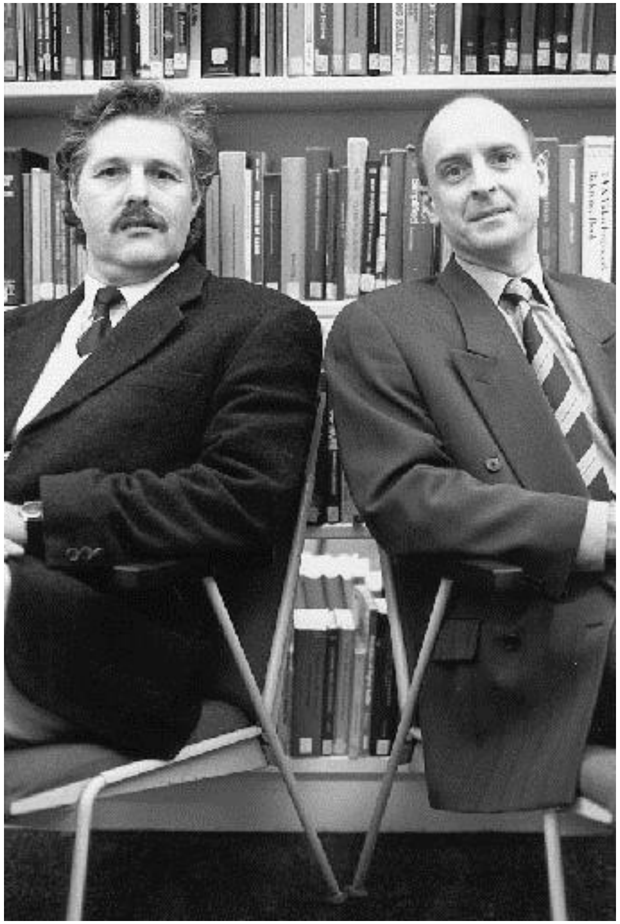
The past of ESI – and architecting

Prof Dr Ir Egbert-Jan Sol

Innovatie Directeur High-tech systems and materials

Do you know them?



1997: ESSI
(Eindhoven Embedded Systems Institute)

Het EESI heeft drie hoofdthema's gekozen:
draadloze thuisnetwerken,
mobiele multimediasystemen en
navigatiesystemen voor transport & logistiek.

Past of E(E)SI

- 1996: Min EZ (Wijers) start 4 Technological Top Institutes
- out of 16 proposal 4 were chosen (food, metals, polymers & telematics)
  - TU/e with Philips has proposed Embedded Systems,
    - but did not lobby for it hard enough in 1996 and was not selected

- One year later Rick Harwig of Philips decided that TU/e should
- start any way and he requested funding from (Philips), ASML, Oce,
- FEI en Ericsson (each 25K) to start the EESI. Next we Martin Rem, Leo Coolen, Patrick Dewilde and .. selected the first project with IS funding

- 1998-2000 already discussion on systems architecturing. (Arian Zweegers (Architecting, 1998) and Rob de Graaf (Concurrent Engineering, 1996) with an assessment method including one on capabilities from initial to mature architecting.

Architecture Competence Program - page 1/4

Input Egbert-Jan for 15 Dec 97 workshop (modeled after Philips)

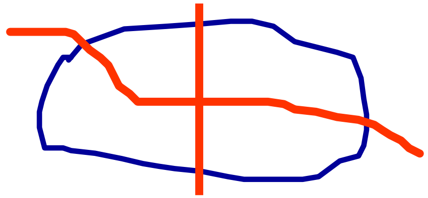
Goal:
Improve architecture competence within Ericsson by supporting the development of top-quality system architects for Ericsson and securing a continuous supply of this scarce competence

Definition of system architecture: (for other definitions see http://www.sei.org)
Complex systems need structuring into modules & interfaces.
Complex systems survive only if they adapt to their environment.
Architecturing (managing modules + interfaces) is maintaining the system integrity during the evolution of complex systems.

Goal of architect is to identify (in advance) the (future) changes in requirements (market) and (new) technology and adapt the system (or build a new one) while maintaining the system integrity

Architectures definition

Complex System:
how to structure
in smaller blocks



Definition: architecture =
modules + interfaces

Interfaces:
Architecturing = management of interfaces
Architect = responsible for the system integrity
and owner of the interface

"One should introduces interfaces to open systems,
but one should never open-up one's own core competences"

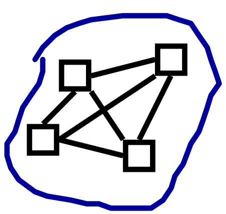
Architecturing

Architecturing is focussing on evolution,
on a facilisation to change, (learning curve)
while maintaining the integrity of the system

Focus on change is the difference from (software) engineering

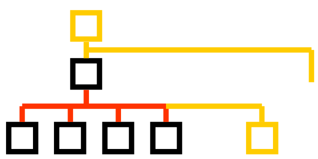
Value of a good architecture:

"Future Flexibility"



bad

good



Architecture Competence Program - page 4/4

Training program and competence network:

30% on architecture of (different and future) Ericsson products
30% on marketing (architects work together with marketers)
30% on (social) communication skills (multi-cultural, presentations,)
10% on theory (definition, means and methods, interface languages, .)

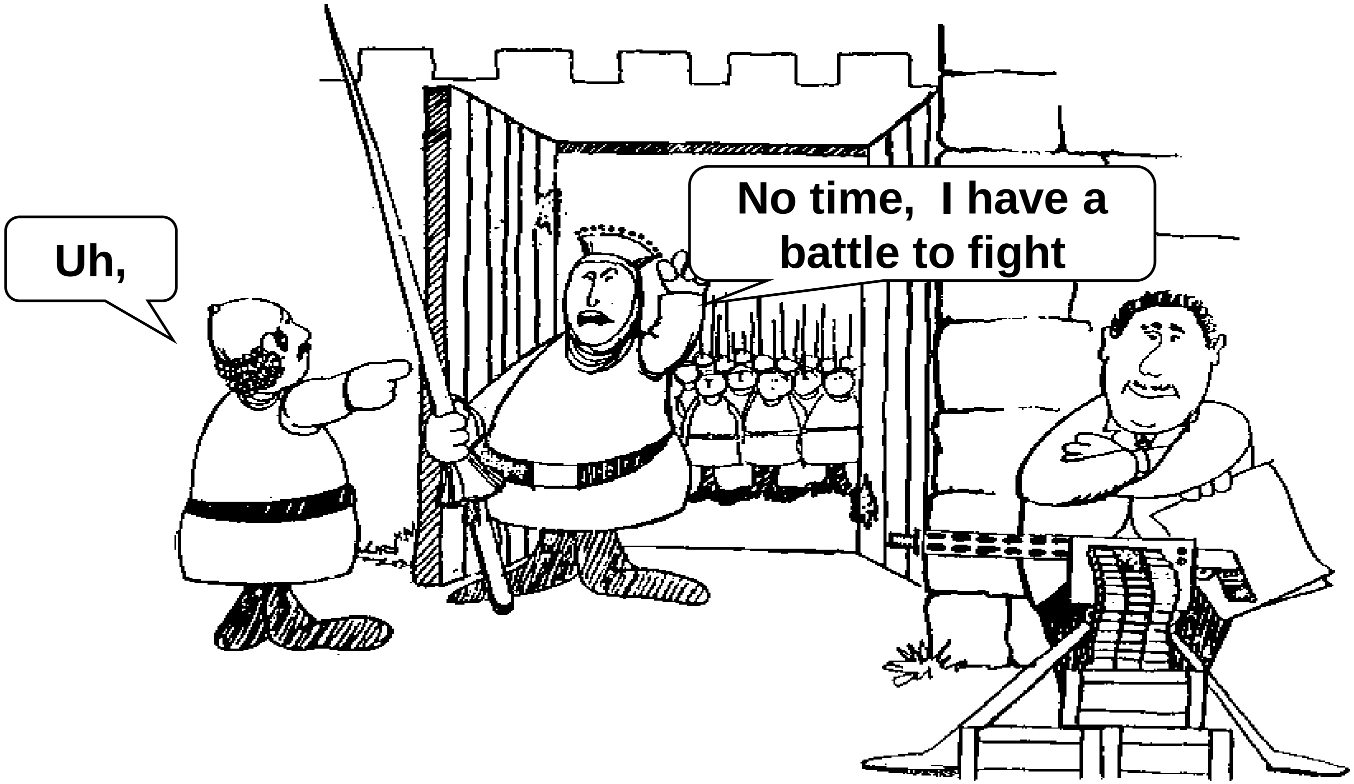
roulation program
rapid learning curve but changing working environment/project more often and be confronted with complete different technologies (hardware/chip design, real-time software, protocols, ….)

(forced) assignment (on part-time basis) to investigation programs
have senior and junior architects from mixed business areas investigate
and experiment with (new) important technologies (and learn to know each other in action (workmeetings) and not during seminars, etc.)

