ARTES

Summer School 2005

August 15-19, 2005

Högskolan i Skövde

www.artes.uu.se


ARTES
SUMMER SCHOOL 2005

Programme

- **August 15, ARTES Tutorials of special industry interest**
  - 8.00 Registration in building G at room 110.
  - 9.15 Tutorial: *Autonomic Computing for Real-Time Systems*, Mike Hinchey, NASA Goddard Space Flight Center, USA
  - 10.00 Break
  - 10.15 Tutorial continues.
  - 12 Lunch
  - 13.15 Tutorial: *Real-Time Issues in Wireless Sensor Networks*, Chenyang Lu, Washington University in St. Louis, USA
  - 15.15 Break
  - 15.30 ARTES Real-Time doctors presentations
    - 15.30 *Reliable Avionics*, Kristina Forsberg, Saab Avitronics
    - 16.00 Håkan Sivencrona, Mecel AB
  - 17.00 Time to put up posters.
  - 19.00 Dinner at Scandic Billingen, Trädgårdsgatan 10.
- **August 16-17, Real-Time in Sweden 2005**
  - 8*th* biennial SNART conference on real-time systems
- **August 18**
  - 8.15 Tutorial: *Componenent-based Software Engineering & Design Exploration*, Michel Chaudron, Eindhoven University of Technology, The Netherlands.
  - 10.00 Break
  - 10.15 Tutorial: *Distribution and fault tolerance, two sides of the same coin*, Jan Lindblad, System Architect, Enea Embedded Technology, Sweden.
  - 12.00 Lunch
  - 13.15 Tutorial: *Execution time analysis, can it be done? How to do?* Jan Lindblad, Enea Embedded Technology, Sweden.
  - 15.00 Break
  - 15-23 ARTES social activity at Läkö castel, boat trip to and dinner at Navens light house.
- **August 19**
  - 8.15 Tutorial: *Experimental sensor networking research*, Thiemo Voigt and Joakim Eriksson, Swedish Institute of Computer Science, Sweden.
  - 10.00 Break
  - 10.15 Talk: *The future for IT*, Karl-Einar Sjödin, Vinnovas unit for information and communication research, participated in the report "Inspiration till Innovation".
  - 12.00 Lunch, end of summerschool.

Contents

The 9:th ARTES summerschool

ARTES++

ARTES++ is a Swedish national graduate school in real-time and embedded systems, supported by Swedish Foundation for Strategic Research (SSF). The school is planned to operate from 2004 until 2007 with 20 students annually. Today is 36 students provided support for international mobility, industrial stay, and attending courses. Other activities include courses, an annual summer school and an annual graduate student conference.

Courses fall 2005

- **Advanced Real-Time Scheduling**, HT'05, 5p.
  Location: MdH, Västerås.
- **Forskningsplanering**, HT'05, 3p.
  Location: 1*st* workshop in conjunction with ARTES Summerschool in Skövde, then at MdH in Västerås. Time plan: 20/8, 26-27/8, and 14-15/11.
- **Hardware/Software Codesign**, HT'05, 5p.
  Location: LIU, Linköping.
- **Introduction to Systems Thinking and its Application**, HT'05, 5p.
  Location: University of Skövde. Time plan: 6-8/9 and 18-19/10.
- **Real Time Communication**, HT'05, 5p.
  Location: Halmstad University. Time plan: Sept. 13-14, Oct. 4-5, and Nov. 1-2

ARTES graduate student conference in spring 2006

Calls to appear

* Courses for 2006
* Invitation to Ph.D. students to apply to the ARTES++ programme for 2006

Applications and questions

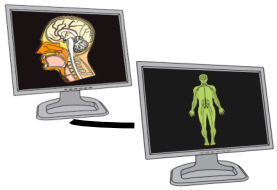
should be sent by e-mail to **info@artes.uu.se**

Send letters to

ARTES
Box 337
SE 75105 Uppsala
Sweden

Participants

Markus	Adolfsson	Högskolan i Halmstad	markus@x3-c.com
Mehdi	Amirijoo	Linköpings universitet	meham@ida.liu.se
Martin	Andersson	Lunds universitet	martin.andersson@control.lth.se
Anita	Andler	ARTES, Skövde	anita.andler@usa.net
Raul	Barbosa	Chalmers Tekniska Högskola	rbarbosa@ce.chalmers.se
Carl	Bergenhem	SP Electronics	Carl.Bergenhem@sp.se
Marcus	Brohede	Högskolan i Skövde	marcus.brohede@his.se
Vicente	Casanova	Technical University of Valencia	vcasanov@isa.upv.es
Michel	Chaudron	Eindhoven University of Technology	m.r.v.chaudron@TUE.nl
Sigrid	Eldh	Mälardalens högskola	sigrid.eldh@mdh.se
AnnMarie	Ericsson	Högskolan i Skövde	annmarie.ericsson@his.se
Johan	Erikson	Mälardalens högskola	johan.erikson@mdh.se
Joakim	Eriksson	SICS	joakime@sics.se
Xing	Fan	Högskolan i Halmstad	xing.fan@ide.hh.se
Elena	Fersman	Ericsson AB	elenaf@it.uu.se
Kristina	Forsberg	Saab Avitronics	kristina.forsberg@saabtech.se
Roland	Grönroos	ARTES, Uppsala	Roland.Gronroos@it.uu.se
Thomas	Gustafsson	Linköpings universitet	thogu@ida.liu.se
Sanny	Gustavsson	Högskolan i Skövde	sanny.gustavsson@ida.his.se
Mike	Hinchey	NASA Goddard Space Flight Center	mike.hinchey@usa.net
John	Håkansson	Uppsala universitet	johnh@it.uu.se
Viacheslav	Izosimov	Linköpings universitet	viaiz@ida.liu.se
Najeem	Lawal	Mittuniversitetet	Najeem.Lawal@miun.se
Niklas	Lepistö	Mittuniversitetet	Niklas.Lepisto@miun.se
Jan	Lindblad	Enea Embedded Technology	jan.lindblad@enea.se
Birgitta	Lindström	Högskolan i Skövde	birgitta.lindstrom@his.se
Chenyang	Lu	Washington University in St. Louis	lu@cse.wustl.edu
Gunnar	Mathiason	Högskolan i Skövde	gunnar.mathiason@his.se
Leonid	Mokrushin	Uppsala universitet	leom@it.uu.se
Robert	Nilsson	Högskolan i Skövde	nilo@iki.his.se
Susanna	Nordström	Mälardalens högskola	susanna.nordstrom@realfast.se
Anders	Pettersson	Mälardalens högskola	anders.pettersson@mdh.se
Paul	Pettersson	ARTES, Uppsala	paupet@it.uu.se
Anil	Reddy	Jönköping University	anilk_reddy3@yahoo.co.in
Jianlin	Shi	KTH	jianlin@md.kth.se
Håkan	Sivencrona	Mecel AB	hakan.sivencrona@mecel.se
Karl-Einar	Sjödin	VINNOVA	Karl-Einar.Sjodin@vinnova.se
Aleksandra	Tesanovic	Linköpings universitet	alete@ida.liu.se
Fredrik	Törner	Volvo Car Corporation	ftorner@volvocars.com
Martin	Törngren	KTH	martin@md.kth.se
Thiemo	Voigt	SICS	thiemo@sics.se

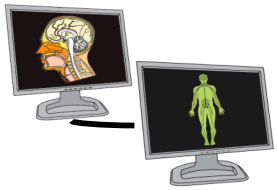


Tutorial
Autonomic Computing in Real–Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

Overview

1. Motivation for Autonomic Computing in Real-Time Systems
2. What is Autonomic Computing?
3. Self-* properties
4. Autonomic Managers and Autonomic Elements
5. New and emerging biological metaphors
6. Examples from NASA Sensor Network applications
7. Another Example: HRSM



(1) Motivation for Autonomic Computing

Tutorial
Autonomic Computing in Real-Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

The goals of the IT industry over the last 20 years:

Goal #1: Improve performance

Goal #2: Improve performance

Goal #3: Improve cost-performance

What big crisis is likely to hit the computer industry ?

The main concern to further progress in the information technology industry:

That Moore's law cannot continue to hold

i.e. that processing power will double every 18–24 months

Breathtaking Statistics

- Price to Performance ratio doubles every 18 months ≈ 100 fold increase per decade.
- Performance improvements in the next 18 months = all performance improvements to date.
- New processing power = sum of all previous processing power.
- New storage = sum of all available storage *ever*.

J.N. Gray, Turing Award Lecture, 1999

Problem

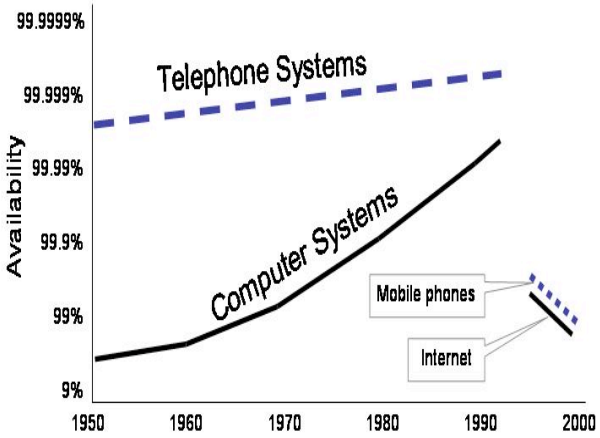
- Assumptions of continuous improvements in system development, that meet our ever more complex requirements.
- Pervasiveness of software, and our growing reliance on it.

Flawed Assumptions

- Human beings can achieve perfection; they can avoid making mistakes during installation, maintenance and upgrades.
- Software will eventually be *bug free*; the emphasis of industry has been to hire better programmers; the emphasis of academia to train better Software Engineers
- MTBF is large (>100 years) and will continue to grow.
- Maintenance costs are a function of hardware costs.

Patterson and Brown, 2001

Software Lags behind Hardware



The real crisis …

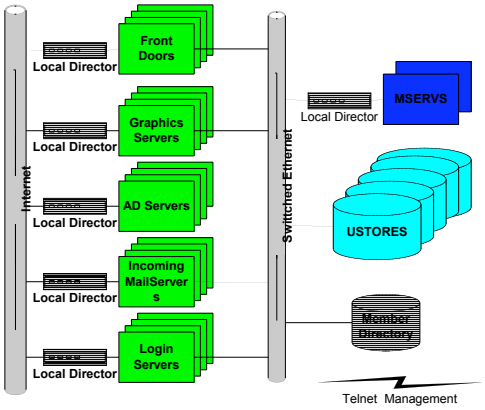
Rather, it is the IT industry's exploitation of the technologies in accordance with Moore's law that has led us to the verge of a complexity crisis…

By 2010, around 200 million IT workers – the working population of the USA – will be required to keep computer systems running.

The problem is complexity …

A Schematic of HotMail

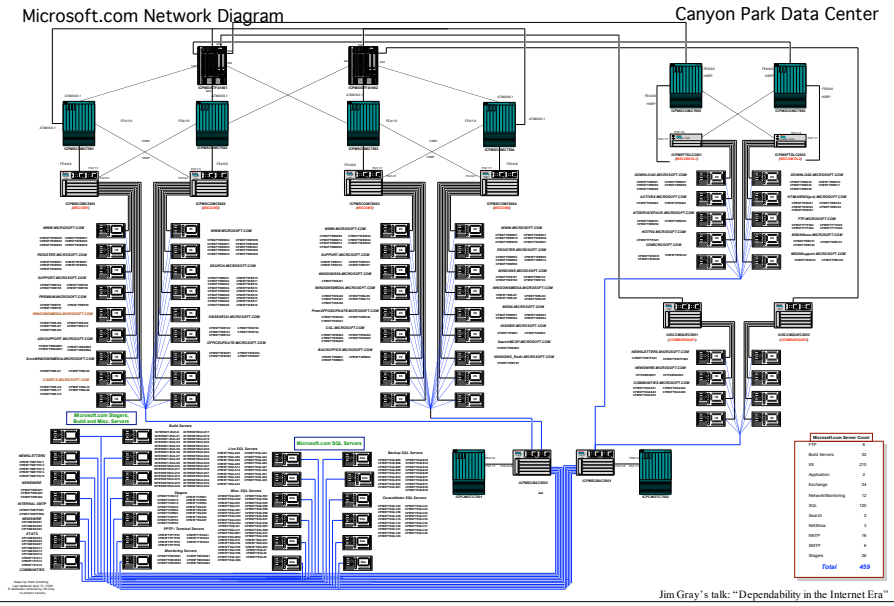
- ~7,000 servers
- 100 backend stores with 120TB (cooked)
- 3 data centers
- Links to
  - Passport
  - Ad-rotator
  - Internet Mail gateways
  - …
- ~ 1B messages per day
- 150M mailboxes, 100M active
- ~400,000 new per day.



Jim Gray's talk: "Dependability in the Internet Era"

The problem is complexity …

One of the Data Centers (500 servers)



Jim Gray's talk: "Dependability in the Internet Era"

Complexity

Any intelligent fool can make things bigger and more complex … It takes a touch of genius and a lot of courage to move in the opposite direction.

Albert Einstein

Conquering Complexity

Today, “complexity” is a word that is much in fashion We have learned very well that many of the systems that we are trying to deal with in our contemporary science and engineering are very complex indeed. They are so complex that it is not obvious that the powerful tricks and procedures that served us for four centuries or more in the development of modern science and engineering will enable us to understand and deal with them...

Herbert A. Simon

Conquering Complexity

… We are learning that we need a science of complex systems and we are beginning to develop it.

Herbert A. Simon

Please raise your hand if you ever…

• trouble connecting to a wired or a wireless network at another work location, hotel or switching from home to work (even with DHCP)?,

• lost a working connection and shouted across the office has anyone else's network connection gone?,

• have gone into the IP settings area in Windows and been confused about the correct settings?,

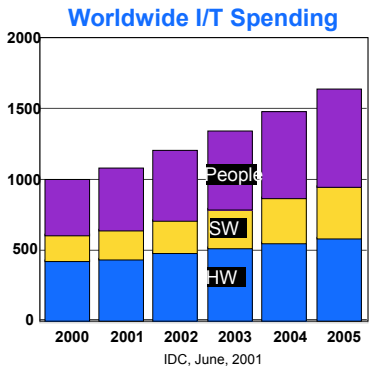
• had a PC which would no longer booted and needed major repair or re-installation of the OS?,

• had a hard-disk crash?

When you consider how many users could honestly be back up and running (with all applications and data) in less than a days work after a hard disk crash or could completely migrate to a new PC in less than a day highlights the indirect (often unaccounted) costs of managing personal computing…

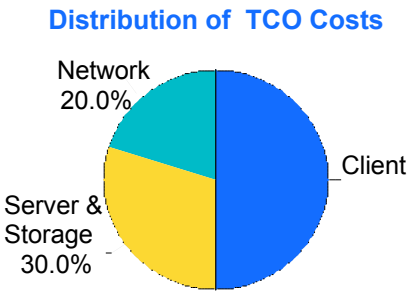
Bantz, Jan 2003

Total Cost of Ownership



IDC, June, 2001

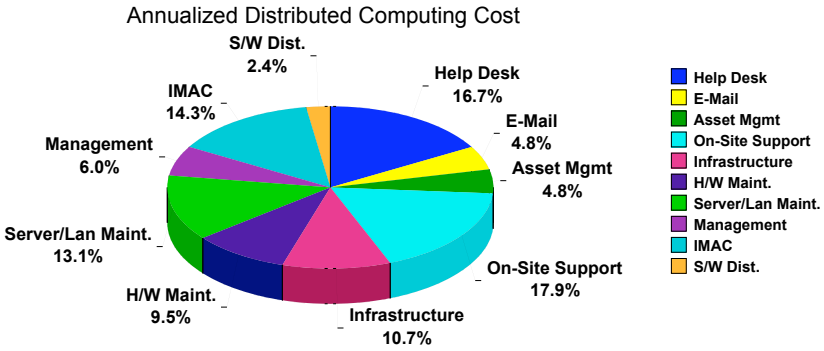
People costs in 2000 is about 1% of the WW economy



Bantz, Jan 2003

Client computing TCO – direct costs

Average cost per desktop seat $100-180/month (org. w/ > 5000 employees)

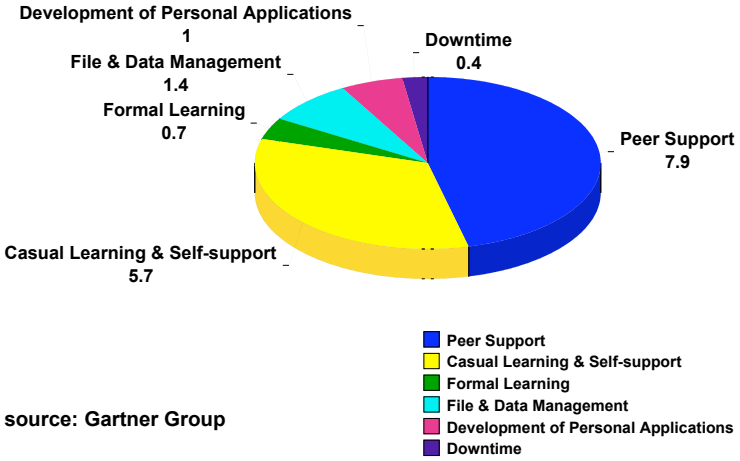


Bantz, Jan 2003

Client computing TCO – indirect costs

Indirect Costs of client computing (in hours/month of end user time)

(205.2 hr./yr @ $50/hr. = $10,260)

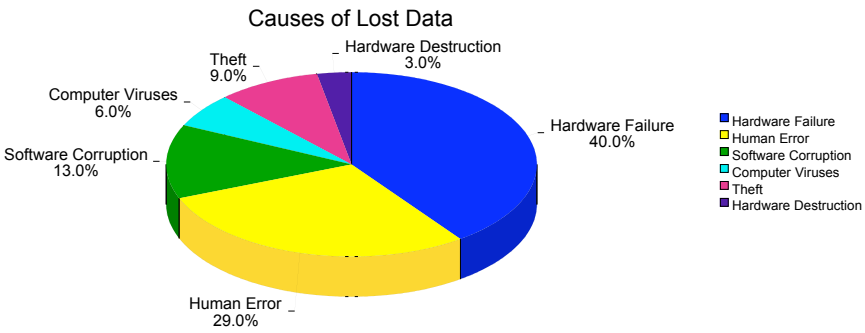


source: Gartner Group

Bantz, Jan 2003

The sources of data loss

Incident rate is approx. 6% of devices per year

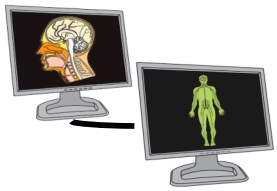


source: "The Cost of Lost Data" by David Smith, Pepperdine Univ.

Bantz, Jan 2003

The Legacy…

- TCO and complexity - the legacy of;
- Downsizing?
- End user computing (EUC)?
- Global Economy?



(2) What is Autonomic Computing

Tutorial
Autonomic Computing in Real-Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

Definitions

au·to·nom·ic (àwtə nómmik)
adj.
Physiology.
- Of, relating to, or controlled by the autonomic nervous system.
- Occurring involuntarily; automatic: *an autonomic reflex.*
- Resulting from internal stimuli; spontaneous.

au·ton·o·mic·i·ty (àwtə nóm i síttee)
n.
- The state of being autonomic.

