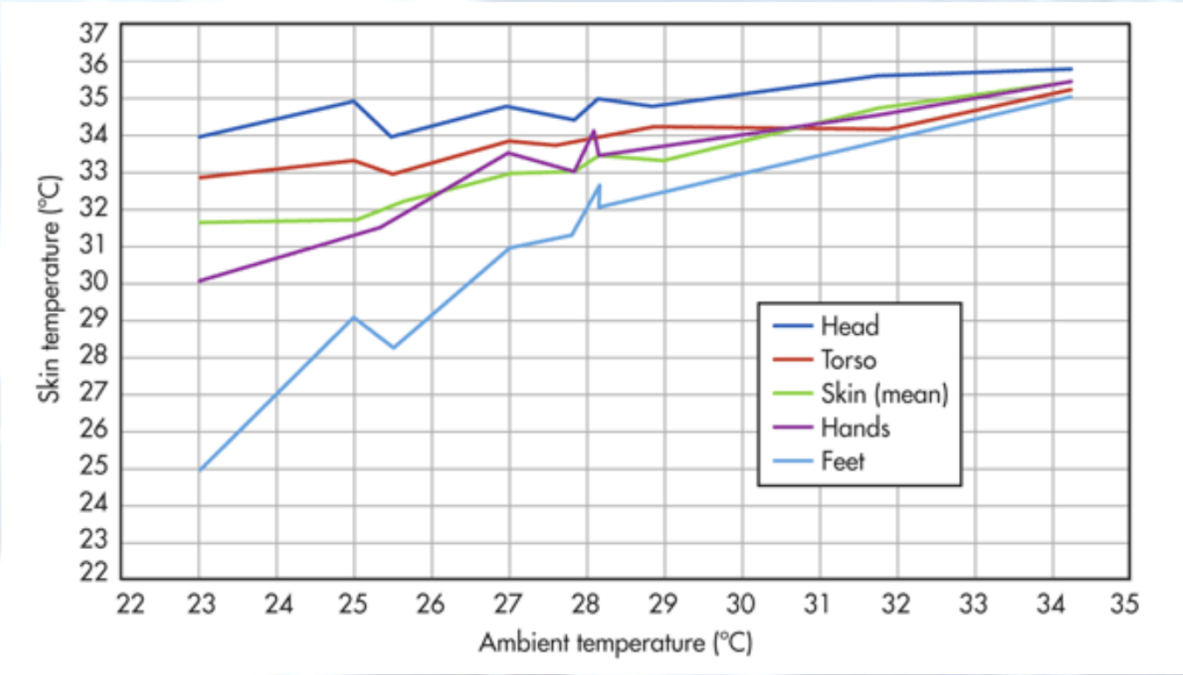
  Thermal considerations

  Material and environment [moisture/humidity/cold, human body reactions, isolation]

  Testing [human-in-the-middle]

Thermal considerations

- A specific aspect is that wearable devices introduce some unique thermal design challenges that should be considered for devices, Apps and the entire system. This is not only referring to operability, but also to a required comfort level for humans. This design challenge is mainly for processor intensive applications and units with complex displays.



Heussner, D. Texas Instruments, USA
http://electronicdesign.com/digital-ics/wearable-technologiespresent-packaging-challenges

Cyberman


E-textiles/
clothing
LED
clothing
Tattoos
Wrist
bands
Smart
shoes
Eyeglasses
Earbuds
Jewelry
Rings
Ankle
bands

Computation/communication/heat issues

Comfort requirements

- Esthetic (color, size)
- Shape (form, fitting)
- Attachment status (mandatory, removable)
- Heat-related (computational, device-material, ambient)

Heat comfort requirements

- Process intensive applications
- Complex display
- Fast data communication (health hazards, alarms, critical applications)

Testing-for-Real on the above is mandatory

- *As wearable devices are quite specific, simply substituting them with emulators is not suitable; as the discipline is evolving in a rapid pace, trusting the results of such emulator is doubtful.*

Solutions and Challenges

SOLUTIONS

- #1 Monitoring the heat on a wearable/implantable device
- #2 Forward intensive executions when a heat threshold is reached:
  - To an idle body devices (for cyberman)
  - To a remote server
- #3 Brig back computation, when comfortably acceptable

CHALLENGES

- Different mobile devices need different user interfaces. With regard to screen size, *automated GUI generation with automated tailoring may become an option*.
- What is specific on designing and testing wearable devices and Apps is that *user experience is more relevant than in traditional approaches*.
- "It is a challenge to develop and test very specific features; e.g., "smart watches have very small screens and almost no buttons, *making the use of space, navigation and user interaction incredibly important*""

Thanks

Qs

WWW.IARIA.ORG