ERICSSON

Architecting: From art to professional discipline

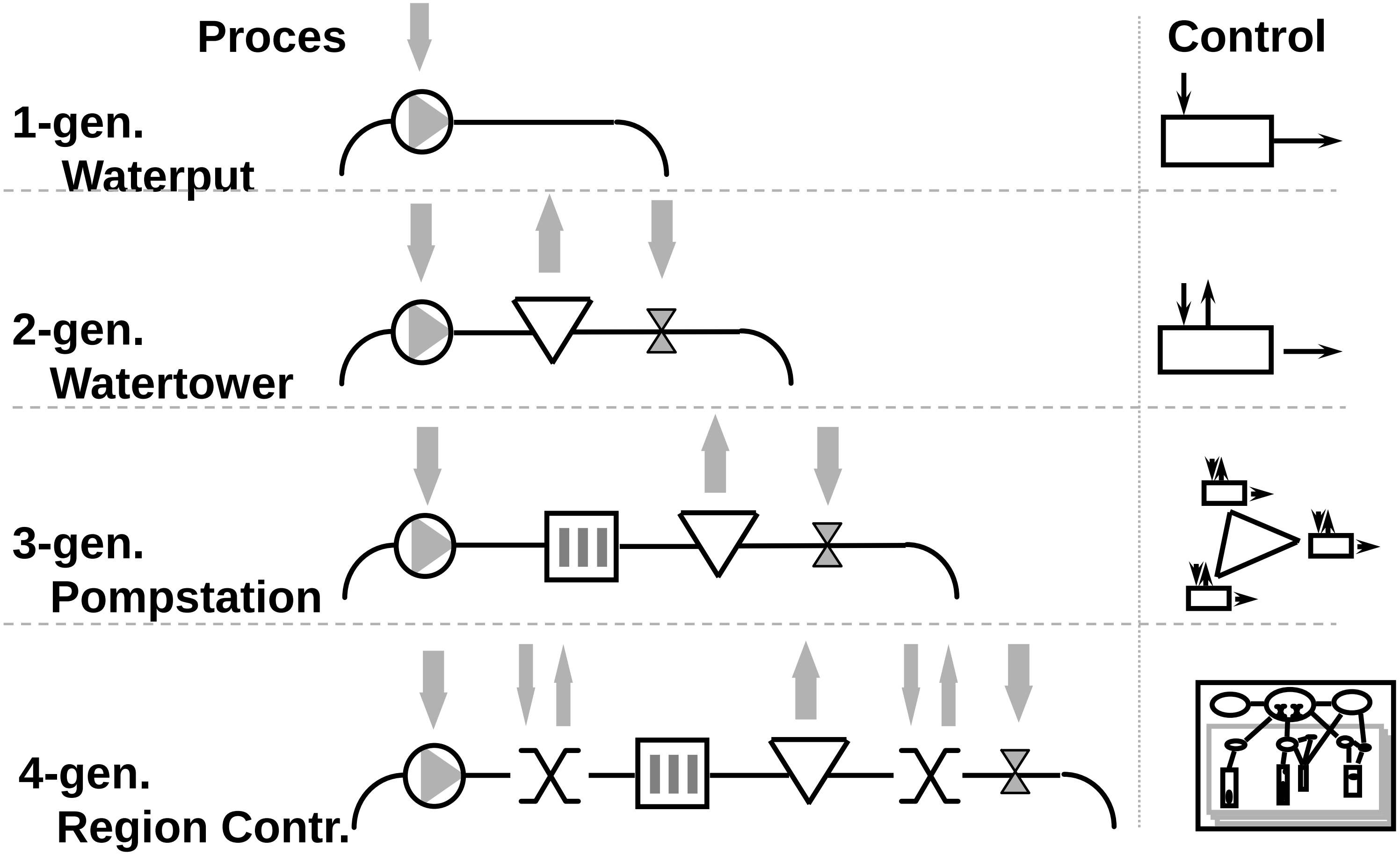
“managing the future flexibility of complex systems”



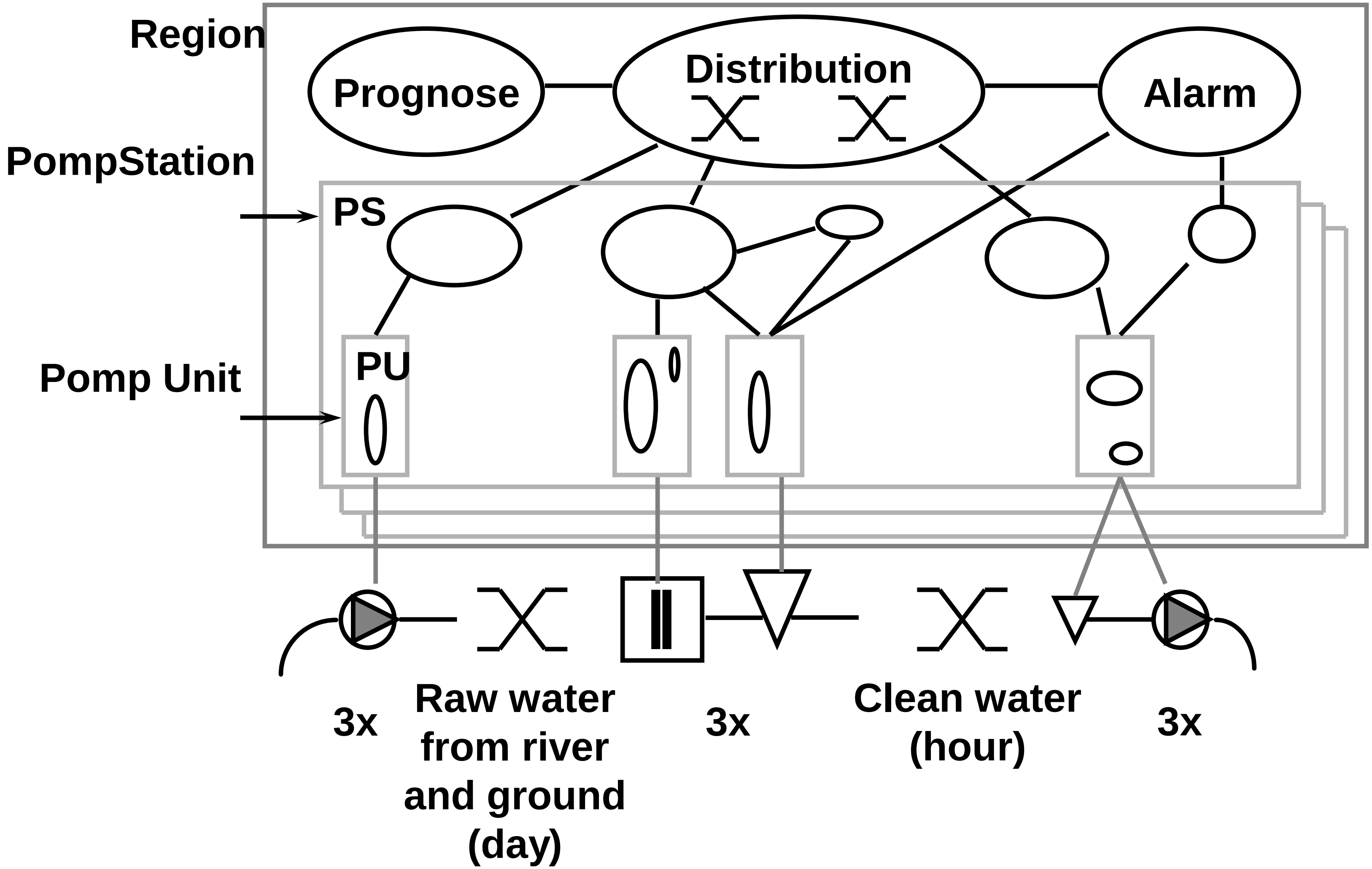
Egbert-Jan Sol

Eindhoven, 19 jan 2000

Basic process & Automation



Architecture

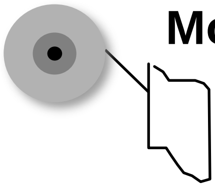


Evolution of software systems

Small program to support hardware

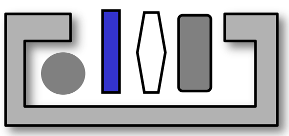
500 - 5K LOC= lines of code

LOC= lines of code



Monolithical software architectures

started around one issue to automate
designed in the 70, implemented in COBOL (<50 KLOC)



Proprietary bus architectures

software size grows larger (500KLOC), while modules were added to legacy systems. Need to structure

Improving large scale software systems

Performance of a systems must improve continuously:
this requires continuous change (learning curve behaviour)

Architecturing: Manage the continuous changes

1. renovate existing code
if maintenance costs too high & functions still OK

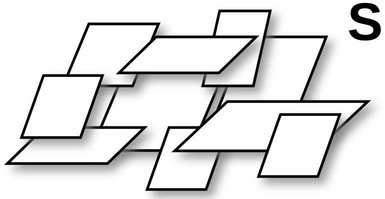
2. make user/interactive/external part more flexible
if many change requests exist in user interface part