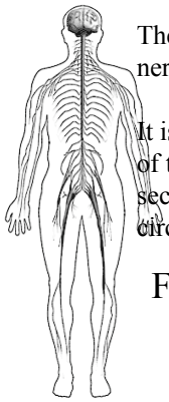
Definitions (2)

au·ton·o·mous (aw tónnəməs)
adj.
- Not controlled by others or by outside forces; independent: *an autonomous judiciary; an autonomous division of a corporate conglomerate.*
- Independent in mind or judgment; self-directed.
  - Independent of the laws of another state or government; self-governing.
  - Of or relating to a self-governing entity: *an autonomous legislature.*
  - Self-governing with respect to local or internal affairs: *an autonomous region of a country.*
- Autonomic.

[From Greek autonomos : auto-, *auto-* + nomos, *law*]

Autonomic Computing – *Biological Inspiration*

Autonomic Computing, launched by IBM in 2001, is emerging as a valuable approach to the design of effective computing systems.



The autonomic concept is inspired by the human body's autonomic nervous system.

It is the part of the nervous system that controls the vegetative functions of the body such as circulation of the blood, intestinal activity and secretion and the production of chemical 'messengers', hormones, that circulate in the blood

Fight or Flight

sympathetic (SyNS)

Rest and Digest

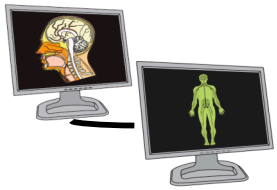
parasympathetic (PaNS)

Computer Software Systems are Not Self-Managing

NASA Satellite Support

Best Case

Worst Case

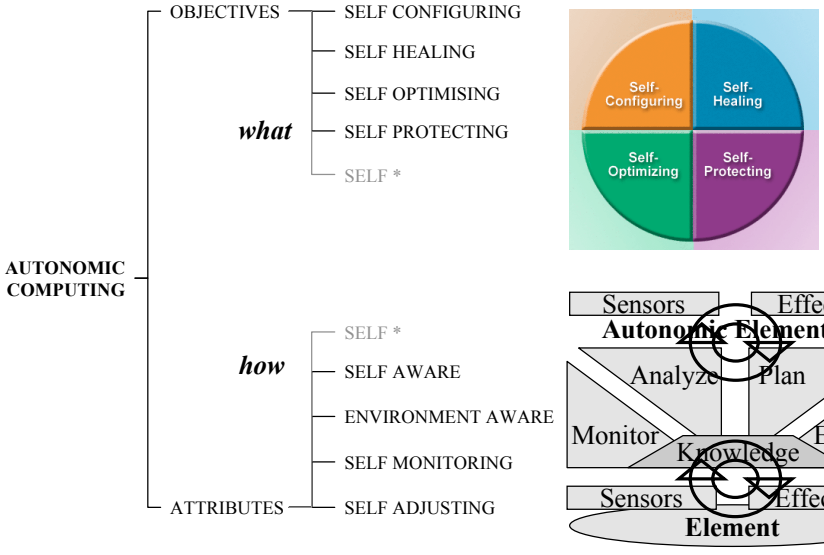


(3)
Self-* properties

Tutorial
Autonomic Computing in Real-Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

Autonomic Systems – *Selfware & self-* properties*



Autonomic Computing – *Other Emerging self-* properties*

- *self-anticipating,*
- *self-adapting,*
- *self-critical,*
- *self-defining,*
- *self-diagnosis,*
- *self-governing,*
- *self-installing*
- *self-organized,*
- *self-recovery,*
- *self-reflecting,*
- *self-simulation*
- *self-stabilizing*
- self-*

Self-managing :
(*wo*)*man in the loop?*

All positive?
- *selfish ☺*

Issue : Trust
Unexpected emergent behaviour
Race conditions etc.

Self- properties*

self-*
Self-managing properties.

self-anticipating
The ability to predict likely outcomes or simulate self-* actions.

self-assembling
Assembly of models, algorithms, agents, robots, etc.; self-assembly is often influenced by nature, such as nest construction in social insects. Also referred to as self-reconfigurable systems.

self-awareness
"Know thy self"; awareness of internal state; knowledge of past states and operating abilities.

self-chop
The initial four (and generic) self-properties (Self-Configuration, Self-Healing, Self-Optimisation and Self-Protection).

self-configuring
The ability to configure and re-configure in order to meet policies/goals.

Self-* properties

self-critical
The ability to consider if policies are being met or goals are being achieved (alternatively, self-reflect)

self-defining
In reference to autonomic event messages between Autonomic Managers: contains data and definition of that data–metadata (for instance using XML).
In reference to goals/policies: defining these (from self-reflection, etc.).

self-governing
As in autonomous: responsibility for achieving goals/tasks.

self-healing
Reactive (self-repair of faults) and Proactive (predicting and preventing faults).

self-installing
As in a specialized form of self-configuration – installing patches, new components, etc or re-installation of OS after major crash.

self-managing
Autonomous, along with responsibility for wider self-* management issues.

Self-* properties

self-optimizing
Optimization of tasks and nodes.

self-organized
Organization of effort/nodes. Particularly used in networks/communications.

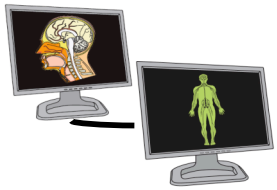
self-protecting
The ability of a system to protect itself.

self-reflecting
The ability to consider if routine and reflex operations of self-* operations are as expected. May involve self-simulation to test scenarios.

self-similar
Self-managing components created from similar components that adapt to a specific task, for instance a self-managing agent.

self-simulation
The ability to generate and test scenarios, without affecting the live system.

selfware
Self-managing software, firmware and hardware.



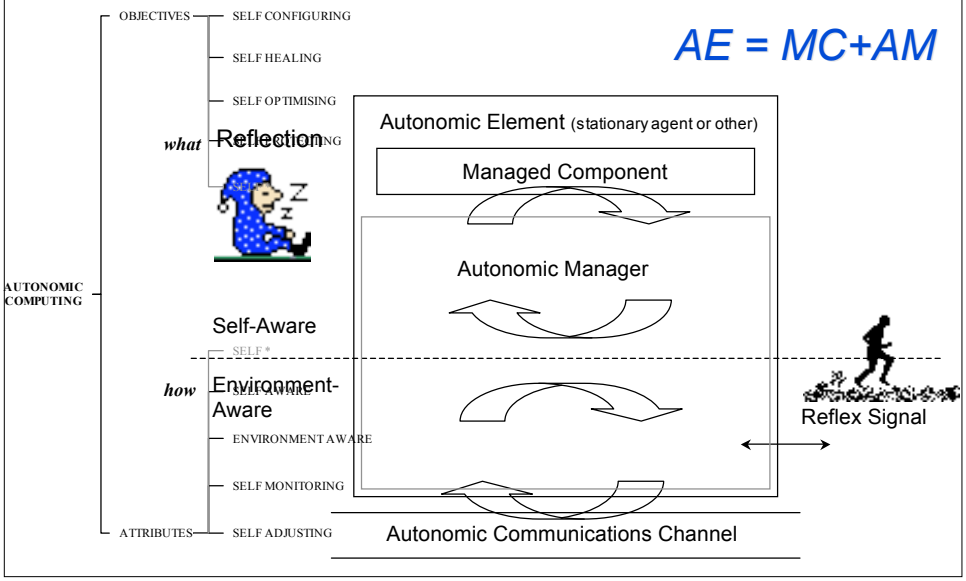
(4)
Autonomic Managers & Autonomic Elements

Tutorial
Autonomic Computing in Real-Time Systems

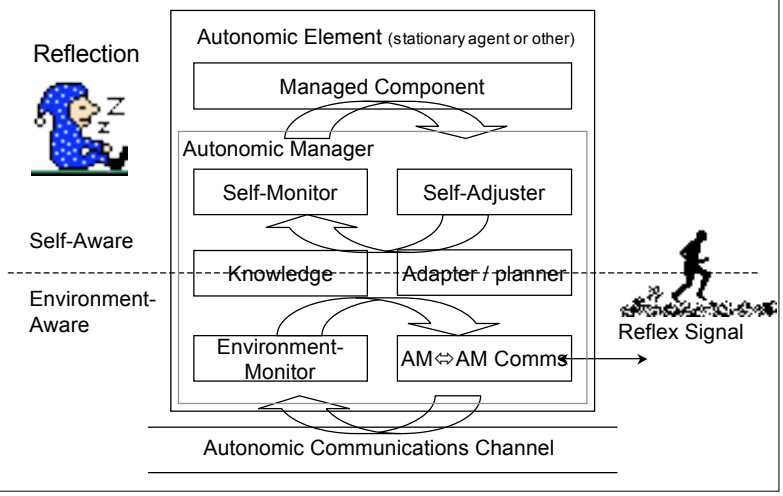
Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

Autonomic Computing – *The Autonomic Element*

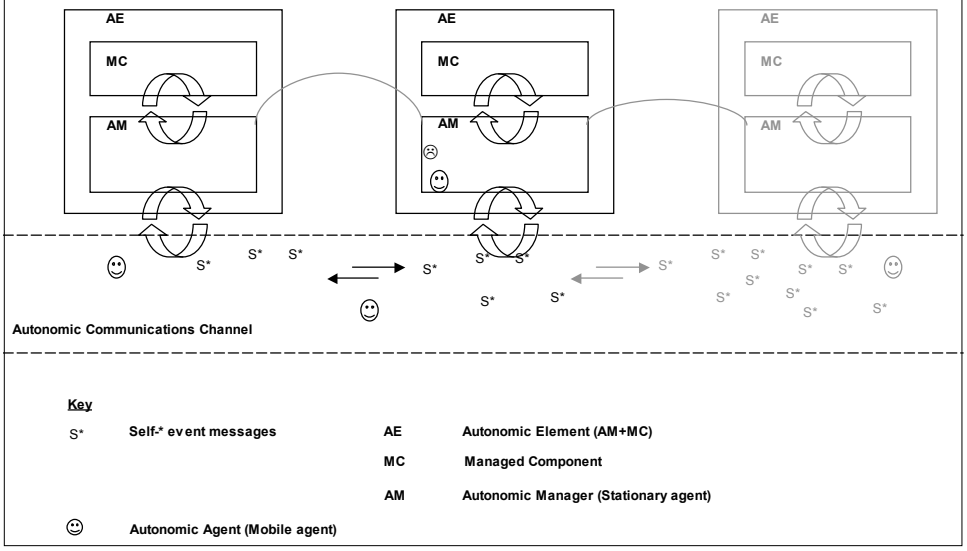
AE = MC+AM



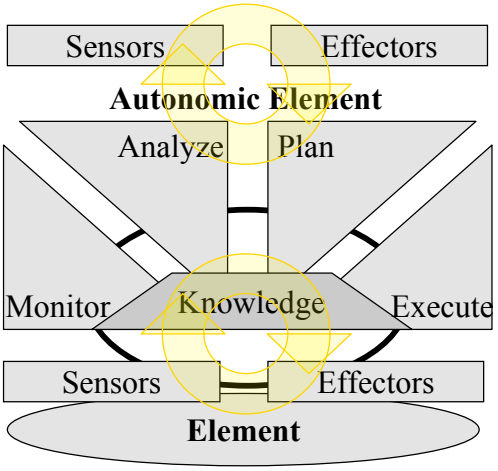
Autonomic Computing – *The Autonomic Manager*



Autonomic Computing – *The Autonomic Environment*



IBM's View

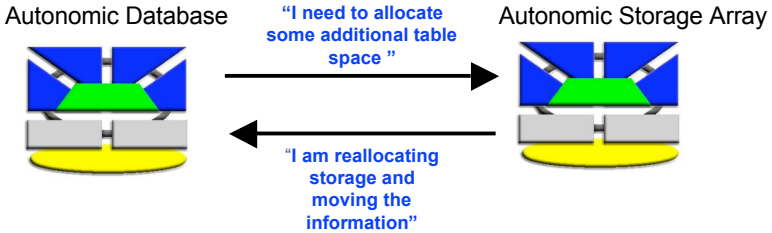


- An autonomic element contains a continuous control loop that monitors activities and takes actions to adjust the system to meet business objectives
- Autonomic elements learn from past experience to build action plans
- Managed elements need to be instrumented consistently

IBM – AC Blueprint 2003 & Ganek, AMS 2003

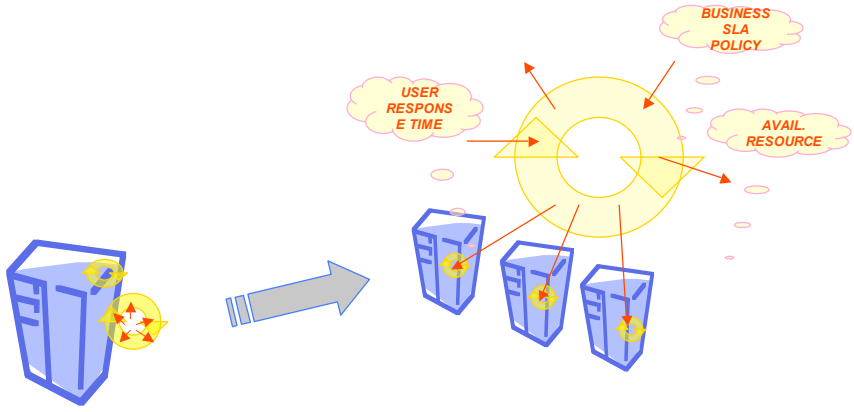
How Do We Make Components Autonomic?

- Autonomic elements have two management tasks
  - They manage themselves
  - They manage their relationships with other elements through negotiated agreements



Ganek, AMS 2003

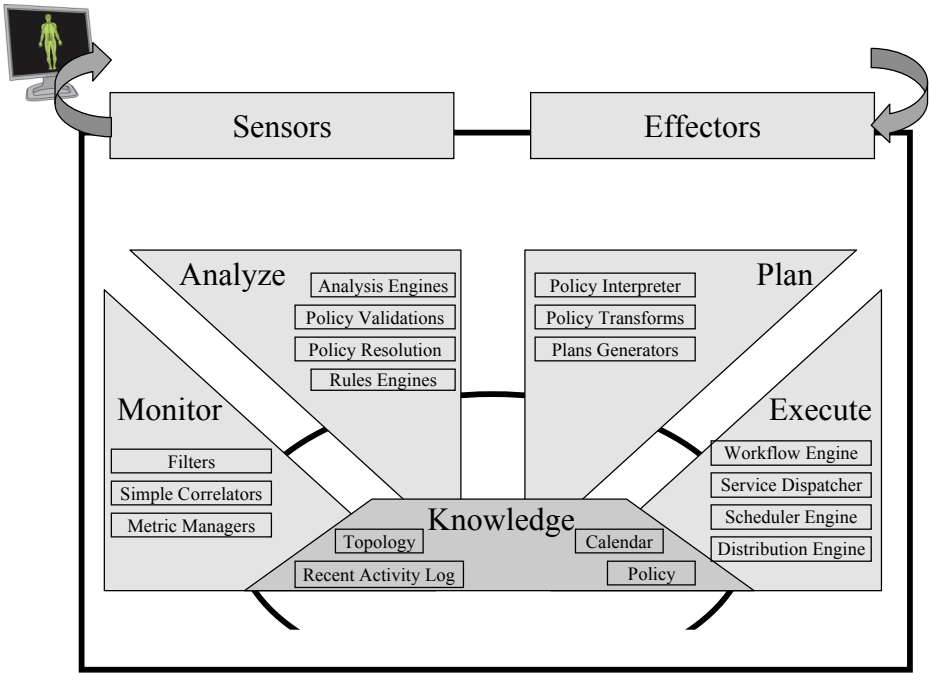
Autonomic control loops: next step evolution



Autonomic features
Local view

Global environment view and knowledge

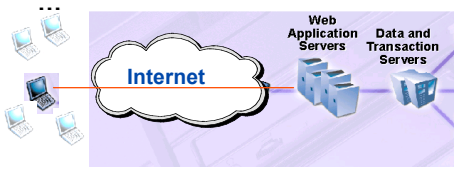
Ganek, AMS 2003



IBM – AC Blueprint 2003

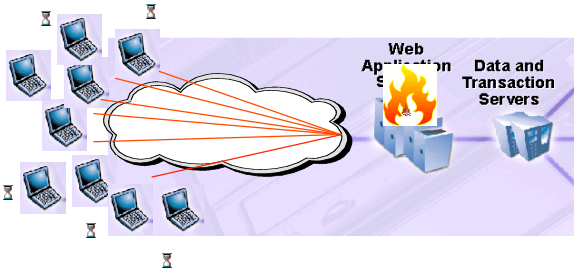
Putting It All Together – An Example

Systems can go from steady state



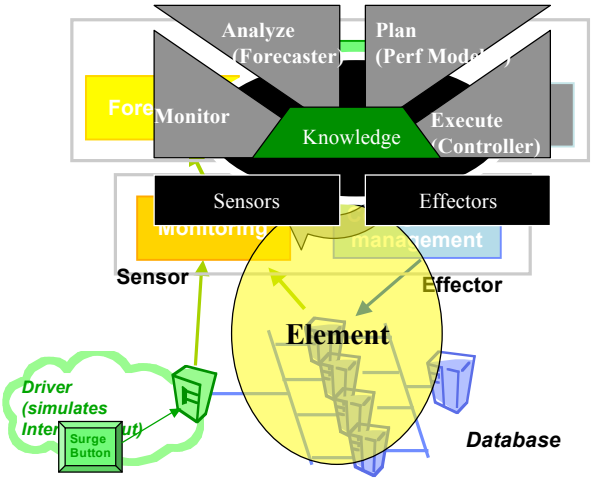
to overloaded without warning

Few minutes later…

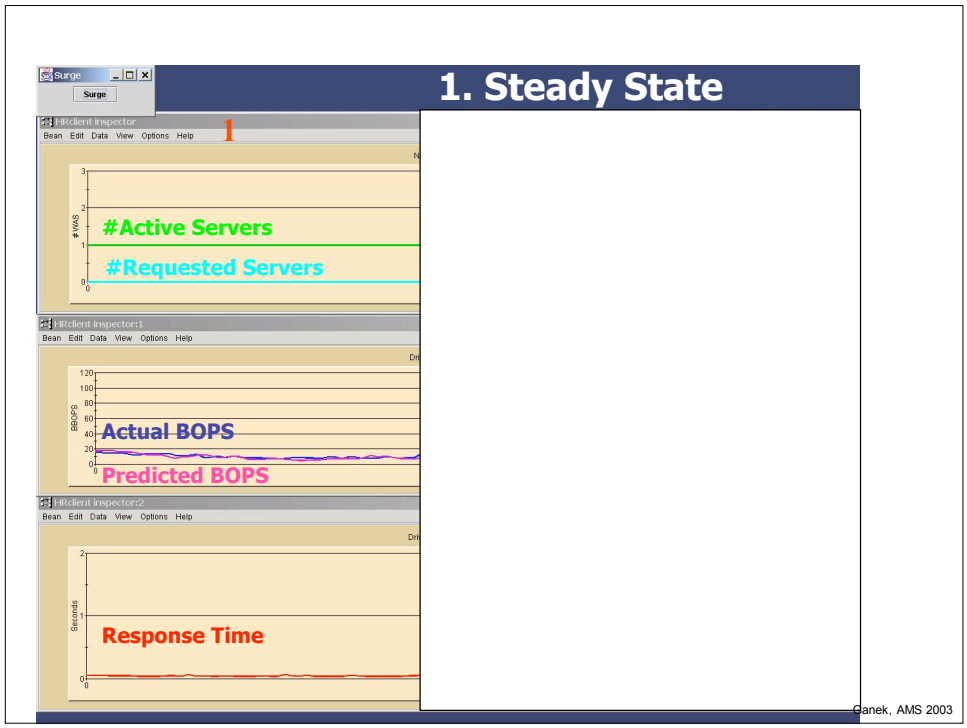


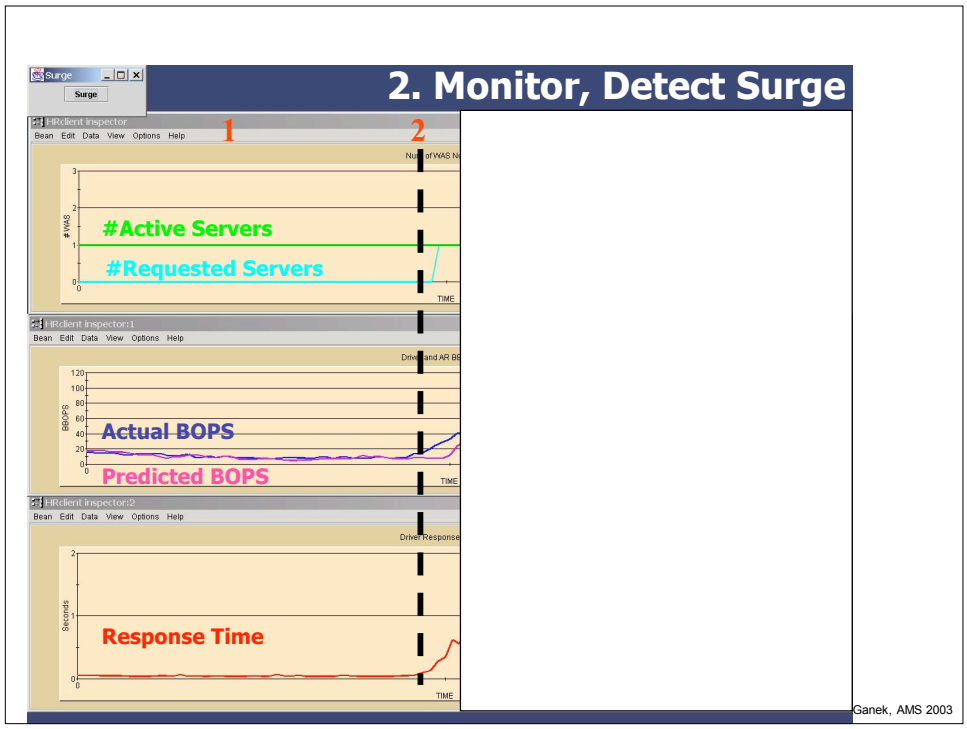
Ganek, AMS 2003

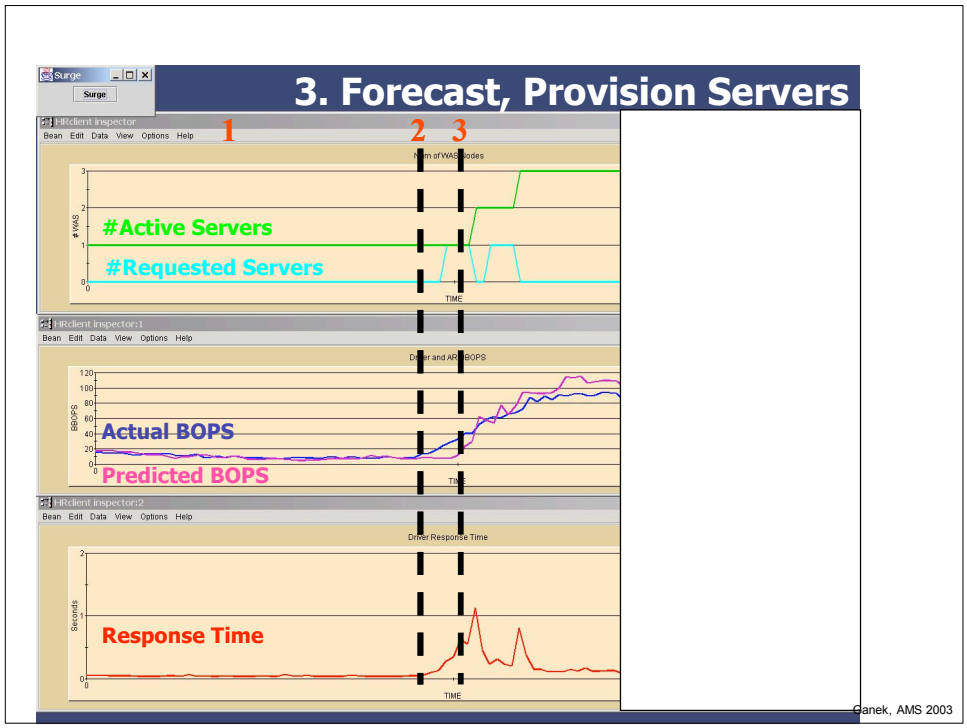
Autonomic Computing: Dynamic Surge Protection

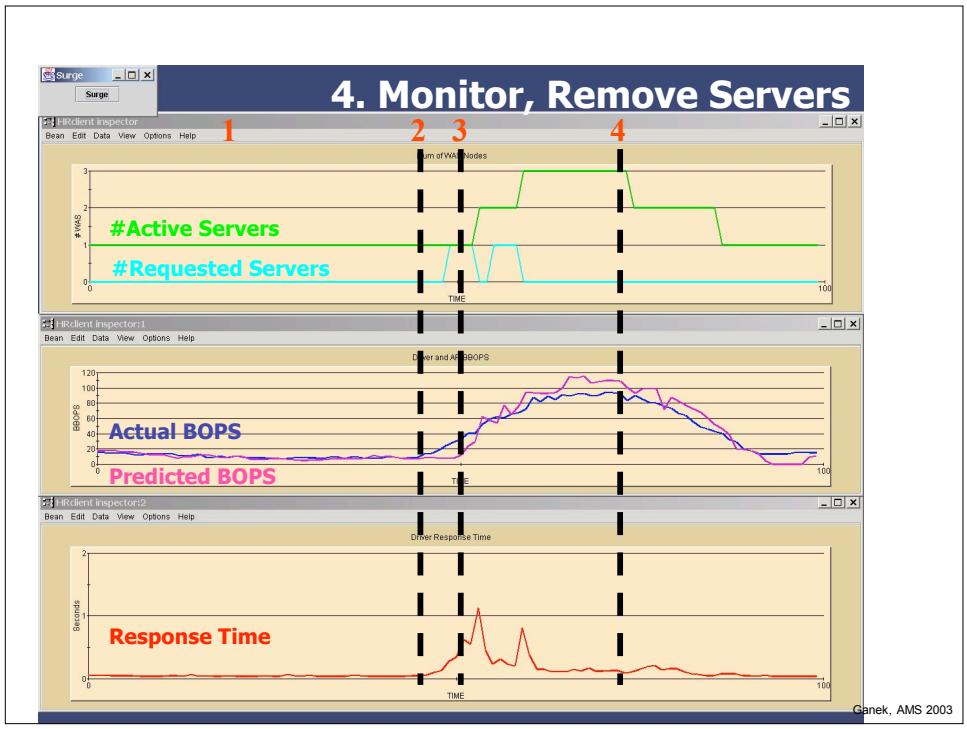


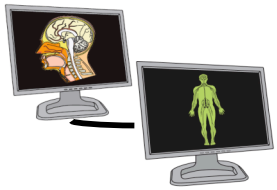
Ganek, AMS 2003


1. Steady State
Surge
HRclient inspector
1
#Active Servers
#Requested Servers
HRclient inspector:1
Actual BOPS
Predicted BOPS
HRclient inspector:2
Response Time
Ganek, AMS 2003


2. Monitor, Detect Surge
Surge
HRclient inspector
1
2
#Active Servers
#Requested Servers
HRclient inspector:1
Actual BOPS
Predicted BOPS
HRclient inspector:2
Response Time
Ganek, AMS 2003


3. Forecast, Provision Servers
1
2
3
#Active Servers
#Requested Servers
Actual BOPS
Predicted BOPS
Response Time
Ganek, AMS 2003


4. Monitor, Remove Servers
1
2
3
4
#Active Servers
#Requested Servers
Actual BOPS
Predicted BOPS
Response Time
Ganek, AMS 2003



(5) New and emerging biological metaphors

Tutorial
Autonomic Computing in Real-Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

Reflex and Healing

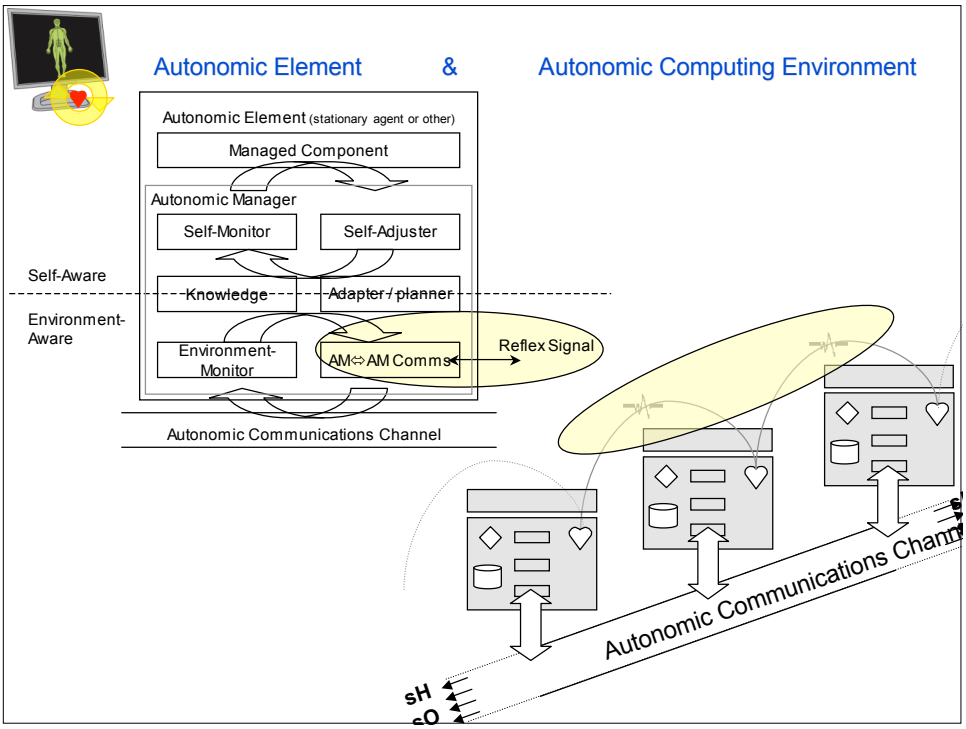
A concept inspired by biological systems is the dual approach of *reflexes* and *healing*. Animals have a reflex system, where the nerve pathways enable rapid response to pain.

Reflexes cause a rapid, involuntary motion, such as when a hot surface is touched.

The effect is that the system reconfigures itself, moving away from the danger to keep the component functioning.

On a much longer timescale, the body will heal itself.
Resources from one part of the system are redirected to rebuild the injured body part, including repair of the reflex response network.

Source: **Bapty et al, ECBS 2003**



Embedded Systems : Heart-Beat Monitoring (HBM)

The typical approach for system management is based on events which are generated and sent under fault or problem conditions.

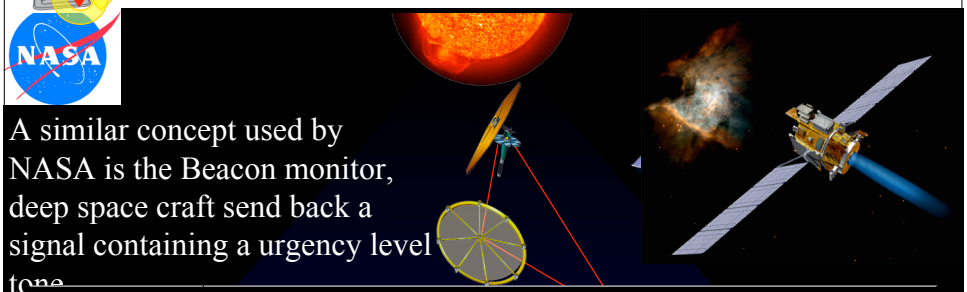
In the embedded system space the opposite is typically the case.

A system management action occurs when something does not occur.

An example is the fault tolerant mechanism of a heartbeat monitor (HBM), through a combination of the hardware (the timer) and software (the heartbeat generator) an 'I am alive' signal is generated periodically to indicate all is well The absence of this signal indicates a fault or problem. Some embedded processors have a hardware timer which, if not periodically reset by software, causes a reset/restart. This allows a particularly blunt, though effective, recovery from a software hang.

This approach offers the advantage that through continuous monitoring problem determination becomes a proactive rather than a reactive process.

NASA's Beacon Monitoring



A similar concept used by NASA is the Beacon monitor, deep space craft send back a signal containing a urgency level tone.

Nominal	All functions as expected no need to downlink.
Interesting	Interesting – non-urgent event. Establish comms when convenient.
Important	Comms need to take place within timeframe or else state could deteriorate.
Urgent	Emergency. A critical component has failed. Cannot recover autonomously and intervention is necessary immediately.
No Tone	Beacon mode is not operating.

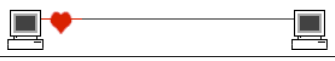
Pulse Monitoring (PBM)

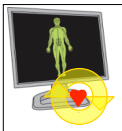
The HBM only informs if a process is alive or dead (assuming comms working)
– not the processes actual health or state of being.

PBM = HBM + Beacon urgency concept

With a *pulse monitor* instead of just checking the presence of a beat, the rate or tone within the beat would also be measured.

This can give a quick measurement as to the state of health of a process.





PBM – a mechanism for autonomicity

The concept of incorporating a pulse monitor to provide

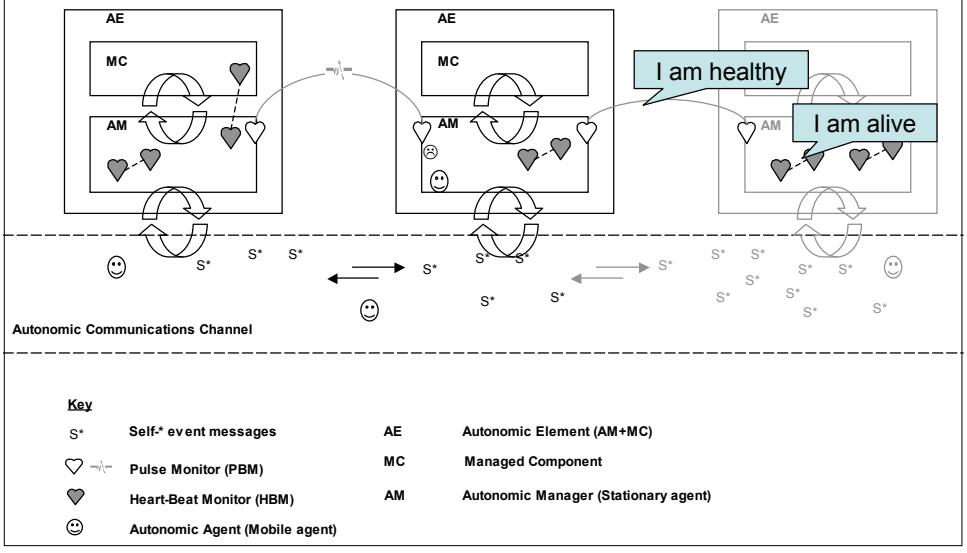
- a reflex reaction for indicating the 'health' based on vital signs

- providing dynamics within autonomic responses and multiple loops of control;

  - some slow and precise (normal event message channel)

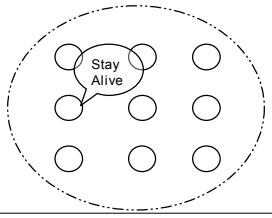
  - others fast and possibly imprecise (PBM)

Autonomic Computing – *The Autonomic Environment*



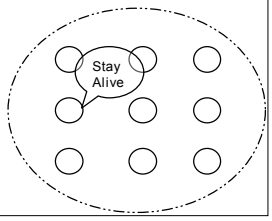
Apoptosis – *returning to Biological metaphors*

- if you cut yourself and it starts bleeding…
- often, the cut will have caused skin cells to be displaced down into muscle tissue
- if they survive and divide, they have the potential to grow into a tumour
- the body's solution to dealing with this is cell self-destruction
- with mounting evidence that cancer is the result of cells not dying fast enough, rather than multiplying out of control
- It is believed that a cell knows when to commit suicide because cells are programmed to do so
- i.e. self-destruct (sD) is an intrinsic property.



Apoptosis – *returning to Biological metaphors (cont.)*

- this sD is delayed due to the continuous receipt of biochemical reprieves.
- this process is referred to as *apoptosis*,
- meaning 'drop out',
  (used by the Greeks to refer to the Autumn dropping of leaves from trees);
- i.e., loss of cells that ought to die in the midst of the living structure.
- the process has also been nicknamed 'death by default',
- where cells are prevented from putting an end to themselves due to constant receipt of biochemical 'stay alive' signals



Self-protection & Trust
Security issues with Agents

Greenburg (1998) highlighted the situation simply by recalling the situation where the server

omega.univ.edu was decommissioned

- its work moving to other machines.
- When a few years later a new computer was assigned the old name,
- to the surprise of everyone email arrived, much of it 3 years old.
- the mail had survived 'pending' on Internet relays waiting for omega.univ.edu to come back up.

The same situation could arise for mobile agents;
these would not be rogue mobile agents –
they would be carrying proper authenticated credentials.

The mobile autonomic agent could cause substantial damage,
e.g., deliver an archaic upgrade (self-configuration)
to part of the network operating system resulting in bringing down the entire network

Self-protection & Trust
self-destruct property a solution?

Generally the security concerns with agents

- misuse of hosts by agents (accidental)
  - accidental or unintentional situations caused by that agent (race conditions and unexpected emergent behavior)

- misuse of agents by hosts, (accidental or deliberate)

- misuse of agents by other agents. (accidental or deliberate)

  - the latter two through deliberate or accidental situations caused by external bodies acting upon the agent.
  - the range of these situations and attacks have been categorized as: damage, denial-of-service, breach-of-privacy, harassment, social engineering, event-triggered attacks, and compound attacks.

Self-protection & Trust
self-destruct property a solution? (cont.)

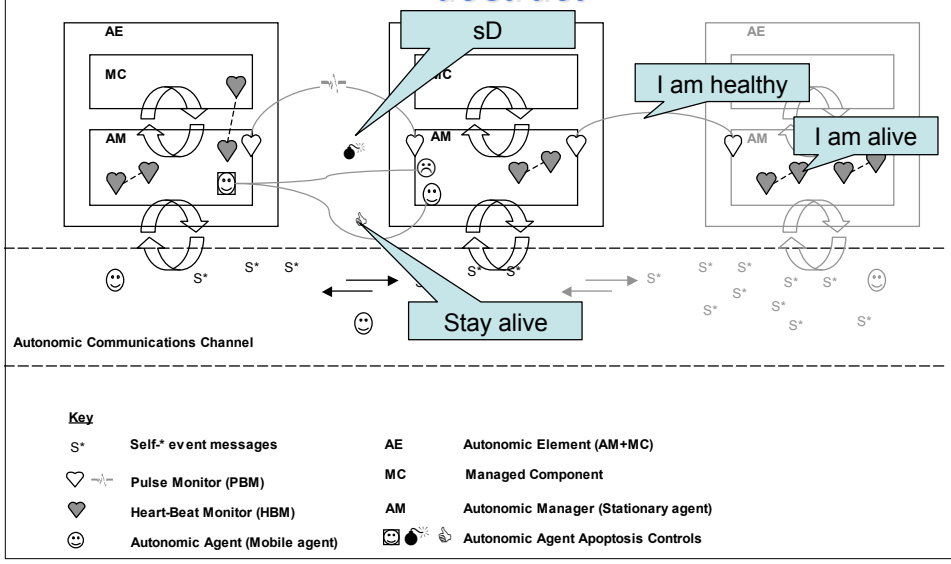
In the situation where portions of an agent's binary image e.g.,
- monetary certificates,
- keys,
- information, etc.