3. make (kernel) transaction part more flexible
if transactions (here connections) cause more problems

4. build new
if business processes change heavily

+ combination of strategies

Shop Floor Control evolutions

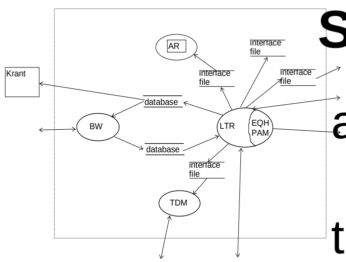


SFC 1

Key module for Work Tracking/Logging:

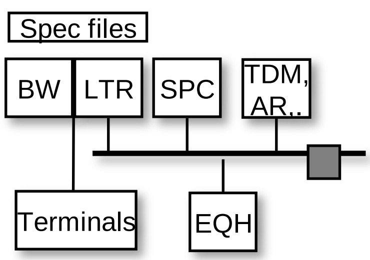
Technical basis:

COBOL (1960), VAX (1980) hardware,
indexed sequential files (1970) as database



SFC 2 (improvement or more extentions)

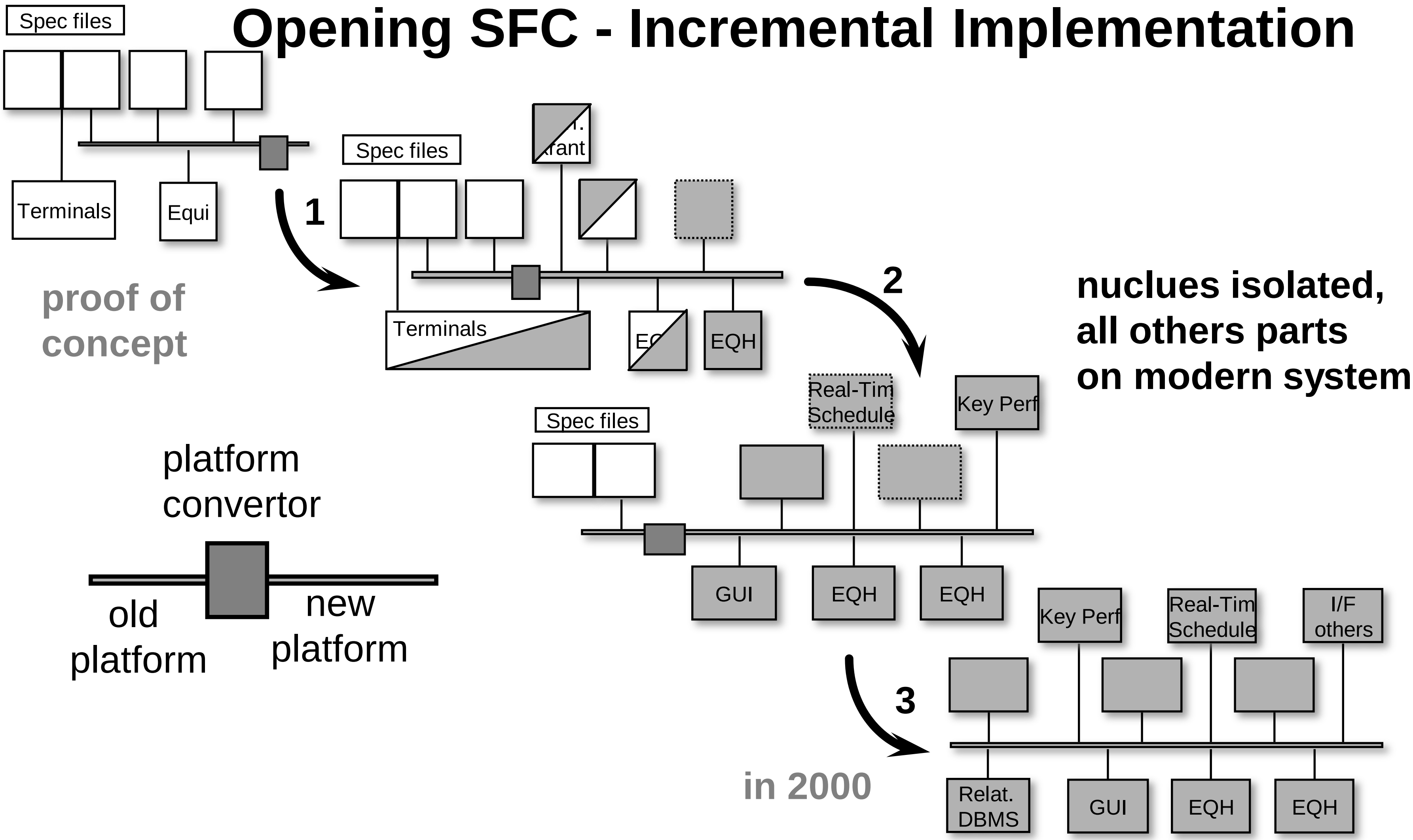
audit: SFC 1 become a legacy application
no global design, mix of functions, complex I/F, improve (re)structu
today: I/F (interfaces) restructured between subsystems



SFC 3: the open application system

but how to get there in a running facto

Opening SFC - Incremental Implementation

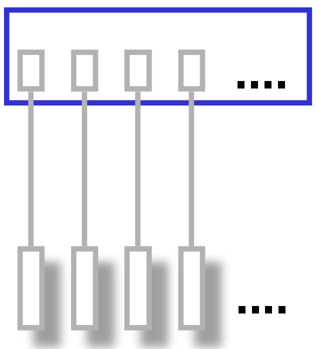


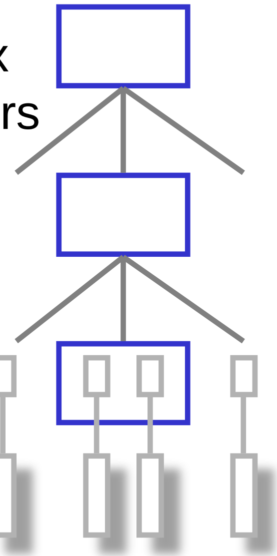
Architecture principles

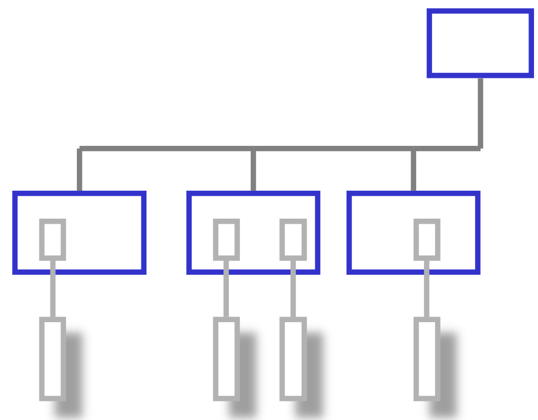
Monolith → **Hierarchy** → **Distributed**

Complex
add layers

Robust
no central point of failure





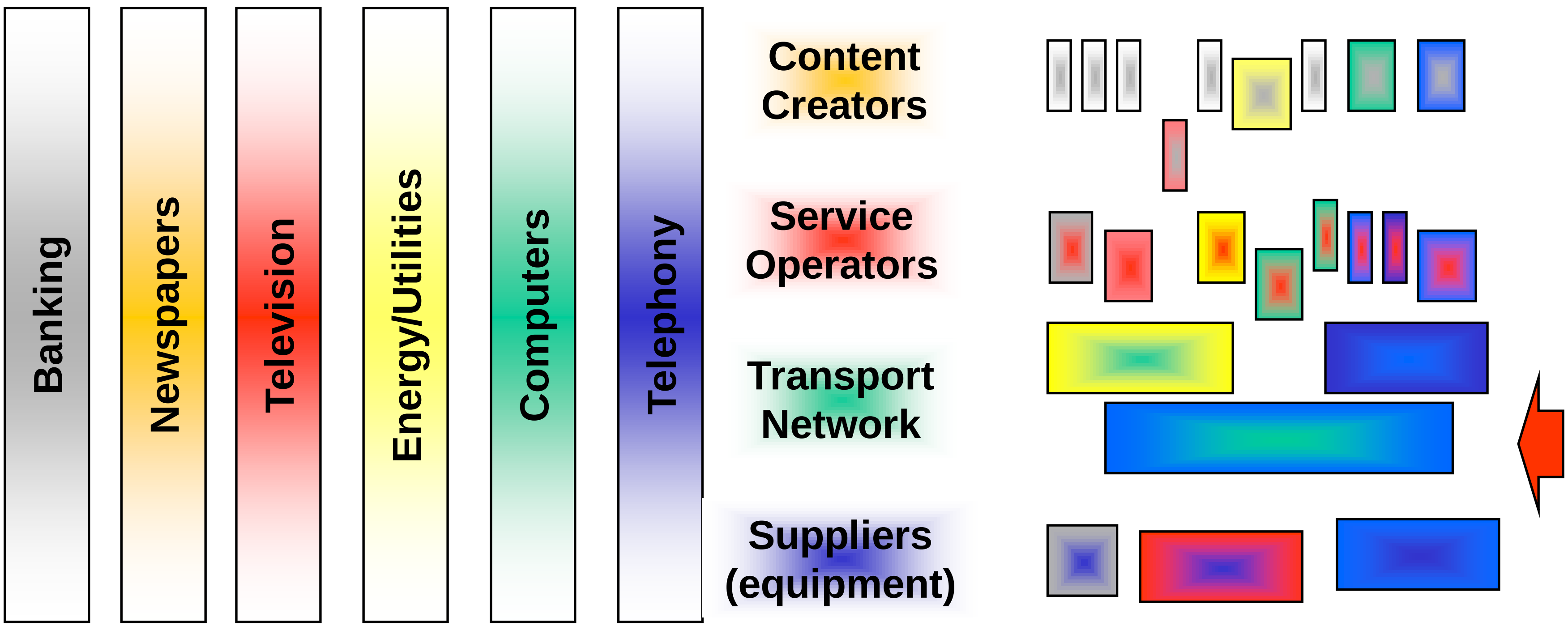


1-TO-1 (POMPSTATION AND OWN UNITS)