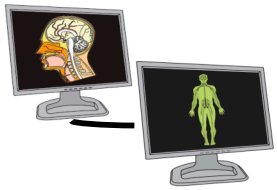
are vulnerable to being copied when visiting a host,
this can be prevented by encryption.

Yet there has to be decryption in order to execute,
which provides a window of vulnerability.

This situation has similar overtones to biological apoptosis,
where the body is at its most vulnerable during cell division.

Autonomic Computing –
The Autonomic Environment inc self-destruct





(6) Examples from NASA Sensor Network applications

Tutorial
Autonomic Computing in Real-Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

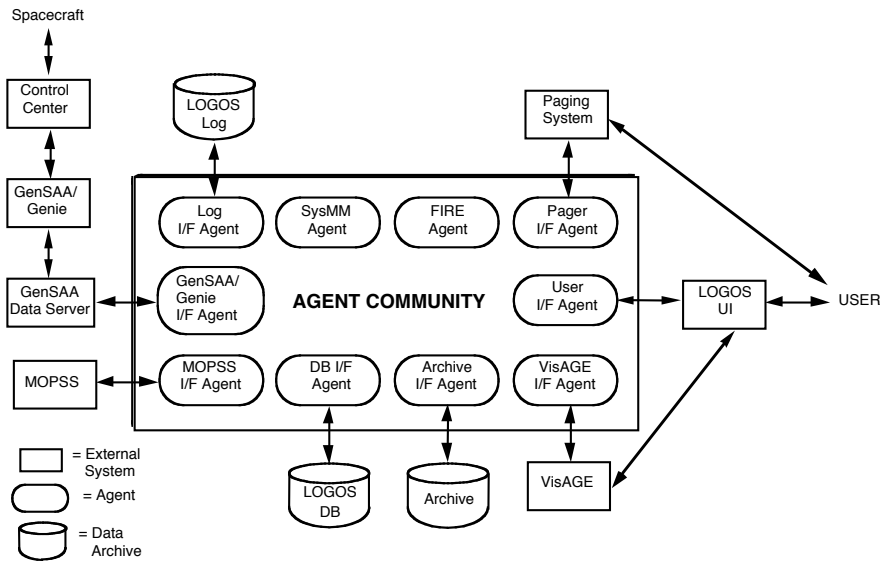
Challenges

- **NASA has set far-reaching autonomy goals for ground-based and space-based systems.**
  - **More reliance on "intelligent" systems and less on human interventions characterize its autonomy goals.**
- **Goddard has a leading role in the development of agent-based systems to realize NASA's autonomy goals.**

LOGOS

- **Lights-Out Ground Operations System**
  - a proof-of-concept system that used a community of autonomous software agents
  - Agents cooperated to perform functions previously performed by human control center operators.

An Overview of LOGOS



LOGOS

Self-configuring	Self-healing	Self-optimizing	Self-protecting
LOGOS self configures when it gets informed that a spacecraft pass is about to occur.	LOGOS self-heals by providing solutions to anomalous situations in the s/c component of itself and through human intervention.	LOGOS self-optimizes itself through the process of interacting with a human operator in dealing with a problem that it cannot initially handle.	LOGOS self-protects in a very limited fashion. An element of self-protection is the authentication of its users.

NASA Space Exploration Missions

- Future space missions will require cooperation between multiple satellites/rovers/craft for e.g.

  ANTS (Autonomous Nano-Technology Swarm)

- Developers are proposing intelligent, autonomous swarms to do new science
- Swarm-based systems are highly parallel and nondeterministic
- Testing these systems using current techniques will be difficult to impossible
- This raises issues for self-protection of system (mission) goals,

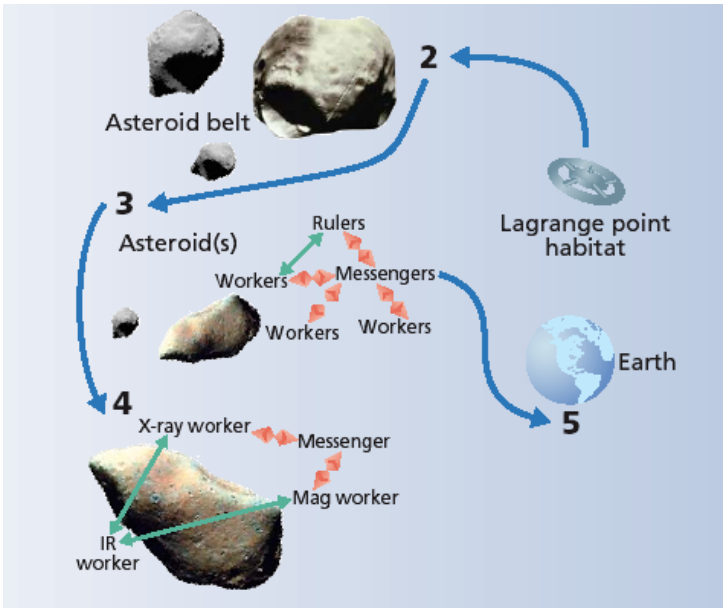
ANTS Space Exploration Missions

- The ANTS architecture is itself inspired by biological low level social insect colonies with their success in the division of labor.
- Within their specialties, individual specialists generally outperform generalists, and with sufficiently efficient social interaction and coordination, the group of specialists generally outperforms the group of generalists.
- Thus systems designed as ANTS are built from potentially very large numbers of highly autonomous, yet socially interactive, elements.
- The architecture is self-similar in that elements and sub-elements of the system may also be recursively structured as ANTS

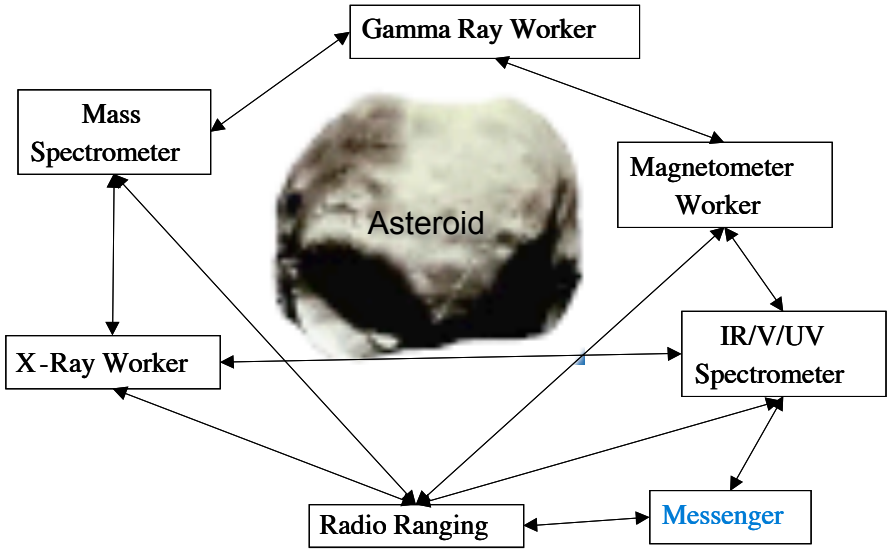
ANTS Space Exploration Missions *(cont.)*

- Targets for ANTS-like missions include surveys of extreme environments on the
  - Earth,
  - Moon,
  - Mars, as well as
  - asteroid,
  - comet,
  - or dust populations.
- The revolutionary ANTS paradigm makes the achievement of such goals possible through the use of many small, autonomous, reconfigurable, redundant element craft acting as independent or collective agents

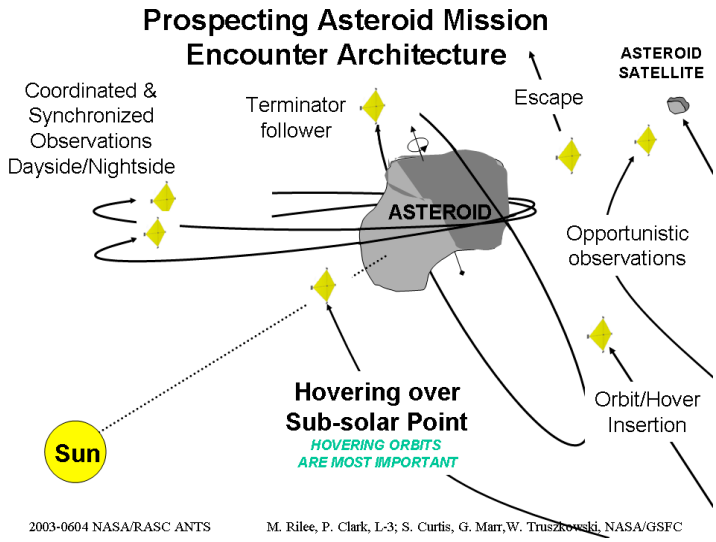
ANTS Mission



Self-Directed Exploration



ANTS MISSION



Difficulty of Testing Swarms

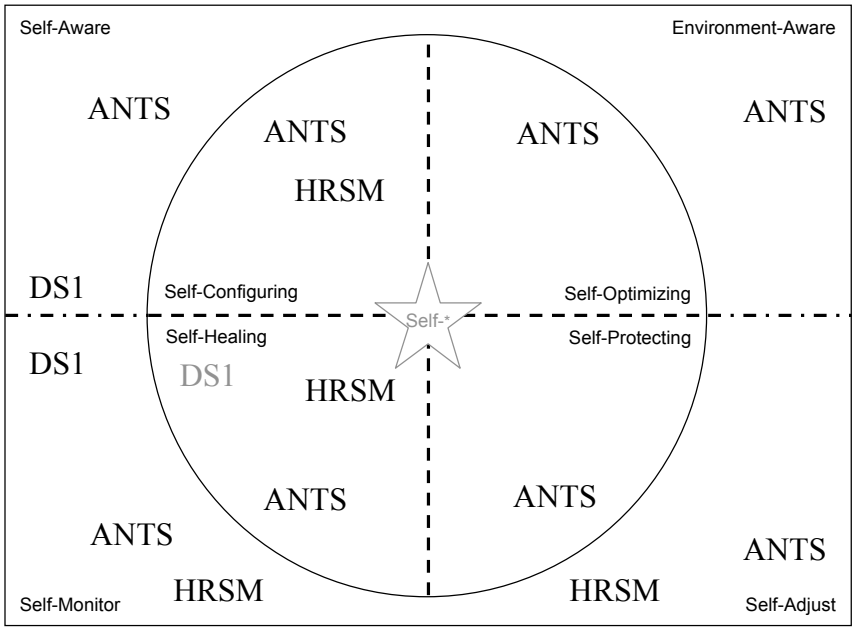
- Emergent properties that may not be known
- Highly distributed and parallel
- Large number of interacting entities
- Worse than exponential growth in interactions
- Intelligent entities (capabilities increase over time)
- Total or near total autonomy
- What if things do go wrong?

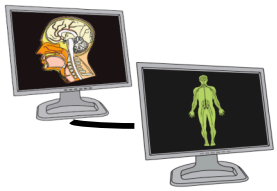
Autonomic Apoptosis in NASA Missions?

- With NASA missions we are not considering a generic situation.
- Mission control and operations is a trusted private environment.
- This eliminates many of the wide range of agent security issues
- Leaving the particular concerns;
  - is the agent operating in the correct context
  - and showing emergent behavior within acceptable parameters?

Autonomic Apoptosis in NASA Missions?

- Suppose one of the worker agents was indicating incorrect operation, or when co-existing with other workers was the cause of undesirable emergent behavior, and was failing to self-heal correctly.
- That emergent behavior (depending on what it was) may put the scientific mission in danger.
- Ultimately the stay-alive signal from the ruler agent would be withdrawn.
- If a worker, or its instrument, were damaged, either by collision with another worker, or (more likely) with an asteroid, or during a solar storm, a ruler could withdraw the stay-alive signal and request a replacement worker.
- Another worker could self-configure to take on the role of the lost worker; i.e., the ANTS adapt to ensure an optimal and balanced coverage of tasks to meet the scientific goals.
- If a ruler or messenger were similarly damaged, its stay-alive signal would also be withdrawn, and a worker would be promoted to play its role.





(6) Another Example
Hubble Robotic Servicing Mission

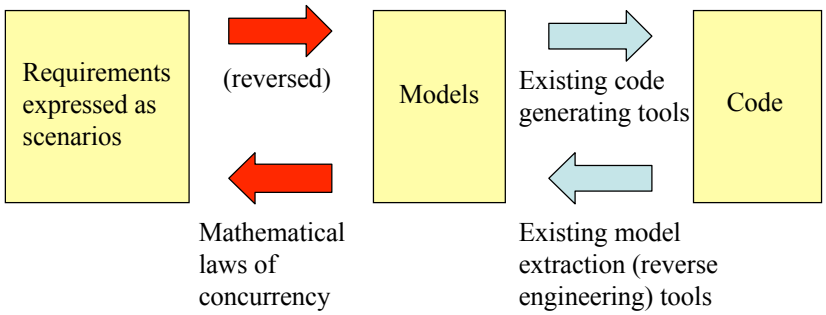
Tutorial
Autonomic Computing in Real-Time Systems

Roy Sterritt & Mike Hinchey
r.sterritt@ulster.ac.uk michael.g.hinchey@nasa.gov

Solution

- Requirements Based Programming
  - Tractability from Requirements all the way through to code
  - Builds on an MBD approach by providing a "front end"
  - Our contention is that RBP should be *formal*

Requirements to Design to Code



Requirements To Design To Code

- This new method (R2D2C) provides mathematically tractable development of systems with provable correctness
- Automated translation of requirements into a provably correct system
- The practicality of the approach is assured because it is transparent and based on requirements expressed as scenarios – an approach familiar to engineers

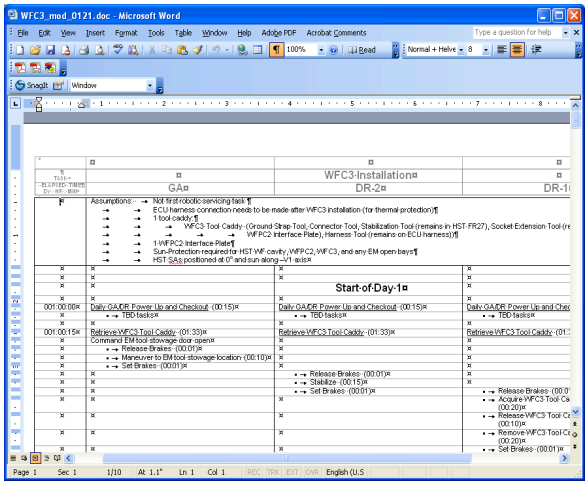
Benefits of the Method

- Automation of entire development process
- Significant increase in quality
- Ability to do formal proof on properties of implementations
- Ability to do formal proof of correctness
- Automated means for requirements analysis
- Guaranteed correspondence between requirements and their implementation as code

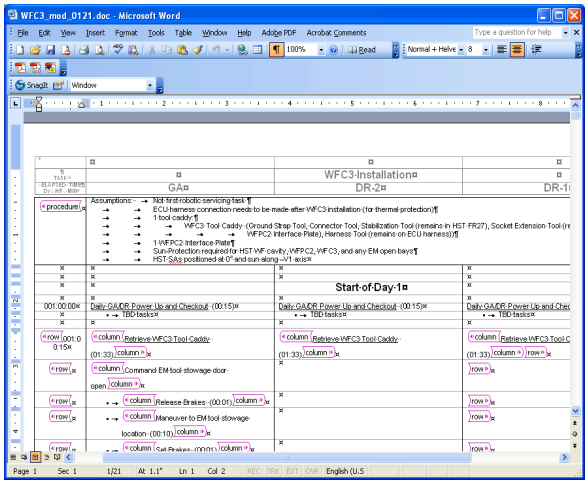
Applications

- End-to-end automatic code generation of provably correct systems
- Automatic reimplementation after any requirements change
- Exploiting re-use across platforms
- Reverse engineering legacy systems to a mathematically sound model
- Analysis and documentation of existing systems (e.g., expert systems)
- Re-engineering of legacy systems to a provably correct new implementation

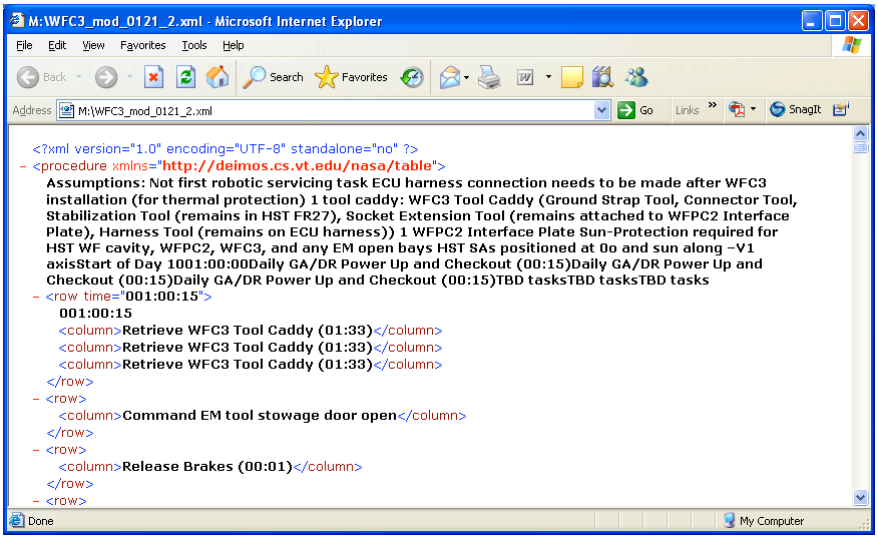
HRSM Procedures



HRSM Procedures

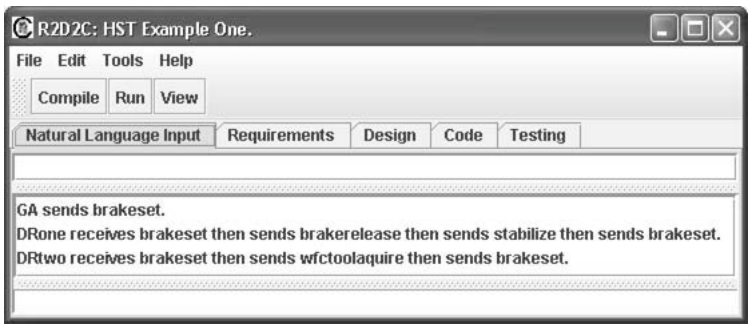


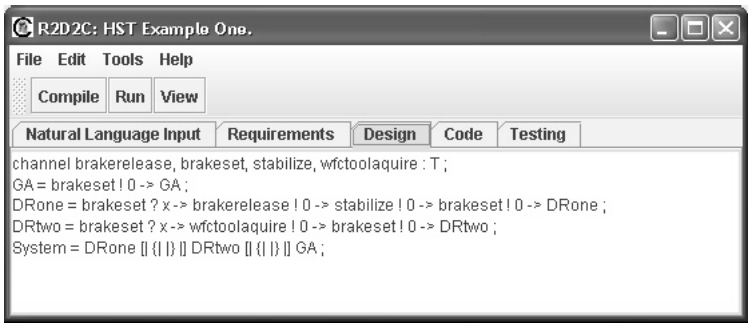
XML Document

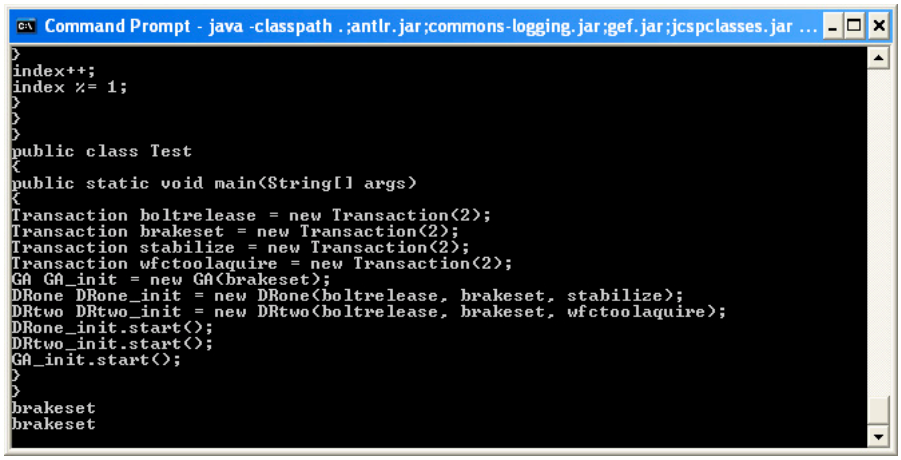


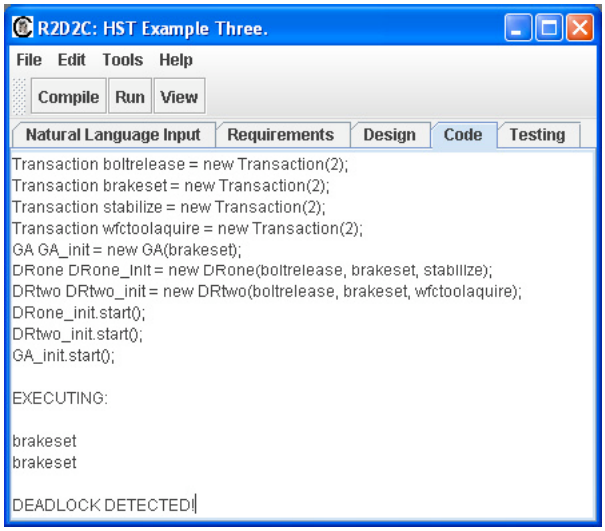
Natural Language Input for R2D2C

```
-- Actor 1
0: <retrieve>(wfc3_tool_caddy)
3: <command>(em_tool_stowage_door)
4: <release>(brakes)
5: <maneuver>(, <to> em_tool_stowage_location)
6: <set>(brakes)
18: <install>(wfc3_tool_caddy, <at> hst_worksite)
21: <release>(brakes)
22: <maneuver>(, <to> hst_wfc3_tool_caddy_stowage_location)
23: <command>(em_tool_stowage_door)
24: <set>(brakes)
31: <install>(stabilization_tool, <in> hst_fr27)
43: <retrieve>(wfpc2_interface_plate)
46: <release>(brakes)
…
```

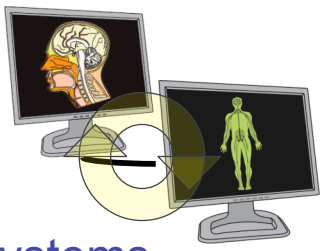



R2D2C: HST Example One.
File Edit Tools Help
Compile Run View
Natural Language Input Requirements Design Code Testing
channel brakerelease, brakeset, stabilize, wfctoolaquire : T ;
GA = brakeset ! 0 -> GA ;
DRone = brakeset ? x -> brakerelease ! 0 -> stabilize ! 0 -> brakeset ! 0 -> DRone ;
DRtwo = brakeset ? x -> wfctoolaquire ! 0 -> brakeset ! 0 -> DRtwo ;
System = DRone [| {| |} |] DRtwo [| {| |} |] GA ;


Command Prompt - java -classpath .;antlr.jar;commons-logging.jar;gef.jar;jcspclasses.jar ...
}
index++;
index %= 1;
}
}
}
public class Test
{
public static void main(String[] args)
{
Transaction boltrelease = new Transaction(2);
Transaction brakeset = new Transaction(2);
Transaction stabilize = new Transaction(2);
Transaction wfctoolaquire = new Transaction(2);
GA GA_init = new GA(brakeset);
DRone DRone_init = new DRone(boltrelease, brakeset, stabilize);
DRtwo DRtwo_init = new DRtwo(boltrelease, brakeset, wfctoolaquire);
DRone_init.start();
DRtwo_init.start();
GA_init.start();
}
}
brakeset
brakeset



Invitation – to joint the new IEEE Task Force on Autonomous & Autonomic Systems



- http://www.computer.org/tab/
- http://www.computer.org/TCsignup/index.htm
- http://tab.computer.org/aas/


ARTES
SUMMER SCHOOL 2005

Real-Time Issues in Wireless Sensor Networks

Chenyang Lu
Department of Computer Science and Engineering
Washington University in St. Louis
http://www.cse.wustl.edu/~lu

Many mission-critical applications require wireless sensor networks to interact with physical environments under stringent timing constraints and severe resource constraints. Examples include intruder tracking, medical care, fire monitoring, and structural health monitoring. This tutorial presents our research on a range of real-time issues in wireless sensor networks including (1) packet scheduling algorithms for end-to-end real-time communication; (2) power management protocols for energy-efficient real-time data collection; and (3) spatiotemporal query services for mobile users. This tutorial also discusses research challenges in the area of real-time wireless sensor networks.

Biography

Dr. Chenyang Lu is an Assistant Professor in the Department of Computer Science and Engineering at Washington University in St. Louis. He received the Ph.D. degree from University of Virginia in 2001, the M.S. degree from Chinese Academy of Sciences in 1997, and the B.S. degree from University of Science and Technology of China in 1995, all in computer science. He is author and co-author of more than 40 refereed technical papers and a recipient of the NSF CAREER Award. His current research interests include wireless sensor networks, adaptive QoS control, and real-time embedded systems and middleware.

References

C. Lu, B.M. Blum, T.F. Abdelzaher, J.A. Stankovic, and T. He, RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), September 2002.

O. Chipara, C. Lu, and G.-C. Roman, Efficient Power Management based on Application Timing Semantics for Wireless Sensor Networks, International Conference on Distributed Computing Systems (ICDCS'05), June 2005.

C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, A Spatiotemporal Query Service for Mobile Users in Sensor Networks, International Conference on Distributed Computing Systems (ICDCS'05), June 2005.

S. Bhattacharya, G. Xing, C. Lu, G.-C. Roman, B. Harris, and O. Chipara, Dynamic Wake-up and Topology Maintenance Protocols with Spatiotemporal Guarantees, International Conference on Information Processing in Sensor Networks (IPSN'05), April 2005.


ARTES
SUMMER SCHOOL 2005

Analysable Components

John Håkansson, Department of Information Technology, Uppsala University

SaveCCM: An Analysable Component Model for Real-Time Systems

SaveCCM is a component model defining the graphical language and run-time framework of the SAVE component technology, SAVEComp

The SAVE component technology is developed with safety critical vehicular systems in mind. It is similar to the Rubus component model, with additional focus on quality attributes and independence of underlying operating system.

Component Based Development

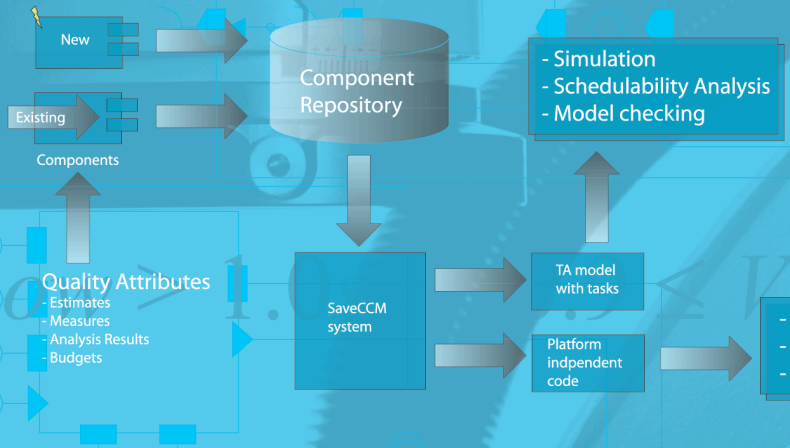
CBD is a promising approach for embedded systems. Typical for embedded software is the presence of resource constraints in multiple dimensions. An essential dimension is time, since many embedded systems have real-time requirements.

The SAVE Project

SAVE is a national programme supported by SSF. The goal is to establish an engineering discipline for systematic development of component-based software for safety critical embedded systems.

Design and Analysis

An overview of the analysis enabled development process of SaveCOMP



The UPPAAL Team

- Wang Yi
- Paul Pettersson
- Leonid Mokrushin
- John Håkansson
- Pavel Krcál
- Shi Xiaochun

TIMES and UPPAAL

Tools for timed automata:
- Modelling and simulation
- Model checking
- Schedulability analysis (TIMES)
- Code generation (TIMES)

www.uppaal.com, www.timestool.com

Case Study

A simple SaveCCM system with a PI controller has been used as a case study. Temporal properties of the controller have been successfully verified using the timed automata model checker UPPAAL.

John Håkansson
Ph.D. student
Uppsala University
johnh@it.uu.se

Architectures for Logistics Telemetry Applications

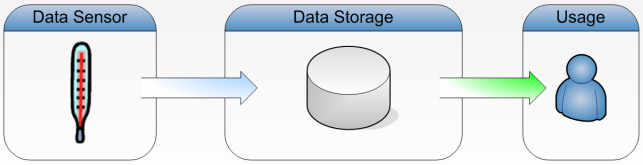
Markus Adolfsson
markus@x3-c.com

Halmstad University & XCube Communication AB
Industrial PhD student

Me surrendering to my son Sebastian's wish to hug / wrestle his dad

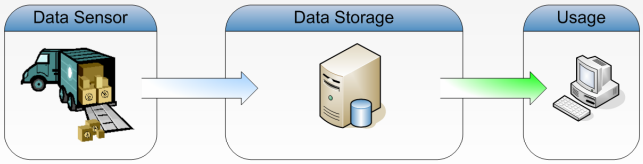
1 What is telemetry?

Tech Encyclopedia defines the concept of Telemetry as *"Transmitting data captured by instrumentation and measuring devices to a remote station where it is recorded and analyzed. For example, data from a weather satellite is telemetered to earth"*



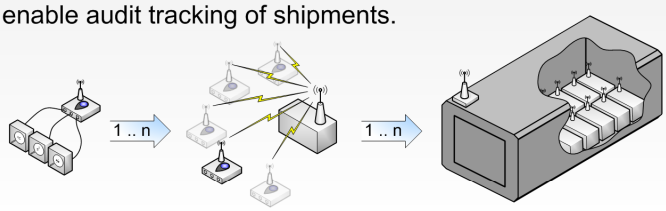
2 Suitable model for Logistics?

Logistic applications striving for "live-tracking" fit well into the telemetry model. E.g., cold-chain surveillance for dairy products and meat transports where there is a great need to monitor such applications by logging entire shipments, but also to alert when cold-aggregates break down, pallets are left too long on docking bays, etc.



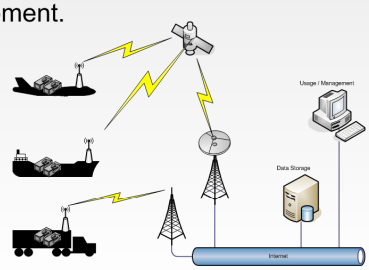
3 State-of-the-art Technology

Embedded Active RFID-tags placed on the packets measure one or more attributes (e.g., temperature, humidity, acceleration). The ARFID tags form a wireless sensor network that periodically sends their measurements to a master node. The master node forwards its received measurements to the central data-storage along with both time- and position-stamps to enable audit tracking of shipments.



4 World-wide, multi-modal tracking

The goal of these kinds of applications is to be able to track and monitor goods worldwide, regardless of the good's current carrier is a truck (*land*), a ship (*marine*), or an airplane (*air*). From a system-perspective, it is still the same shipment.



5 A multi-variable optimization problem

How can an architecture for logistics telemetry applications be modeled? What requirements are there and how do they relate to each other? How can maximum lifetimes be achieved in the deployed, heavily constrained monitoring devices, still conforming to the requirements defined for the application? Real-time requirements enabling efficient detection of new devices in the wireless sensor network, but also to guarantee that measurements reach the application user in time (e.g., ordinary measurements vs. alerts).

Design of Electrical Architectures for Safety Cases

ARTES++ Summerschool 2005

Participants

PhD Student	Fredrik Törner
Academic supervisor	Peter Öhman, Chalmers
Industrial supervisor	Per Johannessen
Research Program	Programrådet för fordonsforskning

Electrical Architecture

Architecture is defined according to IEEE-1471 as:

> "The fundamental *organization* of a system embodied in its components, their *relationships* to each other, and to the environment, and the *principles* guiding its design and evolution"

An architecture is the basis for vehicle functionality and aims to fulfill non-functional requirements such as safety and cost.

Figure 1 shows 26 ECUs distributed in a car connected by CAN and MOST. Together they realizes the electrical system in an XC90, supporting several hundred customer functions.

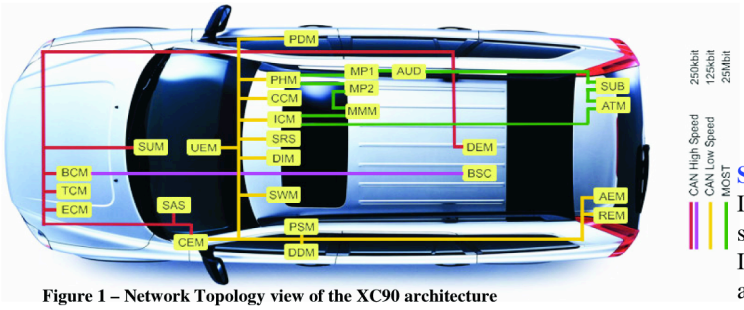


Figure 1 – Network Topology view of the XC90 architecture

Problems

Functionality in today's cars are evolving towards solutions more dependent on electronics. New advanced safety and convenience functionality demands implementation in embedded, real-time electronics.

Integration between systems are necessary to maximize functionality and decrease cost by utilize inherent redundancy between sensors and processors. This brings a high complexity to the systems and raises high demands on the architecture in aspect of reliability and available bandwidth.

Possible solutions to this are real-time communication protocols such as FlexRay. These technologies will be necessary but they are not the complete solution. The integration itself needs to be implemented in a way that does not decrease the safety of the vehicles occupants.

The OEM, who will have the integrator role, will be responsible for the complete vehicles safety, and will have to show this property to a number of stakeholders: e.g. government, customers and liability courts.

Safety Case

In order to show how and why a system is safe an argumentation is needed. A safety case is the collection of justifications why the system is safe and a safety argumentation combining them. The justifications can be of several types e.g. Hazard Analysis results, FMEA results, verification protocols. The exact contents are dependant on the product and its context. **Figure 2** visualize a possible structure of a safety case.

New legislation are demanding safety cases for some systems in the automotive industry, currently there are requirements for brake and steering systems. Safety cases can also be used in product liability cases, providing a solid argumentation of the safety for a system.

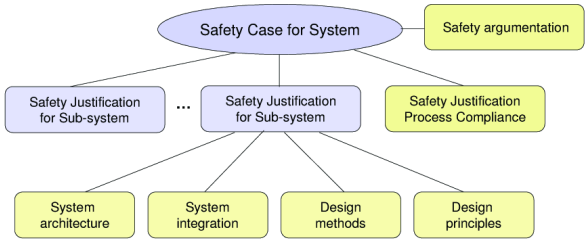


Figure 2 – Safety Case structure

Safety Process

In order to handle development of safety critical systems a few standards exists that may be applied to the automotive industry. IEC61508 is the defacto safety standard today but is found difficult and inappropriate to use in the automotive industry. MISRA is a standard developed for the automotive industry and FAKRA is a possible upcoming ISO standard currently under development.

Figure 3 shows an overview of what could be included in a safety case according to MISRA.

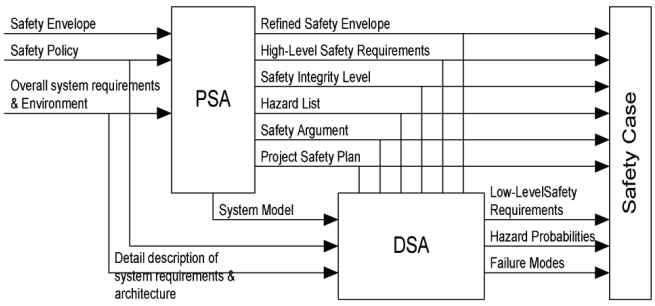


Figure 3 – Overview of safety process

Research Questions

How is a safety case best developed for distributed control systems in the automotive industry?

How can a safety case be developed to support the design of electrical architectures?

Project start | Lic | Project finish PhD

VOLVO
Volvo Car Corporation

Project ModComp

Model-Based Development and Competence Integration

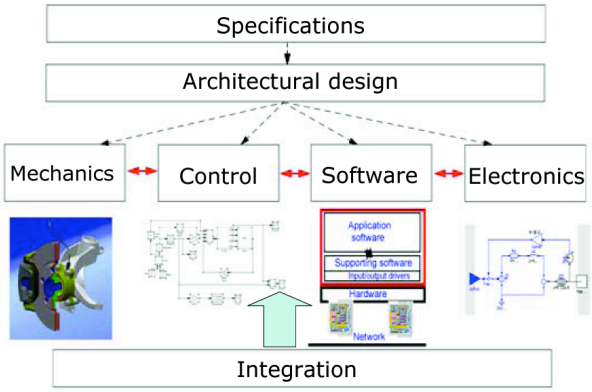
Introduction:

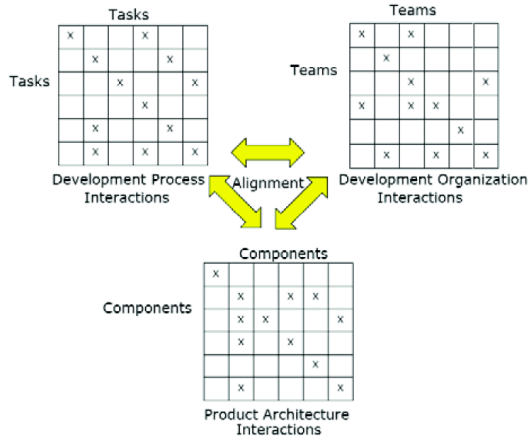
This project is a collaboration between KTH and Volvo Car Corporation (VCC). The project will develop knowledge, supporting methods and prototype tools for efficient development of complex mechatronical products. The focus in the project is on automotive embedded control systems. Central problems in developing such products include how to handle the large teams with many competences involved, fragmented information, and difficulties in integration. The aim is to enhance the understanding about the work procedures, to improve management of the complexity within the development of mechatronics products, and to eventually provide practical means for efficient integration.