CONTROL OVER POMPSTATIONS (PU, PS, Region)

DISTRIBUTE CONTROL,
but much DATACOMM. needed
in a Wide Area Network (WAN)

From vertical chains to segmented value chains



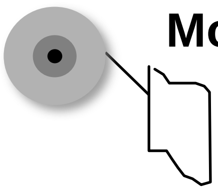
Yesterday' vertical markets
(e.g. in Computers: IBM, Digital, ..)

Tomorrow
(e.g. Microsoft in Operating Systems)

Evolution of software systems

Small program to support hardware

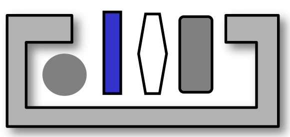
500 - 5K LOC= lines of code



Monolithical software architectures

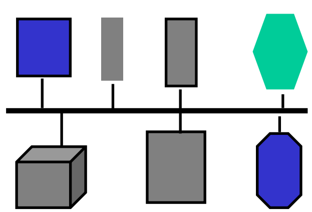
LOC= lines of code

started around one issue to automate
designed in the 70, implemented in COBOL (<50 KLOC)



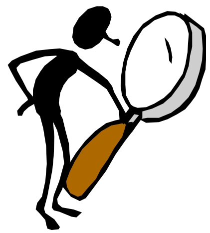
Proprietary bus architectures

software size grows larger (500KLOC), while modules were added to legacy systems. Need to structure



Open bus architectures

buy world-class modules (appli., rel. DBMS) as software otherwise grows too large (toward 5MLOC)
but how interface it: use open standard



Software Development Evolution

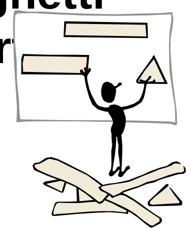
Software: algorithm + data + control

(static/sequential) (dynamics, states)

Algorithm
1975

JUMP, GOTO spaghetti -> Structured Programming (Begin…End)

Basic, Cobol, For (Dijksta, NL) Algol, Pascal, C



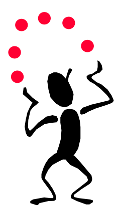
Data
1990

Data defined/used everywhere -> Structured Data Types (Objects

Databases, Object-Oriented Languages (C++)

1995: Java “network centric” (then just a better C/C++)

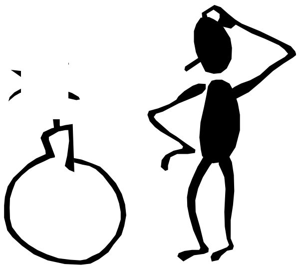
Control
200?



Event, Multiple states, Synchronization, Real-time

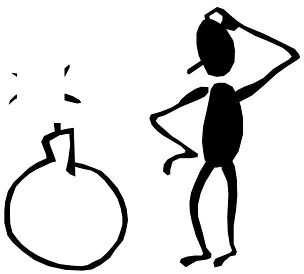
Object sending/receiving messages (what is happening where??)

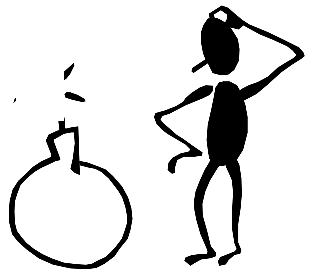
Debugging & Testing



1 Program on 1 Computer ⇒ x number of bugs/
x hours of testing

1 Program on 2 Computers
~~⇒~~ x number of bugs/
x hours of testing
~~⇒~~ 2x number of bugs
2x hours of testing
⇒ $(x)^2$ number of bugs
$(x)^2$ hours of testing





Because it are 2 programs on 2 computers

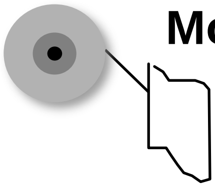
Distributed computing / Network computing is difficult:
you have to test the correct algorithm flow and correct data in all multiple states

Evolution of software systems

Small program to support hardware

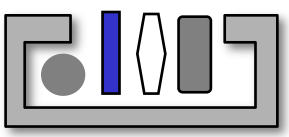
500 - 5K LOC= lines of code

LOC= lines of code



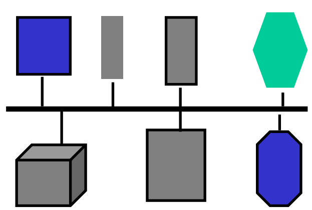
Monolithical software architectures

started around one issue to automate
designed in the 70, implemented in COBOL (<50 KLOC)



Proprietary bus architectures

software size grows larger (500KLOC), while modules were added to legacy systems. Need to structure



Open bus architectures

buy world-class modules (appli., rel. DBMS) as software otherwise grows too large (toward 5MLOC) but how interface it: use open standard

??

Networked Software Architectures

Higher abstraction level:
Architecting becomes key

Architecting

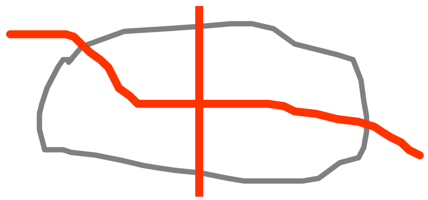
Architecting is focussing on evolution,
on a facilitation to change (learning curve),
while maintaining the integrity of the system

Focus on change is the difference from (softw.) engineering

Phase	Purpose	Output
Reference Model	Common Language	Terms, Definition
Architecture	Define what (functions)	Modules & Interfaces
Design	Define how (cost/preformance)	Specifications Drawings
Realization	Build/Use	a product, control system, a building, ...

Architectures definition

Complex System:
how to structure
in smaller blocks



Definition: architecture =
modules + interfaces

Interfaces:
Architecturing = management of interfaces
Architect = responsible for the system integrity
and owner of the interface

"One should introduces interfaces to open systems,
but one should never open-up one's own core competences"

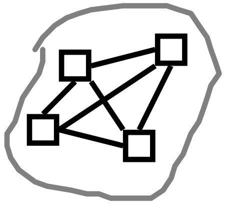
Architecturing

Architecturing is focussing on evolution,
on a facilisation to change, **(learning curve)**
while maintaining the integrity of the system

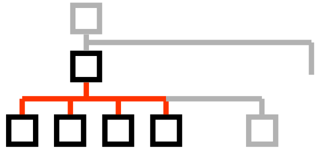
Focus on change is the difference from (software) engineering

Value of a good architecture:

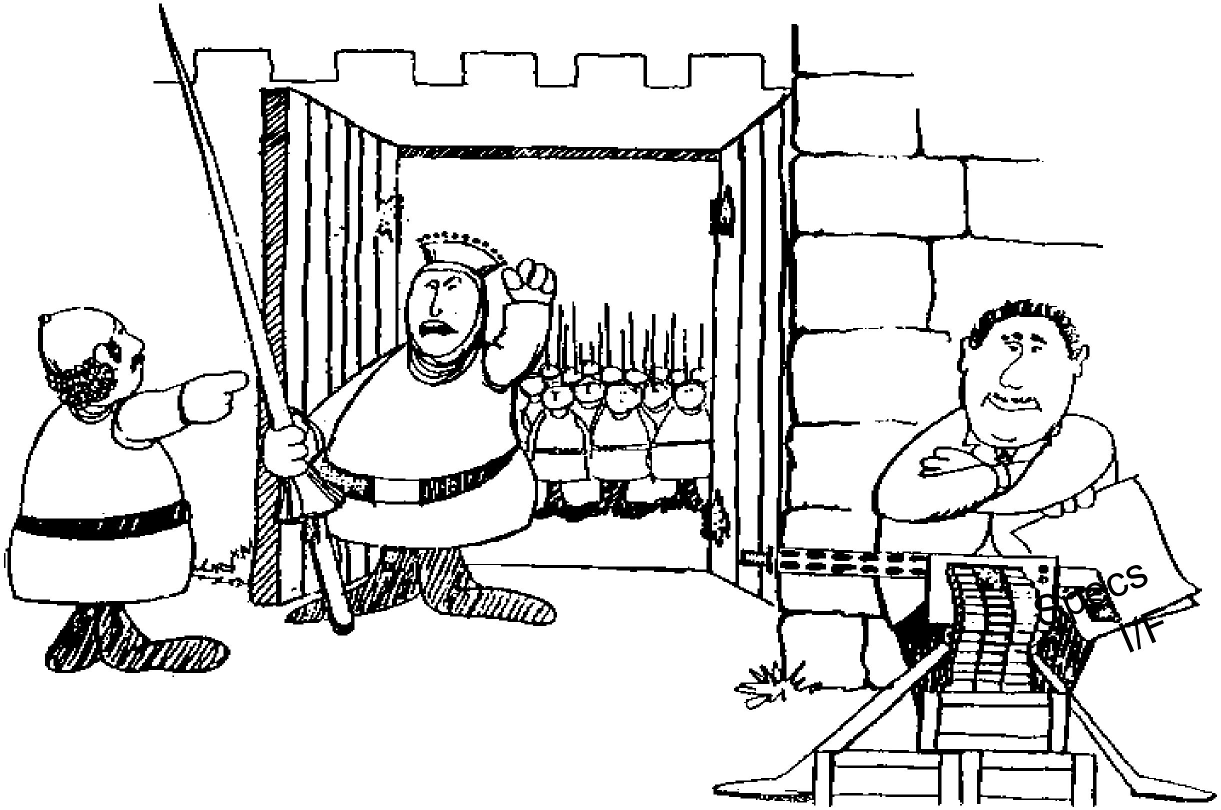
"Future Flexibility"



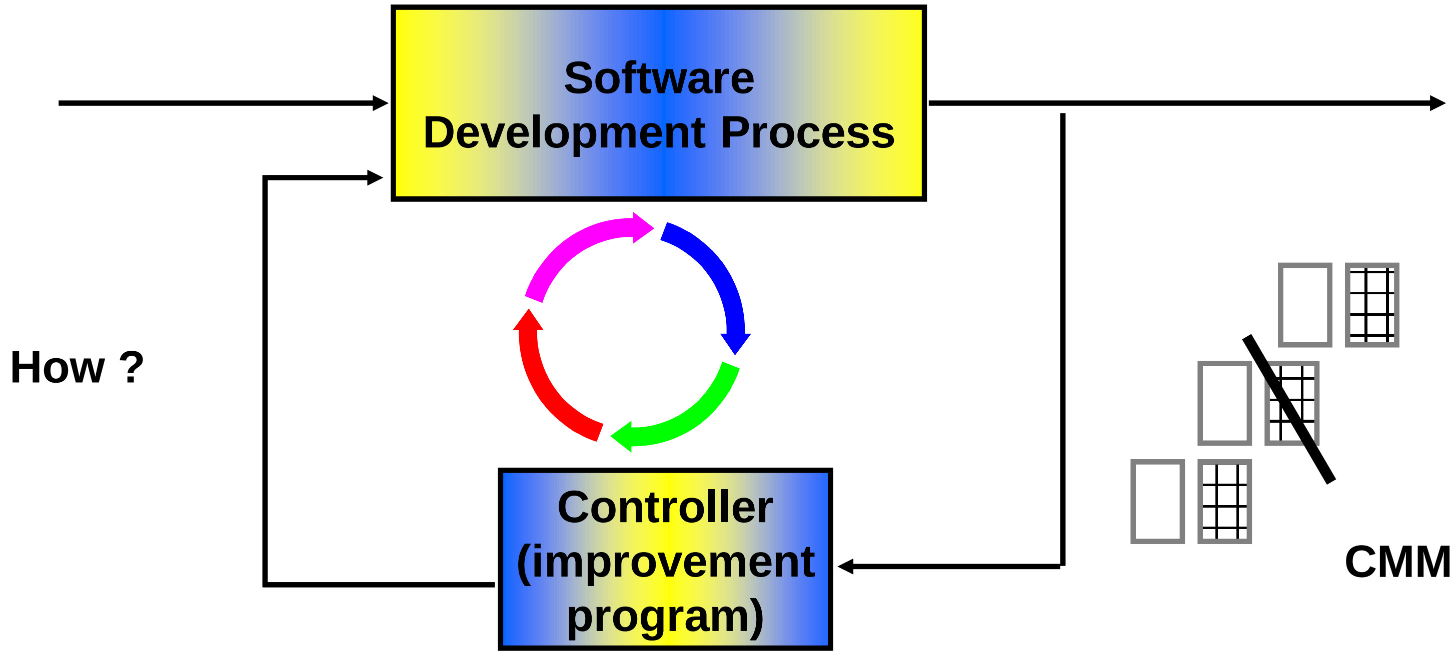
bad **good**



We need System Architects



1-Dimensional Straight Forward CMM

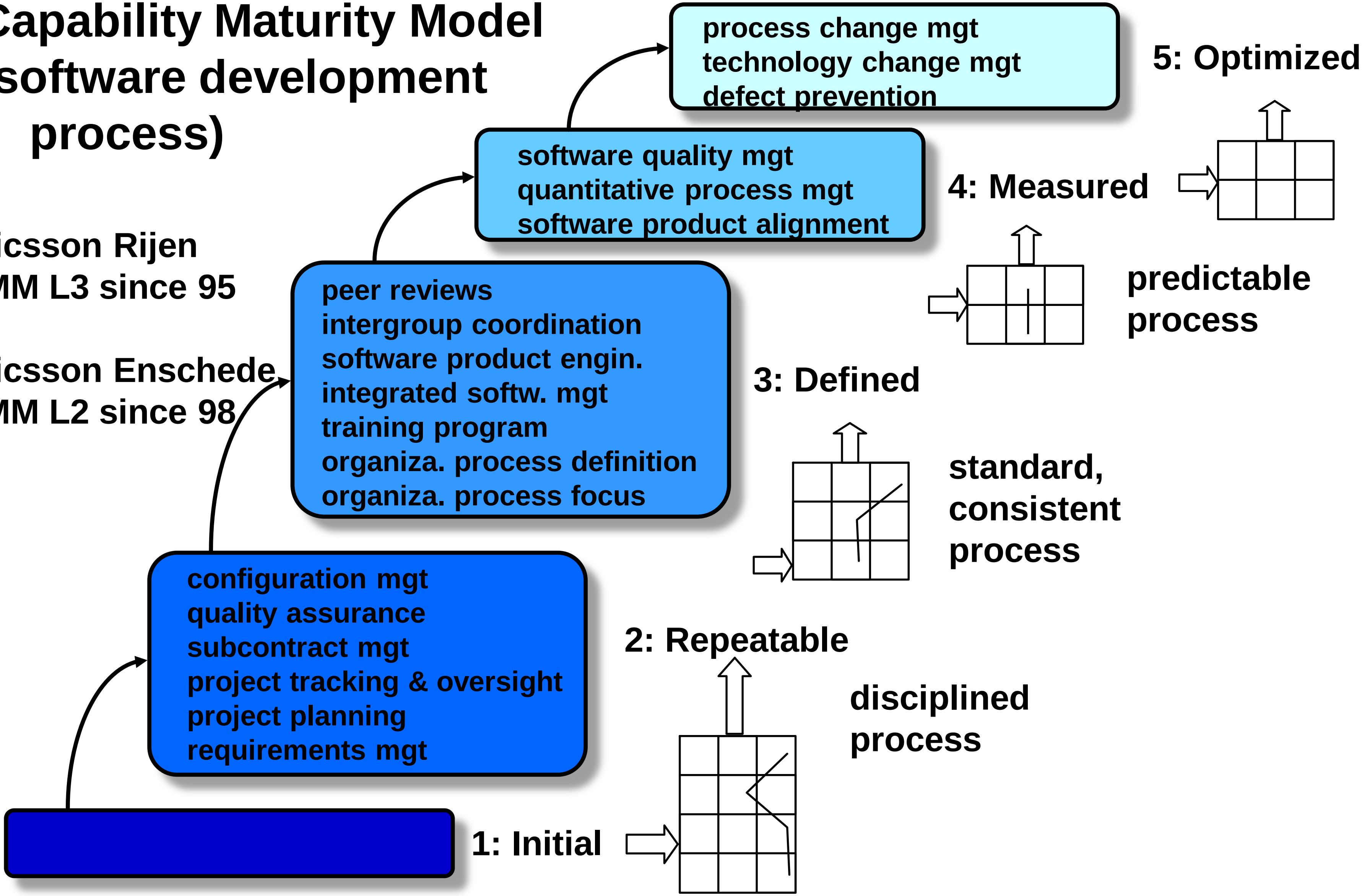


Increment one level
(from L2 to L3 and from L3 to L4)