The project runs from 2004 through 2006.

Expected Contributions:

- Documentation and analysis of the current industrial situation.
- State of the art for multi-domain model based development and integration.
- Requirement specification for a data management and integration platform.
- Methodology and ontology with respect to integration in mechatronics system and organizational development.
- Contributions to a research platform for model and tool integration.
- Improved product development at VCC through new work methods implementation and use of new tools.





Technology and competence integration in the mechatronics. The overall question posed in the project: How to manage the model/tool integration and the competence integration in the mechatronics products development?

Some questions and challenges for multi-disciplinary development of automotive embedded control systems:

- What are the design trade-offs and communication required between different competences?
- What are appropriate work procedures?
- What services should CAE tools provide?
- How can data in different tools, representing different aspects or views, be managed and integrated?

Participants:

KTH, Department of Machine Design:

Mechatronics - Professor Jan Wikander (Project leader)

Embedded control system - Professor Martin Törngren, Junior Researcher Ola Redell, PhD student Jianlin Shi

Integrated product development - Professor Margareta Norell, Assistant professor Sofia Ritzén, PhD student Diana Malvius

Volvo Car Corporation:

Niklas Vemdahl and Nils Edeus

Contact:
Jianlin Shi, Doctoral Student
jianlin@md.kth.se, +46-8-790 6776
Diana Malvius, Doctoral Student
malvius@md.kth.se, +46-8-790 7806
Ola Redell, Doctor
Ola@md.kth.se, +46-8-790 8343

KTH Machine Design

www.md.kth.se

Project home page:
http://www.md.kth.se/RTC/modcomp

This project is funded by the Swedish Agency for Innovation System (VINNOVA)

Monday dinner

Scandic Billingen Trädgårdsgatan 10 at 19.00

Scandic Billingen byggdes år 1888 och har sedan dess varit känt för sin vackra arkitektur och trivsamma atmosfär. Hotellet är centralt beläget vid järnvägsstationen och med centrum runt hörnet.

Meny

Päron- och chevrétoast serveras med en knaprig serranoskinka

Chateau Briand serveras med kraftig rödvinsås samt en ost- och potatismuffin och rostade grönsaker

Husets chokladkaka serveras med grädde smaksatt med likör.


ARTES
SUMMER SCHOOL 2005

Component-Based Software Engineering

ARTES Summerschool
Skovde, August 2005

M.R.V. Chaudron
M.R.V.Chaudron@tue.nl
www.win.tue.nl/~mchaudro/
Technische Universiteit Eindhoven

- Introduction CBSE & Reuse
  - Motivation
  - Concepts, Definitions, Terminology

Observations on the practice of SE

About 80% of software engineering deals with changing existing software

It is not the strongest of the species that survive, nor the most intelligent, but the ones most responsive to change.
-- Charles Darwin

Time to market is an important competitive advantage: incorporate successful innovations quickly

→ Systems should be built to facilitate change
→ easy removal and addition of functionality

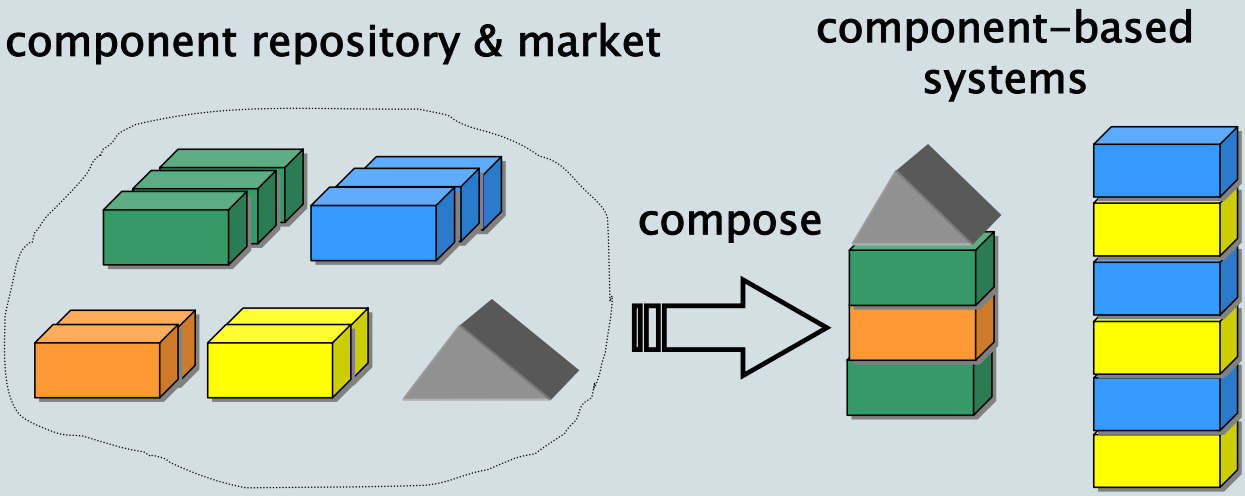
Problems of Software Engineering

- The size & complexity of software increases rapidly
- Single products become part of product families
- Software is upgraded after deployment
- The time-to-market must decrease significantly
- The cost of products must be reduced

The CBD-'Solution'

Systems should be assembled from existing components

Idea dates (at least) to the1968 NATO Conference
Douglas McIlroy: Mass Produced Software Components

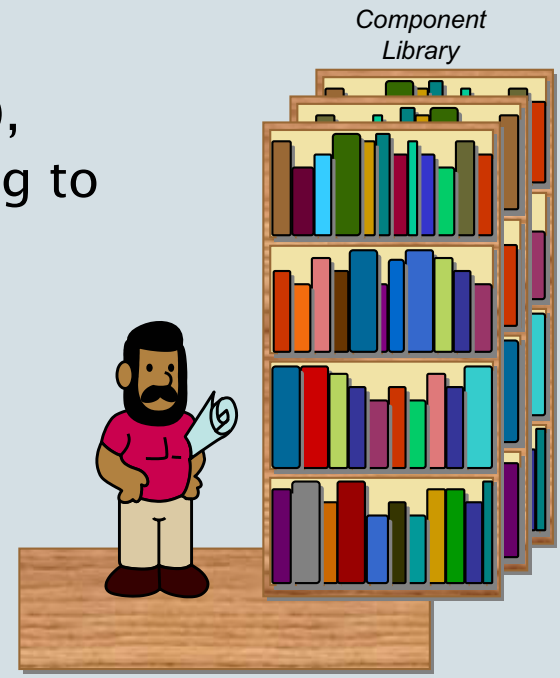


Why Components?

Following other engineering disciplines (civil and electrical), software engineering is looking to develop

- a catalogue of software building blocks
- connection standards

Confusing or helpful?



What is CBSE?

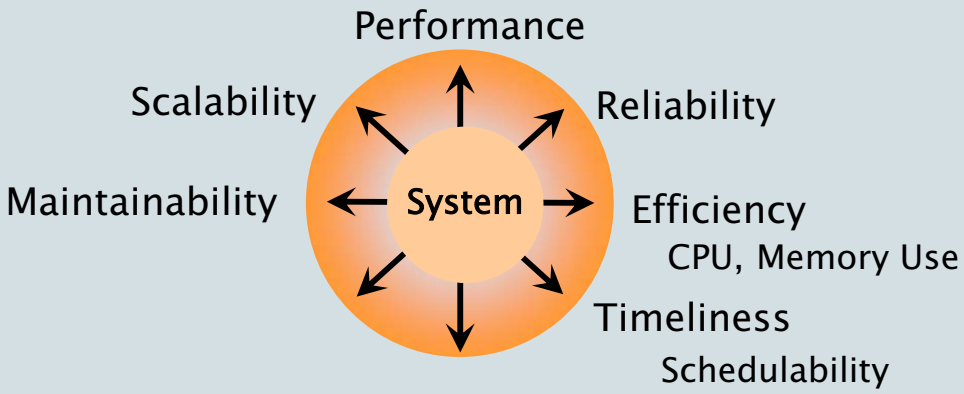
based on definition of SEI in CMU/SEI-2000-TR-008

Component-based Software Engineering is concerned with the *rapid assembly* and *maintenance* of component-based systems, where

- components and platforms have *certified properties*
- these certified properties provide the basis for *predicting properties of systems* built from components.

Predictability is a *key* property of mature engineering disciplines.
It enables feedback on design and adaptation;
i.e. development time is reduced because we can analyze prior to building

Extra Functional Properties



Essential system engineering problem:

- **a plurality of contradictory goals**
- **a plurality of means (technology, process)**
  each of which provides a varying degree of help or hindrance in achieving a given goal

Business Drivers for CBD

Improve Productivity:

Build more software using fewer resources through enabling the assembly of systems from components that may be independently developed by different parties.

Independent in time and space
→ independent from ultimate application
→ independent from (future) peer-components

Motivations for CBSE

- Productivity
- Quality
- Time-to-market
- Maintenance

Strategic business goals that increase

- Turnover
- Market share

reduce

- Cost of development
- Cost of ownership

€ € € profit € € €

CBSE & Software Productivity

Increase competitiveness (sw/€):

- Reduce cost of development
- Increase software/€

Limited human talent (sw/people):

- Increase software/person
  ⇒ reuse existing solutions, rather than invent them

CBSE & System Quality

Improve Quality:

Idea: Assuming that a collection of high-quality components is available, assembling these should yield systems of high-quality.

1. The cost of establishing the high quality of components is amortized over multiple use.

2. Multiple use of a component increases the likelihood of finding and removing errors.

CBSE & Maintenance

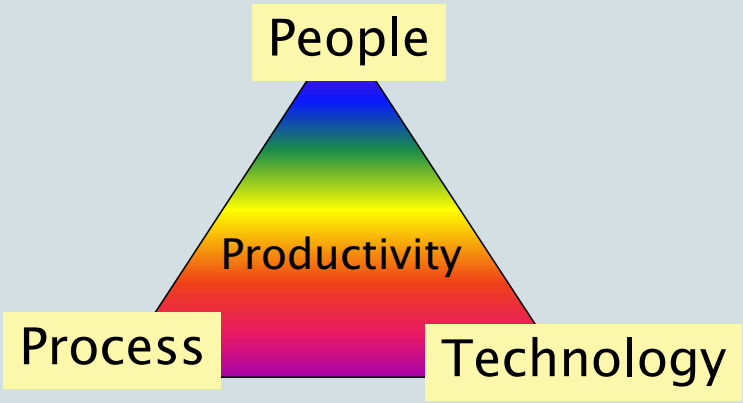
The use of CBD requires good modular design.

This modularity provide quality properties like
- comprehensibility/understandability
- maintainability
- flexibility
- …

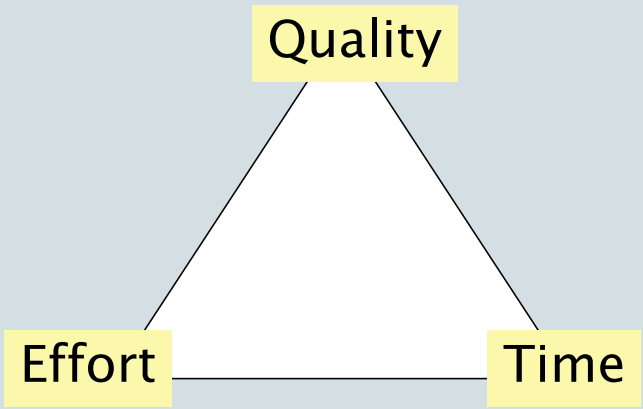
CBSE & Time-to-market

If the reuse of a component requires less time than the development of a component, systems can be built faster.

Productivity & Quality



Dependent dimensions



Technical Drivers for CBSE

CBSE may help improve system qualities

Reuse-based Software Engineering

- Reuse-based SE has many **business drivers** in common with CBSE:
  - increase productivity & quality
  - reduce time-to-market,
  - reduce development cost

However, reuse imposes less technical- and design-constraints on the unit of reuse (*asset*).

CBSE enables Reuse, Reuse is not sufficient for CBSE.

Reusable Assets

Virtually any product of the SE process can be reused:

- Requirements
- Architectures
- Designs
  - design patterns, interfaces
- Source Code
  - ranging from to libraries, patterns, to modules, to macros, coding conventions, ...
- Test Scripts

Questions?

What is a software component?

How can you recognize a software component?

Suggestions from the audience?

Reflect on differences between civil and electrical engineering on the one hand and software engineering on the other hand

Cross-cutting concerns

What is a Component?

Probably more definitions than for ‘software architecture’

A software component is a unit of composition with contractually specified *interfaces* and *explicit context dependencies* only

A software component is *independently deployable* and subject to *composition* by third parties.

Clemens Szyperski, 1997

What is *independent deployment*?

A software component is a unit of *independent deployment*

→ no dependencies on peer-components
  → some 'meaningful' functionality by itself
    components tend to be 'large grained'

→ never partially deployed

What is a Component?

A reusable software component is a logically cohesive, loosely coupled module that denotes a single abstraction.

Grady Booch, *Software Components with Ada, 1987*

Tries to provides some design guidance.
What is cohesive? loosely coupled? single abstraction?

Object Technology and CBSE

"OT is Neither Necessary Nor Sufficient for CBSE"

- OT was a useful and convenient starting point for CBSE
- OT did not express full range of abstractions needed by CBSE (*insufficiency*)
- It is possible to realize CBSE without employing OT(*non-necessity*)

CBSE might induce substantial changes in approach to system design, project management, and organizational style

What is a Component?

"A binary unit of independent production, acquisition, and deployment that interacts to form a functioning system."

– C. Szyperski, *Component Software*

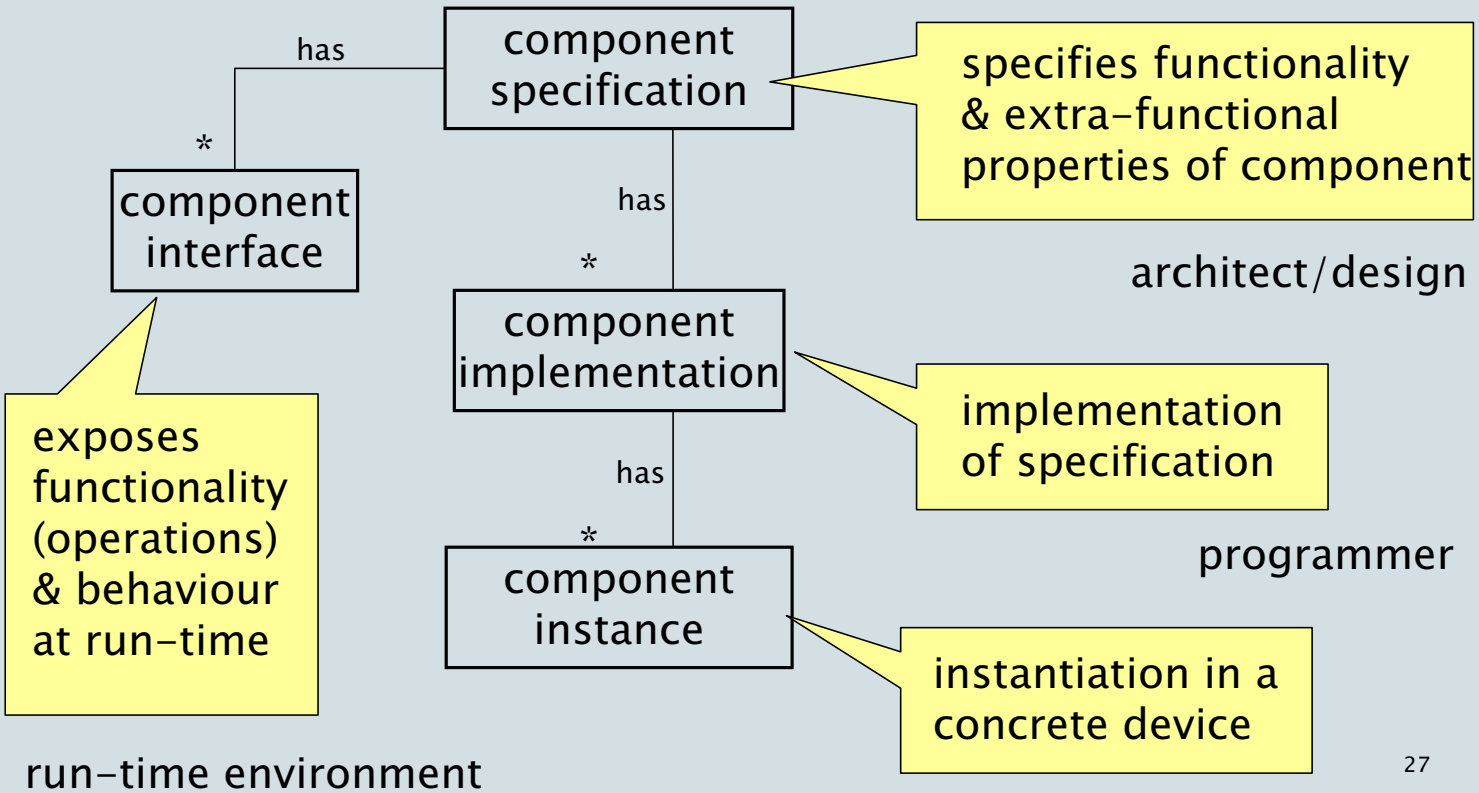
"A component is an independently deliverable package of operations."

– *Texas Instruments Literature*

"A replaceable unit of development work which encapsulates design decisions and which will be composed with other components as part of a larger unit."

– Desmond D' Souza, in *Catalysis*

Useful Distinction



It depends (on what?)

It depends to some degree on what you want to do with components:

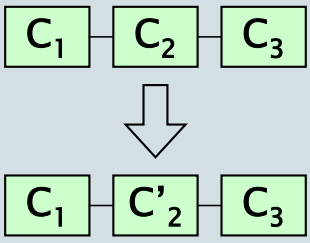
- REPLACING one part of an implementation with another:
  - then adhering to a well defined specification is sufficient
  - at run-time, then mechanisms are needed for
    - registering, locating / binding components
- EXTEND components
  - extend interfaces, merge implementations

It depends (on what?)

- TRADING (use across multiple organizations)
  - need mechanism for
    - hiding intellectual property
    - certification
- RE-USE
  - need repository for searching / matching, versioning
  - economics impact granularity
- Application domain BUSINESS/EMBEDDED
  - type and granularity of component

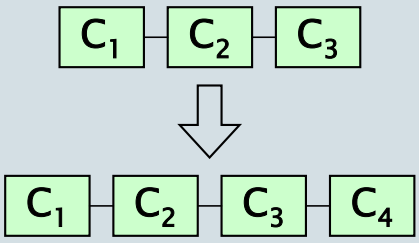
Different usage requirements

Substitutability



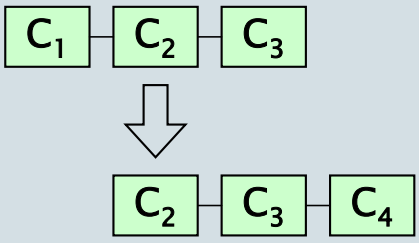
complete specification

Extensibility



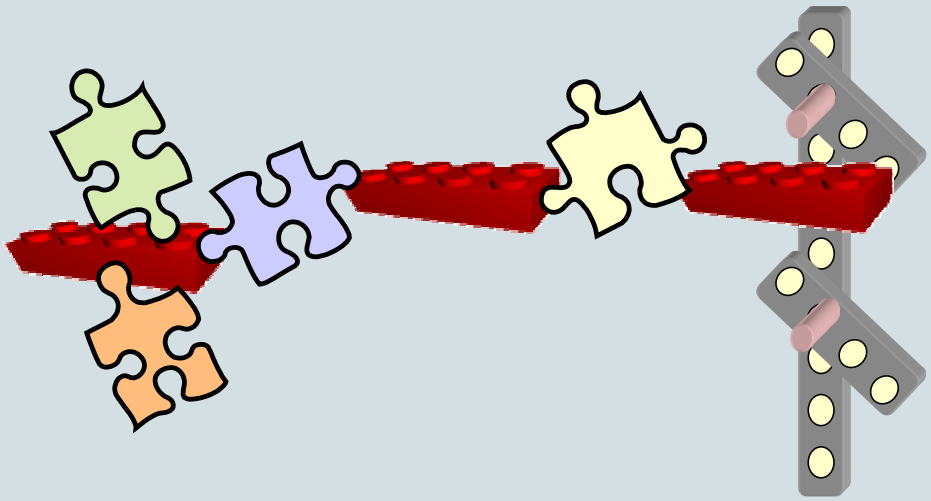
extensible architecture

Decomposability



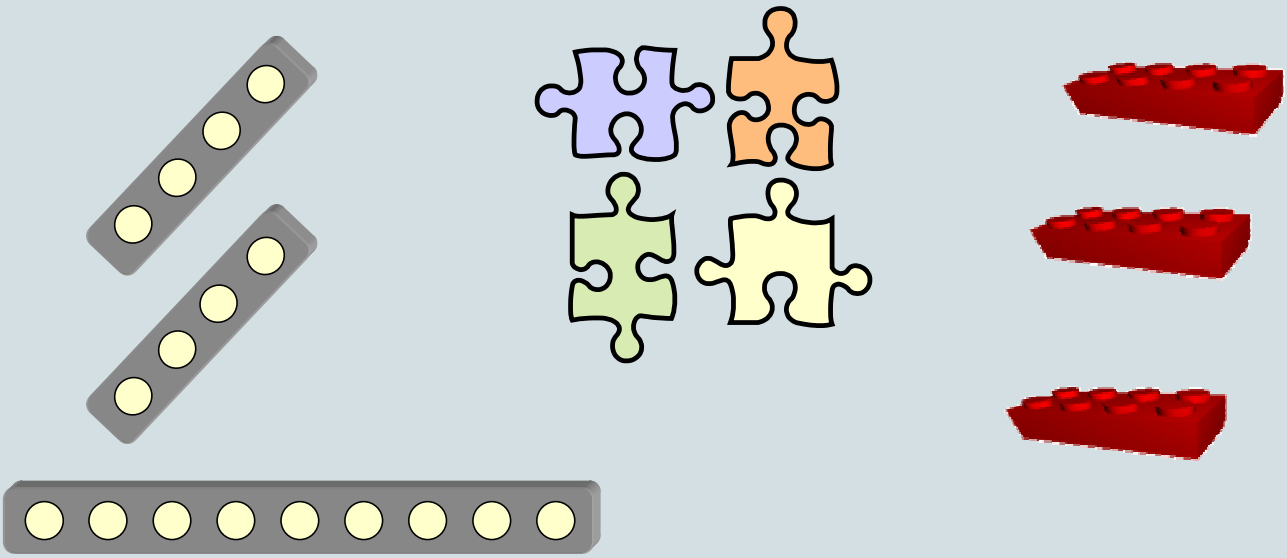
generic components, flexible architecture

CBD in Practice



Lego + Fisher Technik + Meccanno + Ministek + ...

A component can be used within the scope of a component-model



.Net, Enterprise Java Beans, Corba Components + ...

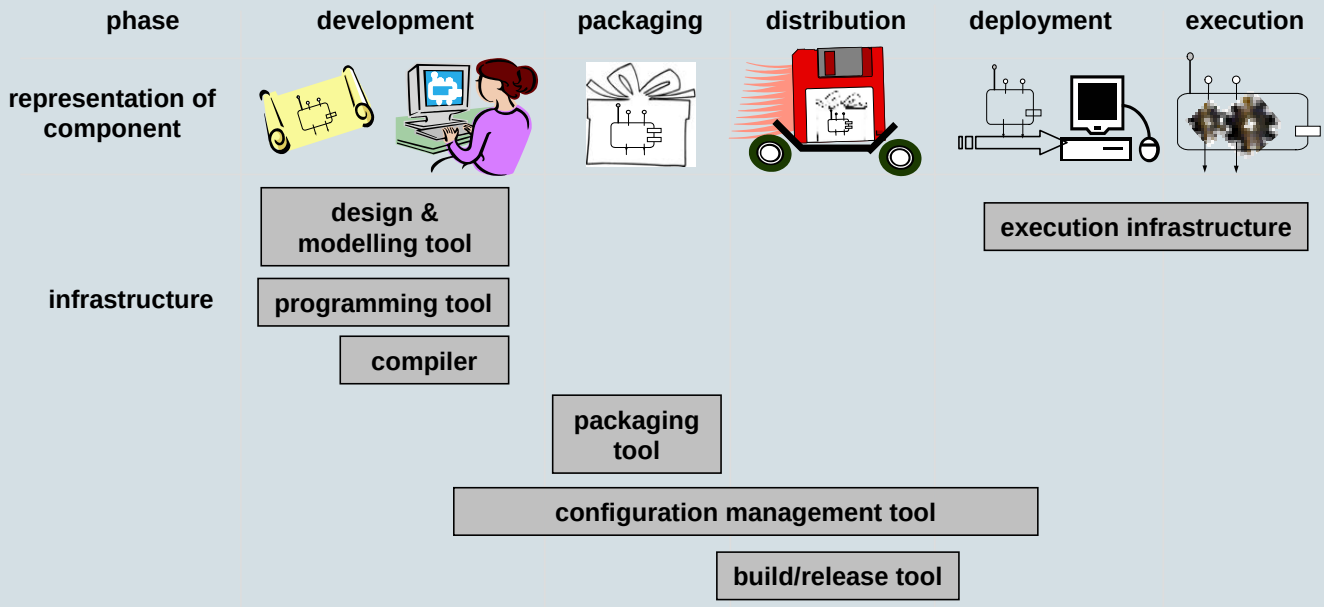
Component Model

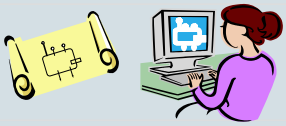
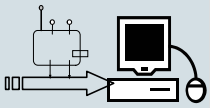
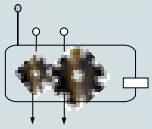
Definition A *component model* specifies the standards and conventions that are needed to enable the composition of independently developed components.

Multitude of Component Models

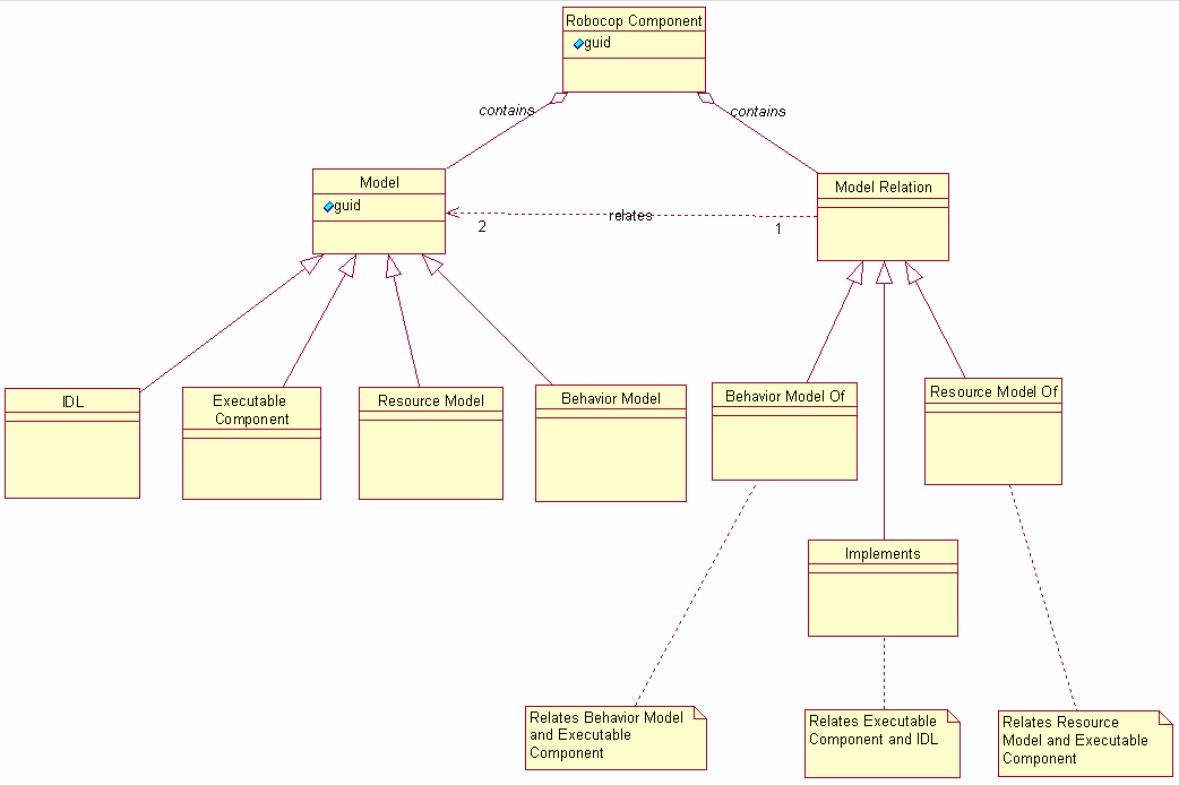
- Q: Why are there so many component models?

- A: because there are many different requirements
  - Technically: on component-systems
  - Organizationally: development process


phase
development
packaging
distribution
deployment
execution
representation of component
design & modelling tool
execution infrastructure
infrastructure
programming tool
compiler
packaging tool
configuration management tool
build/release tool

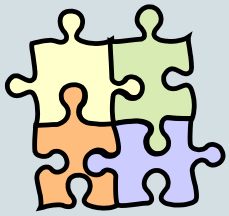
phase	development	packaging	distribution	deployment	execution
					
representation of component	model / specification e.g. UML, Petri-net				
	source code & data files e.g. C++	Structured archive e.g. zip-file			
			• Solid-medium that preserves file-structure • network: requires conventions on where to install		
				unpack & place comp in memory	
					executable code - low level machine code - high level code for interpretation

Component Packaging (UML)



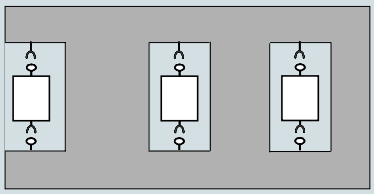
Architecture vs. Generic Components

Architectural component



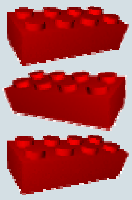
there is exactly one component that fits each place in the system

Framework component 'plug-in'



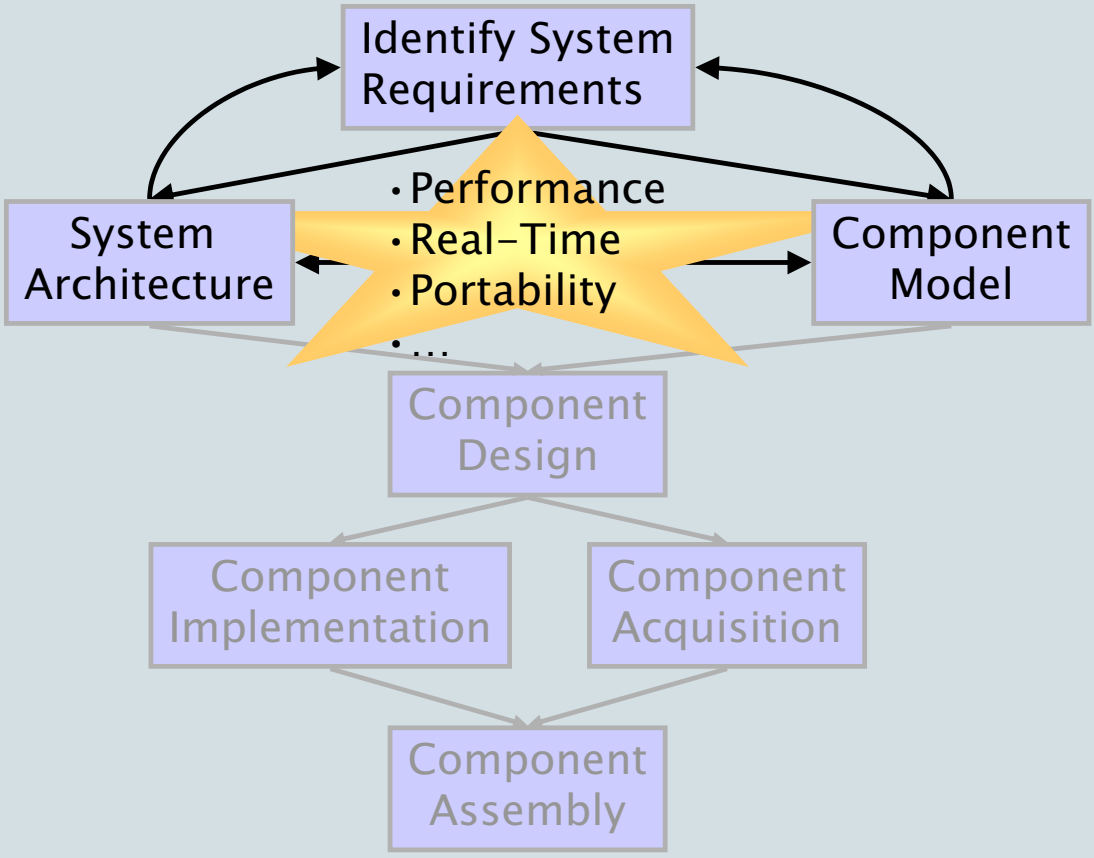
several components fit at a particular place in the system

Generic component



every component fits at every place

CBSE Development Activities



Different types of requirements

- When to compose?
  - At what stage is flexibility needed?
    - Design-, compilation-, run-time
- Different extra-functional requirements

Aspects of Component Models

A component model is a set of agreements that is needed to enable the *combination* of components.

A component model typically addresses

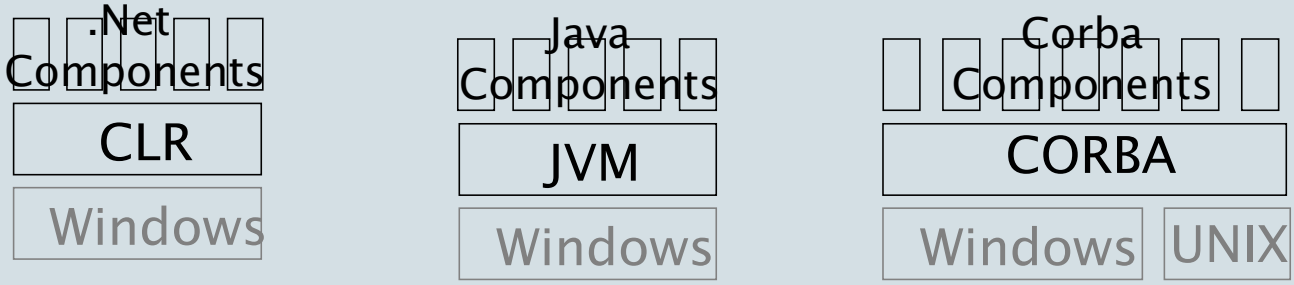
- Life-cycle management: instantiation, (de)activation, removal
- Binding mechanisms
- Interaction style
- Data exchange format
- Process model

Related: Packaging Model

Component Platform

A component platform is **run-time incarnation** of the component model.

For example:



Component Platform

Platforms typically provide support for

Inter-component services

- binding
- interaction

Component lifecycles

- install, replace, remove

Extra-functional / resource aspects

- scheduling
- quality of service management
  - (dynamic) load balancing
  - (re)negotatiation
- security
- fault tolerance (replication)
- interoperabiliy (language/OS)

Component Platform

What 'features' does the infrastructure provide?

- (dynamic) load balancing / scheduling
- fault tolerance (replication)
- quality of service negotiation
- security
- heterogeneous platforms (language/OS)

Summary

- CBD aims to earn money through improving Productivity, Quality, Time-to-market, Maintenance
- There are many types of components because there are many different domains with different requirements on component-based systems.
  ➔ many component models ➔ many component platforms

Questions?

You should know
- an answer to 'What is a component?'
- what a component model is
- the relation between reuse, CBD, and OO

Summarizing, CBSE is about

Plug & Play: components can be composed with little effort – preferably at run-time

Interface-centric: components can be composed without knowing their implementation

Standardization: component can be manufactured by multiple vendors and widely reused across corporations

Distribution through Market: components can be acquired and improved though competition market, and provide incentives to the vendors

Architecture-centric: components may be designed on a pre-defined (product-line) architecture

Concluding Remarks 1

What's New in CBSE?

It deals with components that were not a-priori designed to be part of the same application

There will be multiple types of software components – belonging to multiple component models.

Concluding Remarks 2

CBD aims to

- increase productivity of SE
- change-ability of software systems

It is likely that every system will need some degree of custom development

- Wrapping is inevitable

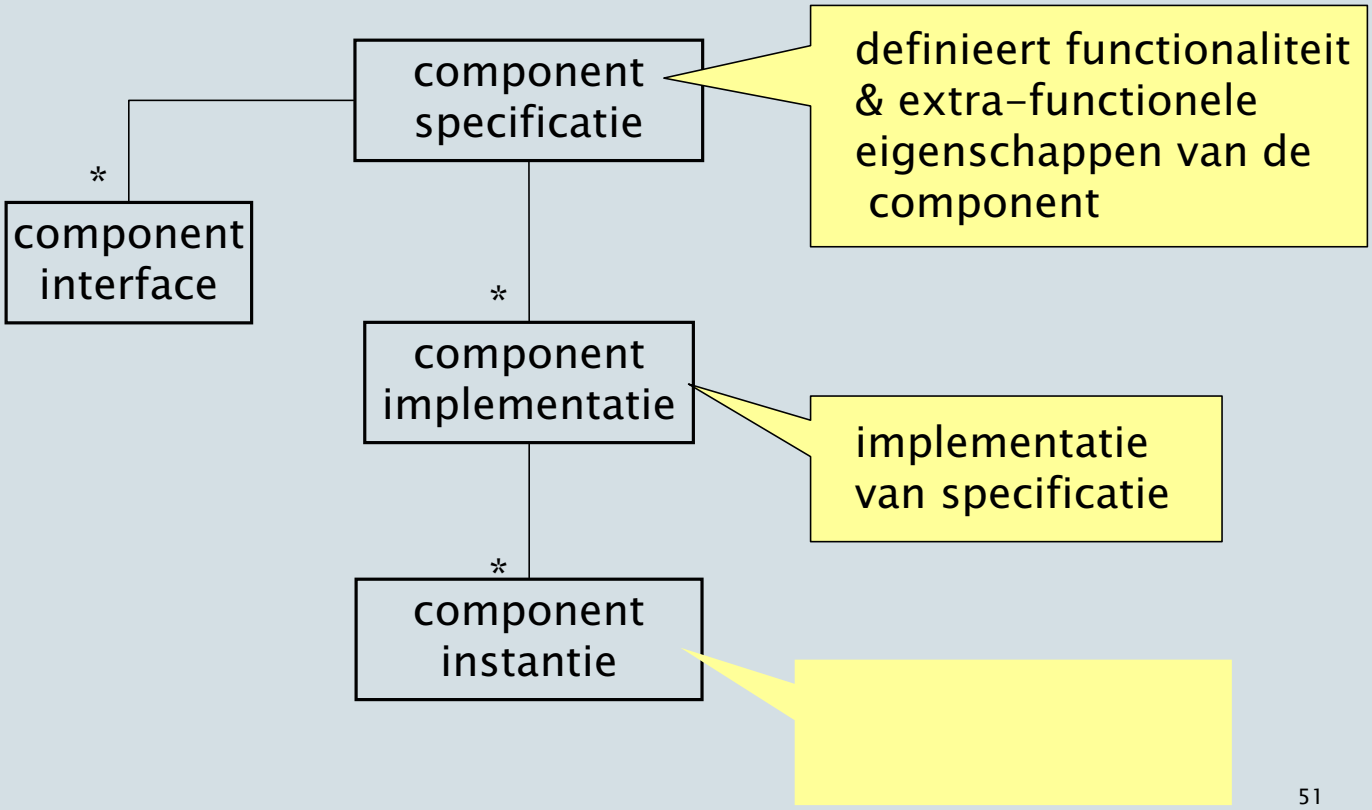
CBSE is about managing dependencies

- Is this not addressing the symptoms, rather than the disease?
- Q: Why not try to avoid dependencies in the first place?