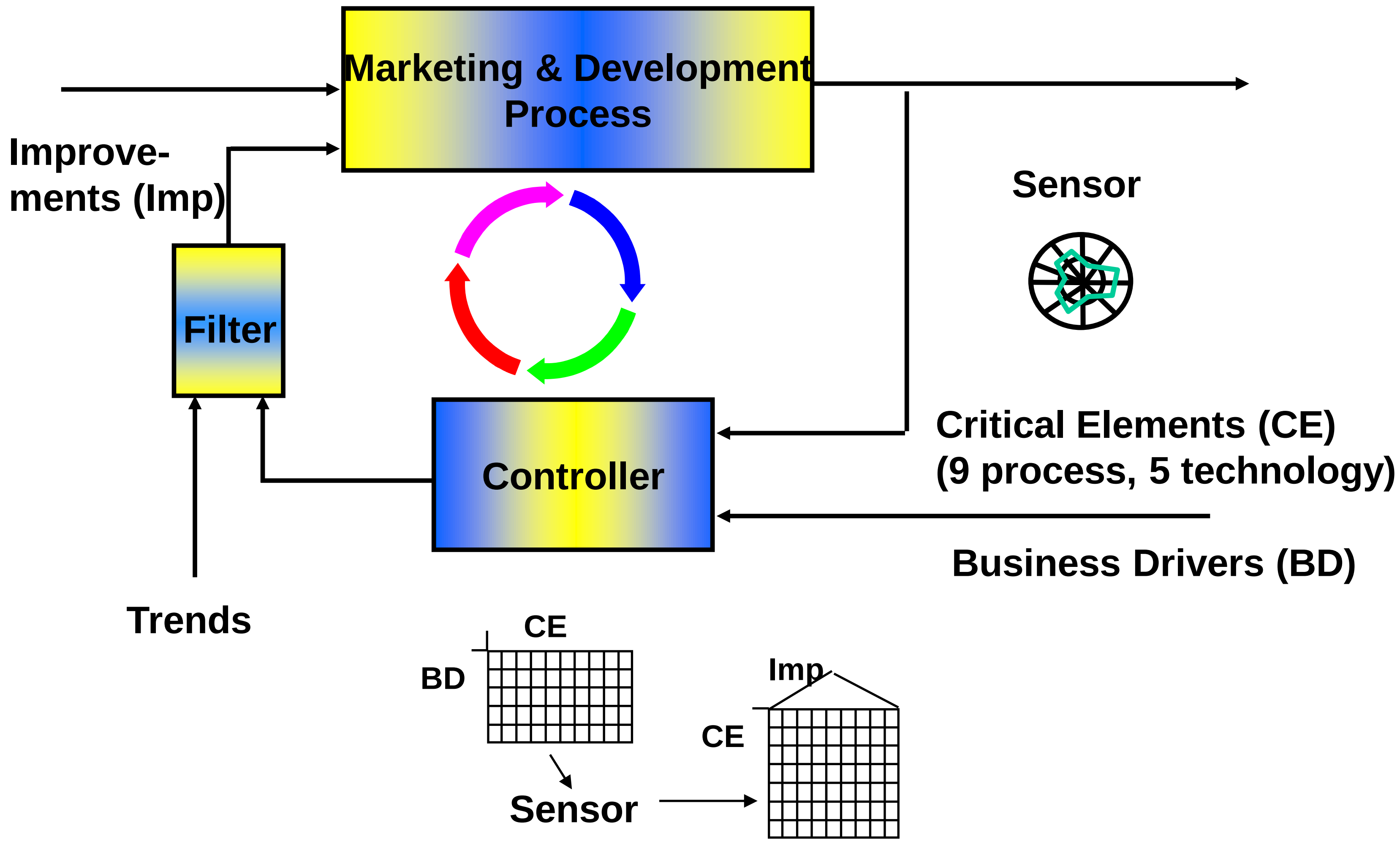
Capability Maturity Model (software development process)

Ericsson Rijen
CMM L3 since 95

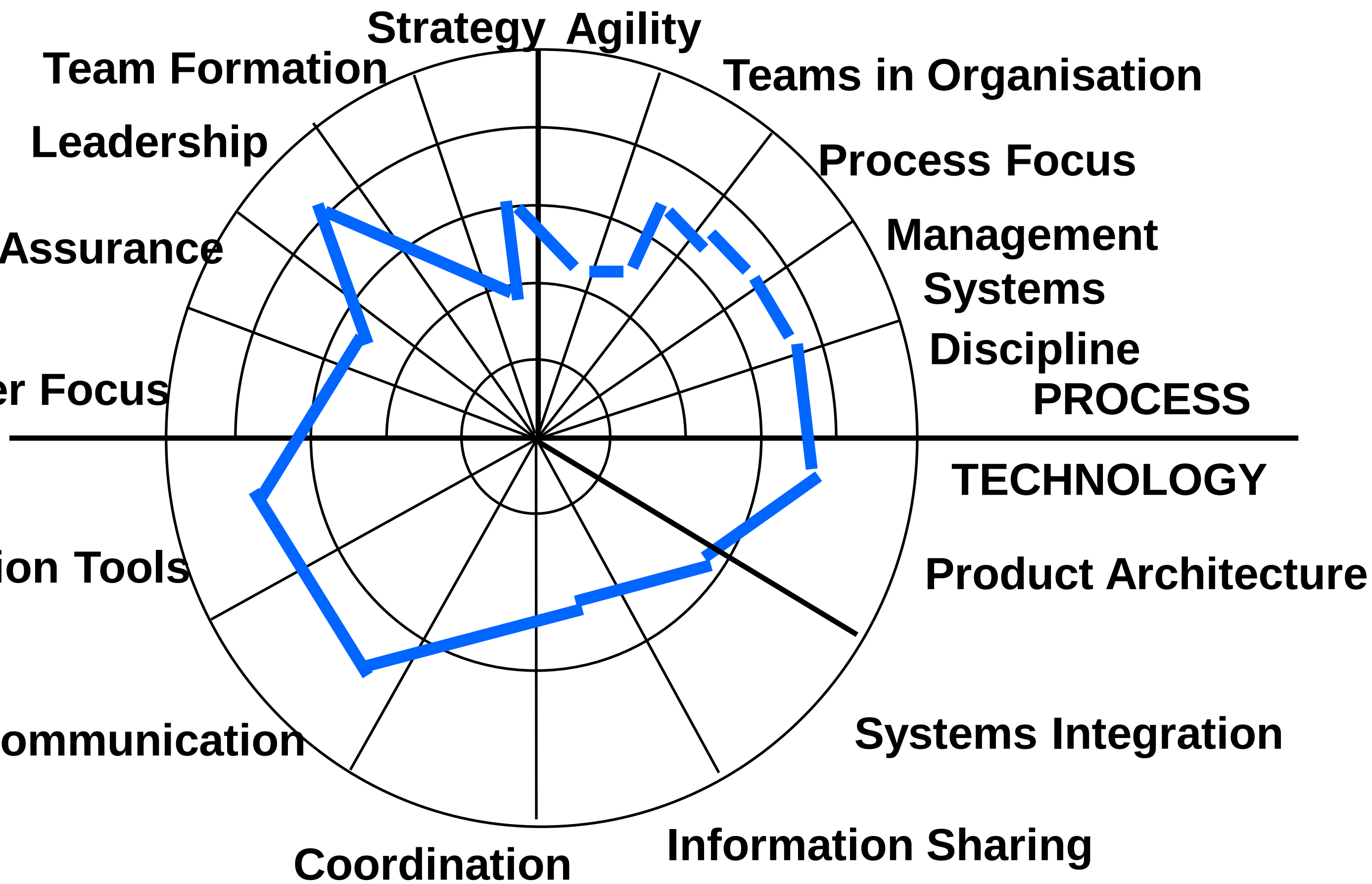
Ericsson Enschede
CMM L2 since 98



BRACE structured change management



BRACE model Process-Technology



From: BRACE questionnaire

Product Architecture Support

2. Are the restrictions that apply to realization of the functional requirements illustrated?

3. Are interfaces of the subsystems & external influences that can disturb these interfaces modeled available in an early stage?
4. Is the coupling between the subsystems provided as a reference in the information system?

5. Is the robustness of key interfaces ensured by the information system in use?

6. Can the information system suggest components for reuse during the development of a new product?

7. Are the down-stream consequences of choices concerning reuse that are made during development communicated automatically to the involved disciplines?
8. Are relations between components and interfaces coordinated by information systems?

9. Is the generic product family model electronically available to all disciplines?

10. Are dedicated software modules used for description of the product's interfaces?

11. Are discrepancies between modules identified automatically?

12. Does a workflow management system ensure the product, its modules and interfaces are developed according to its decomposition sequence?
13. Can suppliers view design information concerning the parts of the product that influence their product development process?

A product architecture is used to define the relationship between requirements and product specifications at a certain abstraction level **that is useful to an organization.** The usefulness is determined by four major aspects:

1. Complexity Reduction.

Components or subsystems can be designed relatively independently, with reference to the interface only and not to the whole product or system.

2. Reuse.

Product architectures enable **reuse of components across product families**, enabling commonality of components in contemporary products. Furthermore, product architectures can be employed to achieve reuse of components in future products.

3. Project Organization.

As product architectures illustrate the relationships between components, they indicate which issues need to be discussed in the development team. The project can be **organized around the product architecture, with high concurrency among the development of the components.**

4. Product Strategy.

Finally, the product architecture can be used to **define components with added value.** The design team should then outsource the components that have little added value. The product could also interface with an environment that consists of standard components. This way, future improvements in the price performance ratio of these components are automatically incorporated.

Architectures - conclusion

Architectures to

- **reduce complexity**
  hierarchy vs distributed
  advantages/dis-advantages

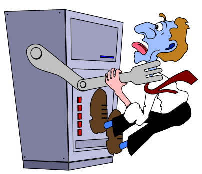
- **re-use**
  of modules with fixed interfaces
  interfaces: command request, status indication, ...

- **project support (sub projects)**
  one common backplane “bus”
  plus independent modules developed
  in parallel or sequential in time

- **product strategy (compete on interfaces)**
  “computerless computer company” article

The Computerless Computer Company

HBR, Jul-Aug, 1991

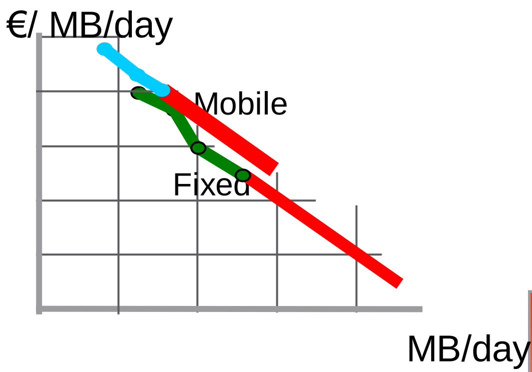


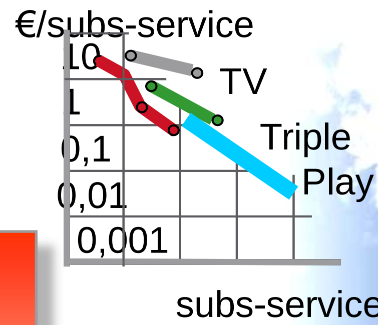
1975-1985	1985-1995	Apple	Microsoft
	SOFTWARE IS SCARCE	**Apple**	**Microsoft**
			BIOS
HARDWARE IS COSTS	**HARDWARE IS CHEAP**		**IBM, Compaq, Taiwan, ...**

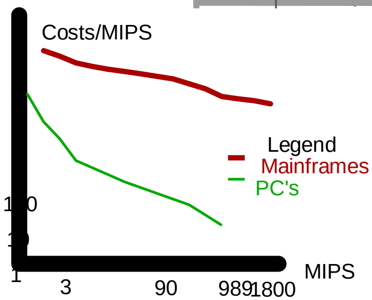
1975-1985: Hardware Decade

1985-1995: Software Decade

Replacing:
- typewriter by a text editor
- calculator by a spreadsheet







Hardw. (IBM)

Softw. (Micro-Soft)

Hardw.

Comm. (KPN)

Softw.

Hardw.

Appli. & Services (You !!)

Death of distance

Open Source

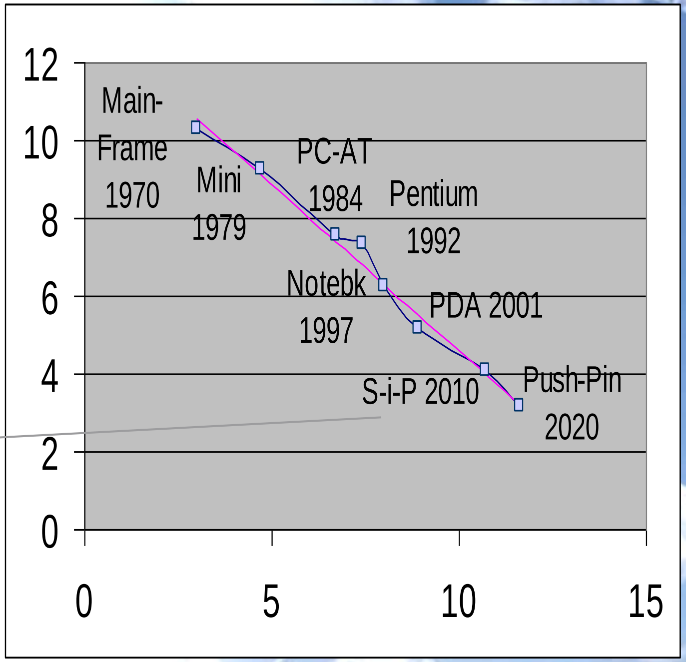
Micro-systems

70-80

80-90

90-2000

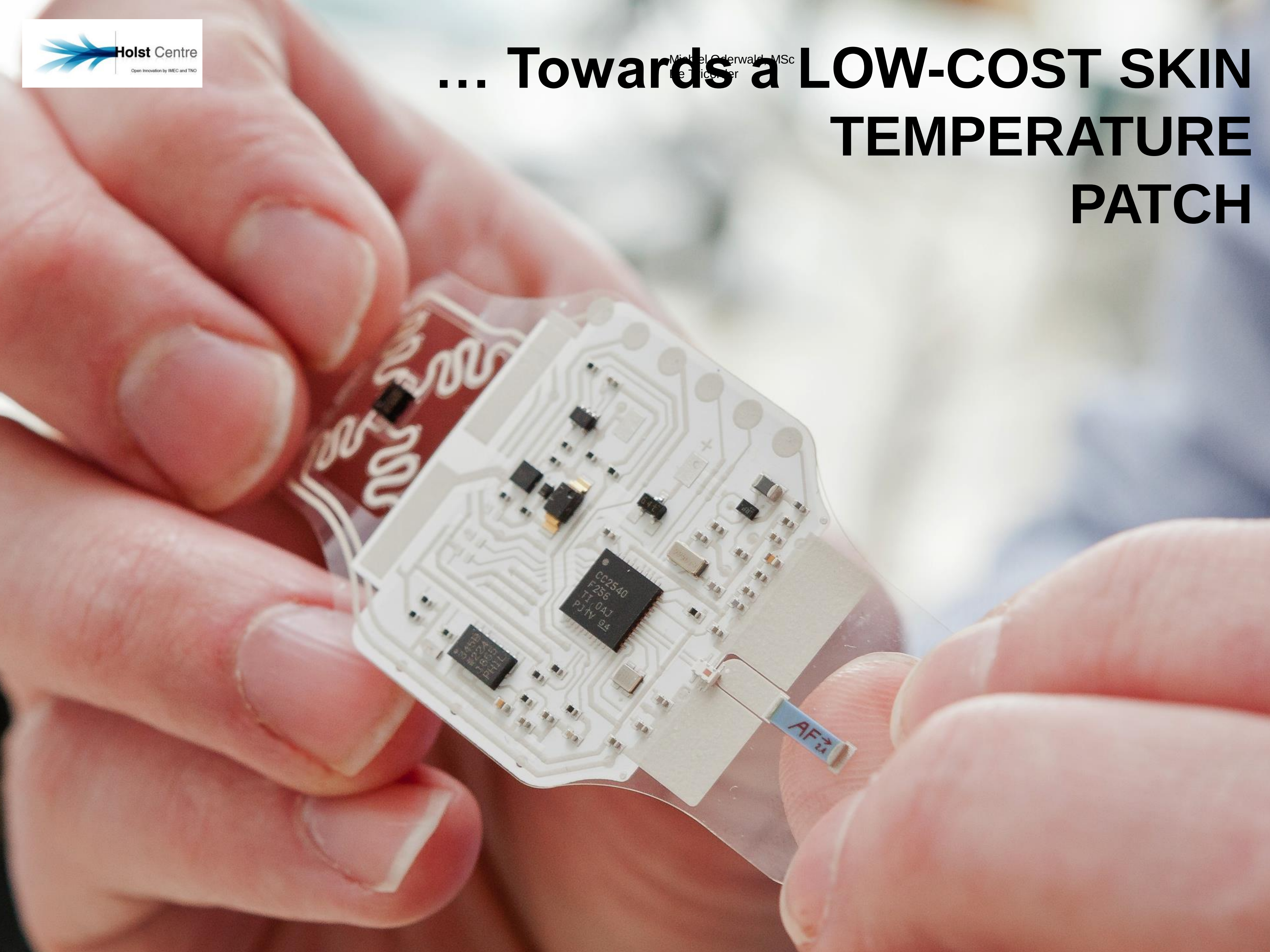
2000-2020



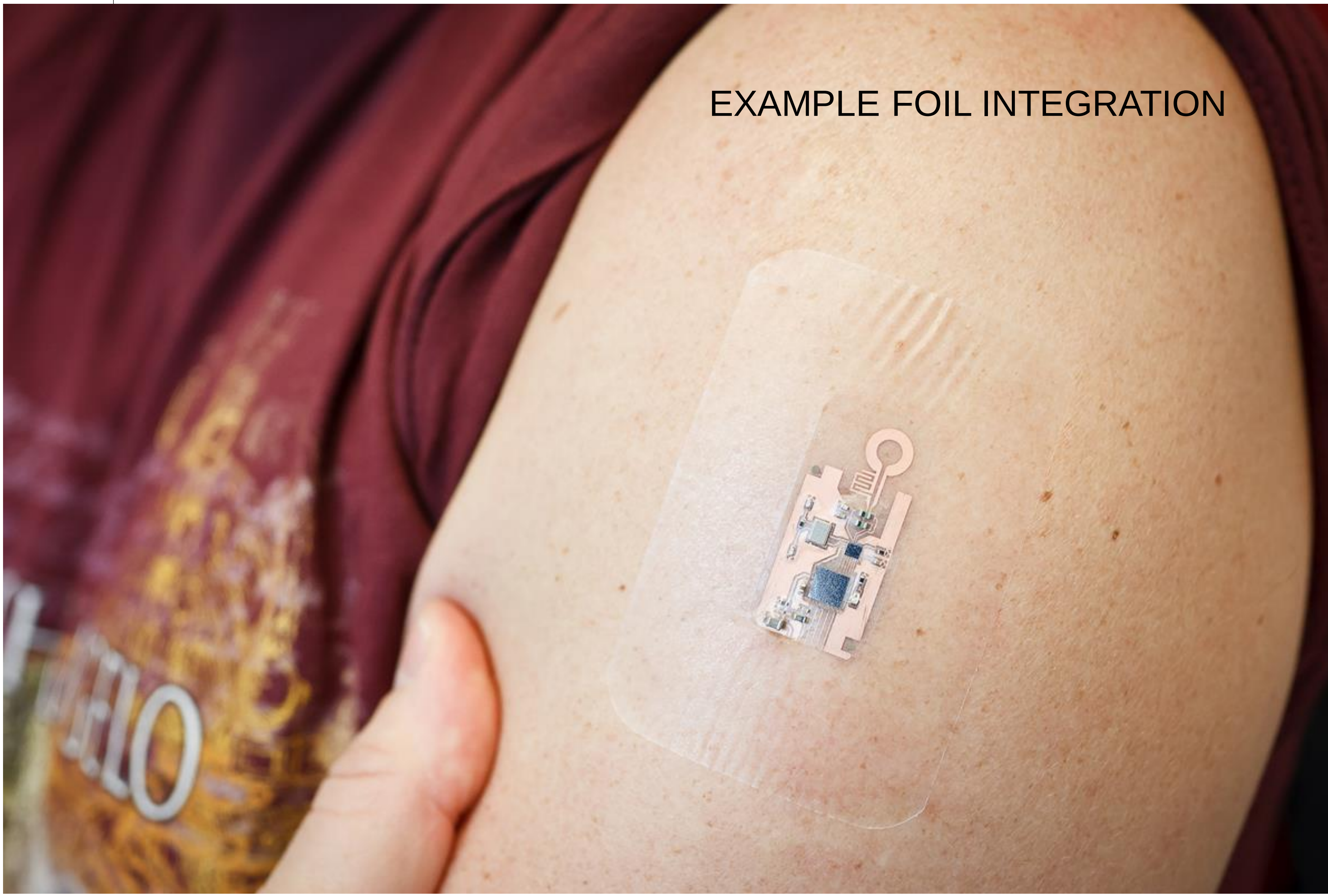
Holst Centre
Open Innovation by IMEC and TNO

… Towards a LOW-COST SKIN TEMPERATURE PATCH

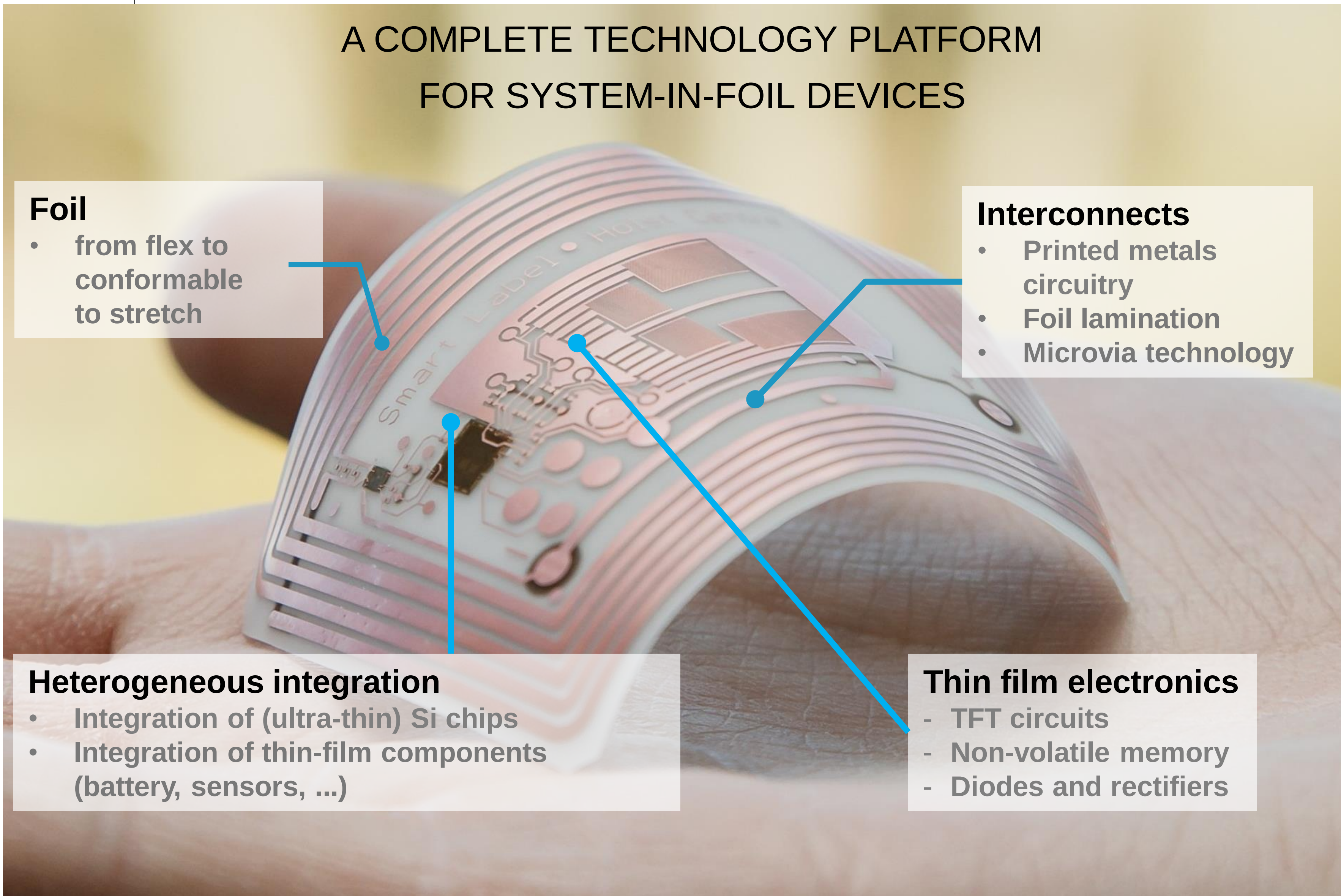
Michel Oderwald, MSc
De Tricorder



EXAMPLE FOIL INTEGRATION



A COMPLETE TECHNOLOGY PLATFORM FOR SYSTEM-IN-FOIL DEVICES



Foil
- from flex to conformable to stretch

Interconnects
- Printed metals circuitry
- Foil lamination
- Microvia technology

Heterogeneous integration
- Integration of (ultra-thin) Si chips
- Integration of thin-film components (battery, sensors, ...)

Thin film electronics
- TFT circuits
- Non-volatile memory
- Diodes and rectifiers

Van 1998 tot 2014

- Lou Feijs (1998-2001)
- Martin Rem – EESI became ESI (Ericsson stapt uit,
  - assessment of ESI by Patrick Dewilde – focus on embedded in equipment (1m^3), no on chips
- Ed Brinksma
- Boudewijn Haverkort
- Frans Beenker (TNO-ESI)

TNO 2014

- Vroeger cooperatieve research (1932-1960)
  - Veel instituten, bijna per branch (hout, lederwaren, etc.)
- 1960-2010 van forse rijksbijdrage (75%) naar meer competitieve subsidies en later meer en meer bedrijfsbijdrage (B2B, testen) (40% rijk, 30% competatief, 30% bedrijf)
- 2010-
  - Testen en repeat afgestoten, rijksbijdrage nog verder naar beneden en in NL geen innovatie subsidies (25% rijks, 10% TKI, 15% EU, 40% B e.g.)
  - Forse focus op europese, nu H2020 subsidies en
  - Shared Research Programma
    - Holst, ESI, maar nu ook Solliance, Snellius, van't Hoff, DITCM, ..
      - In feite cooperative research, maar nu met EU en TKI funding
  - SRP's: 5M+/year, own board, own branding/way of working, within TNO.

So, my question is what has changed in 15 years?