Type of dependencies:

design time:	tooling
compile-time:	compiler, other components
run-time:	platform, other components

Useful Distinction




ARTES
SUMMER SCHOOL 2005

ENEA

Distribution vs. Fault Tolerance

Jan Lindblad
Enea Embedded Technology

Enea www.enea.com

- Founded 1968 by 4 KTH Students
- Headquarters in Kista, Stockholm
- Ca 500 employees, majority in Sweden
- Offices: 4 in Sweden, 4 in Europe, 5 in USA, 2 in Asia

- First Swedish UNIX system
- enea.se first domain in Sweden
- OSE family of operating systems
- Telecom dominates

OSE www.enea.com/ose

- Operating System for Embedded applications
  - Microkernel, message passing, distributed systems
  - File systems, IP-stacks, Program loader, Memory management, Virtual machines, … as plug-ins
  - PowerPC, ARM, MIPS, DSPs, 16-bit CPUs, …
- OSE kernel safety certified to IEC 61508-3
- Thousands of concurrent OSE processes
- Interrupt latency < 1µs on many processors

ENEA Embedded for leaders

Jan Lindblad jan . lindblad @ enea . se

- System Architect, System Manager at Enea Embedded Technology
- Built first computer at 12, designed & built sound card at 16, wrote first compiler at 17
- Married and three kids in villa outside Stockholm

Fault tolerance in distributed systems

Distributed and fault tolerant software have a lot in common. As applications are growing more widely distributed, issues like

- **fault tolerance**
- **network management**
- **security**

are becoming more and more relevant. This tendency is even more accentuated with the recent development of ad hoc networking technology, such as the Bluetooth standard.

In the Fault tolerance world

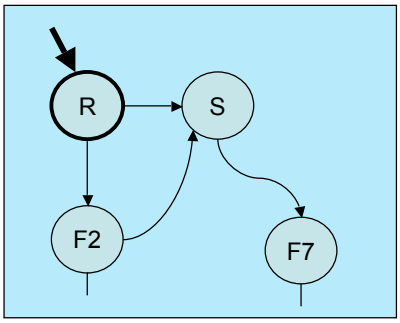
Fault tolerant design and programming is based on

- **Stable backwards recovery**
- **Single point of failure avoidance**
- **Simplicity**
- **Supervision**
- **Separation**

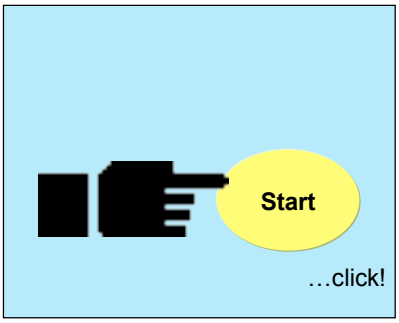
Let's see what they are and how many problems in the fault tolerant world are similar to those in distributed systems, and many of the solutions for fault tolerance are also applicable to distributed systems

Backwards Recovery

Process state diagram



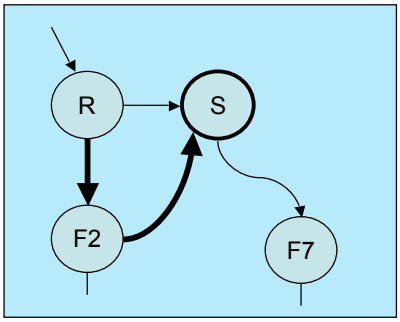
Action



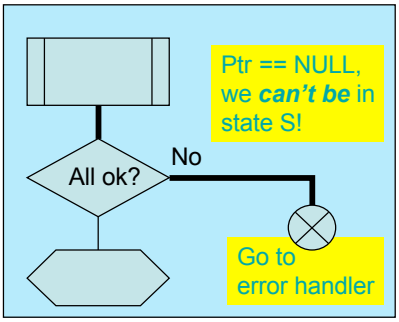
ENEA Embedded for leaders

Backwards Recovery

Process state diagram

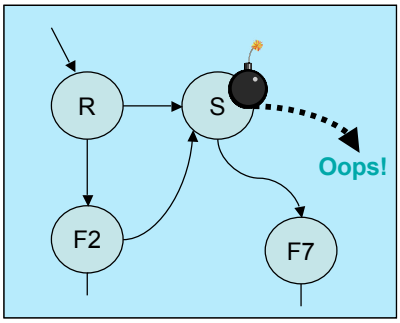


Code flow in state S



Backwards Recovery

Process state diagram



Error handler

A state error has been detected

We thought we were in state S, but we can't be -- we're lost!

There is no safe way back

ENEA Embedded for leaders

Backwards Recovery

When a state error has been detected, all you can be sure of (prove) is that something is wrong. There is no way to safely bring the current (unknown) state back to a known state. This is mathematically proven.

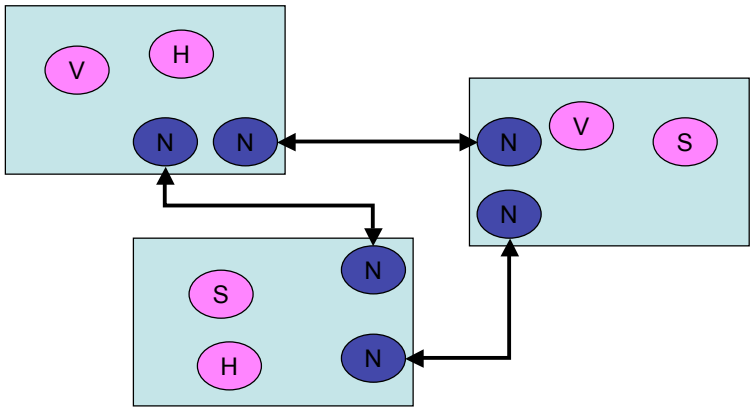
Therefore it's a good idea to "nuke" the whole context with the state problem, and all parts of the system that could be "infected" by this context. And do it fast, before the problem spreads.

Backwards Recovery

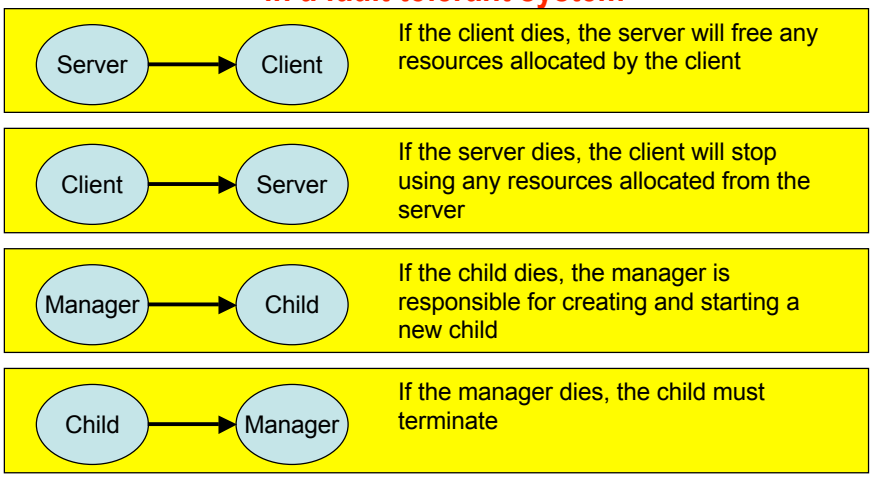
Most importantly: any cleanup that needs to be done when killing “infected” parts of the system must not be done by the “infected”!

In a distributed system, any cleanup that needs to be done when a connection is lost cannot be done by the remote process!

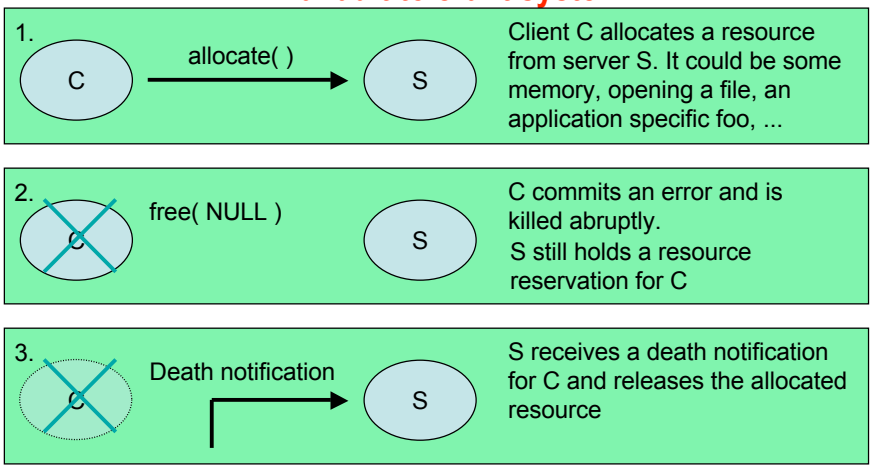
Single point of failure avoidance



Supervision in a fault tolerant system

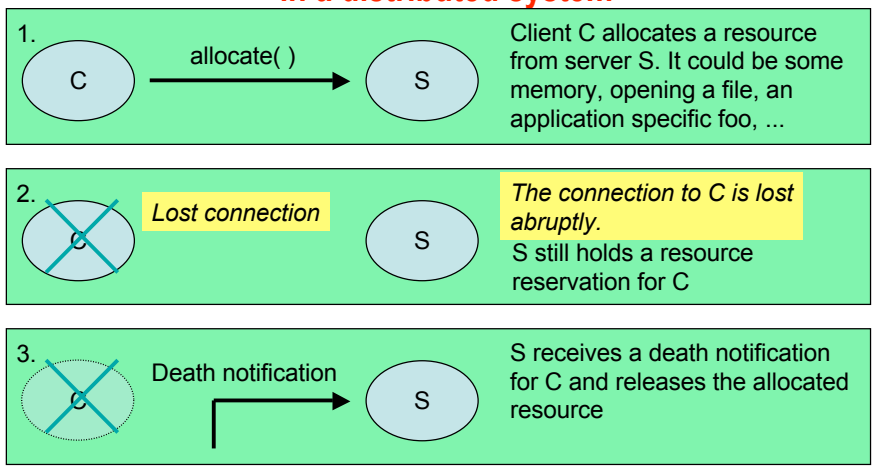


Supervision in a fault tolerant system



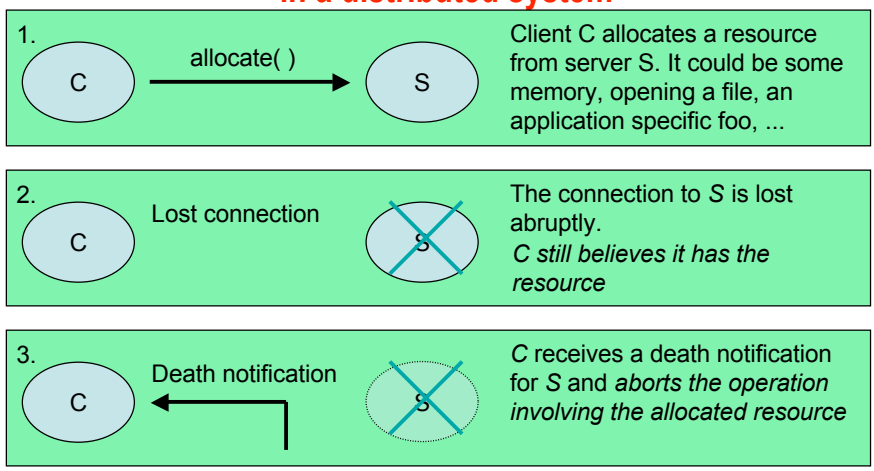
ENEA Embedded for leaders

Supervision in a distributed system



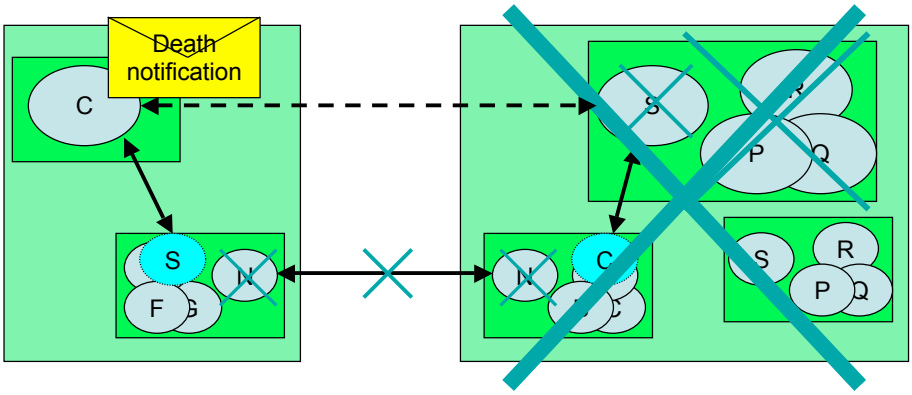
ENEA Embedded for leaders

Supervision in a distributed system



ENEA Embedded for leaders

Supervision in a distributed system



ENEA Embedded for leaders

Simplicity

Many small units that work together are much more likely to produce a good result than few large units.

Think of when you were building Lego models, many small and highly specialized parts give the best result

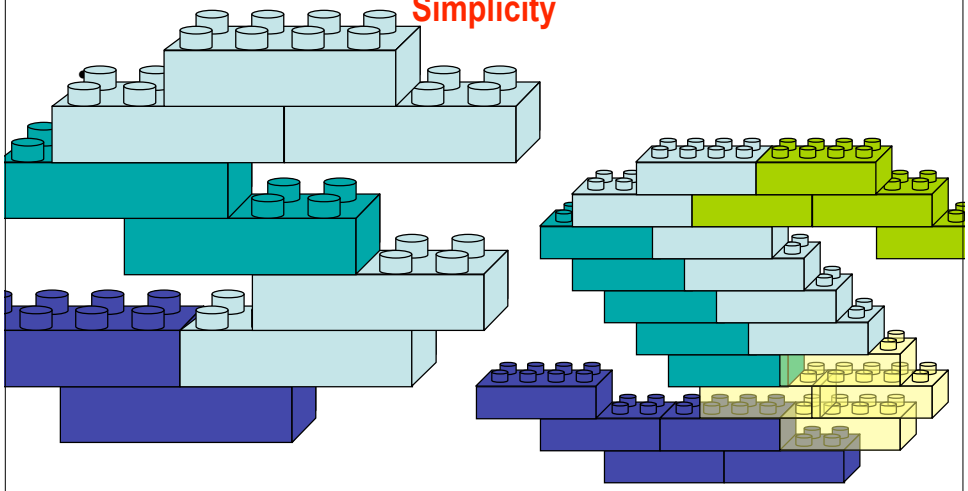
The complexity of a software module is exponential in it's size. This fact is very important in systems that have to be certified/verified/tested.

In distributed systems there is an added benefit, it's easier to create an optimal system configuration using small units, it's simply more configurable.

ENEA

Embedded for leaders

Simplicity

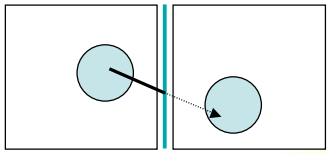


ENEA

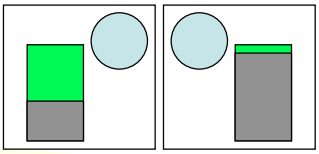
Embedded for leaders

Separation

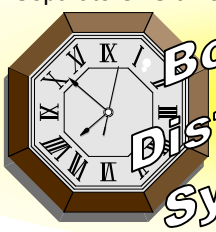
Memory protection



Separate memory pools



Separate CPU time



Bonus in Distributed Systems

Separate device drivers & interrupt sources

In the Fault tolerance world

We've seen some areas where the fault tolerant way of designing systems solve difficult problems of distributed systems

Let's stay in this environment and go on with looking at some other design issues of distributed systems

- **Interprocess communication (IPC)**
- **Different networking styles**

Interprocess Communication (IPC)

Even though the system needs separation, it also needs forms of communication across the boundaries

- **Messages**
  Perfect for local and transparent remote communication
  Kernel
- **Sockets**
  Standard, works both local and remote, a little slow
  Networking stack

Interprocess Communication (IPC)

- **Pipes**
  Possible locally, a little slow
  File system
- **Semaphores**
  No! Violates separation and will not work remotely
  Applications

ENEA Embedded for leaders

Interprocess Communication (IPC)

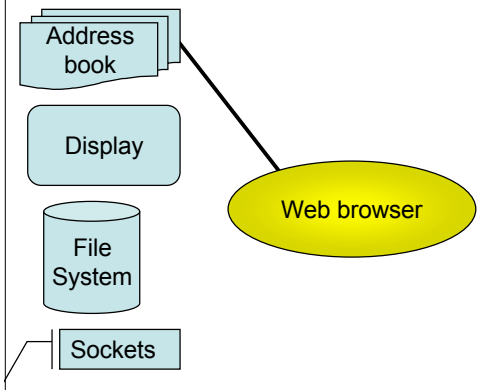
Let's take a closer look at what communication styles an ordinary application employs in various situations

We'll take a look at how a Web browser communicates with

- **the address book**
- **the display**
- **the file system**
- **the network**

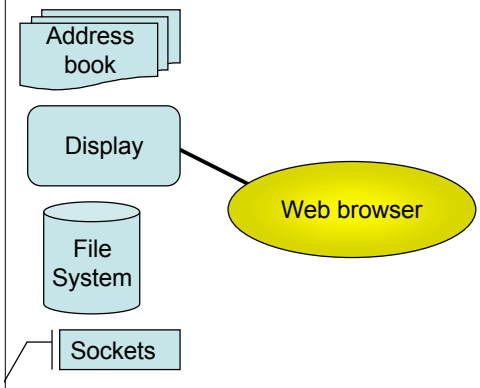
This will give some insights regarding how applications can be designed to be more easily distributed

Networking styles
address book



- Access: direct function calls into separate program module (.dll / .so)
- Not distributed

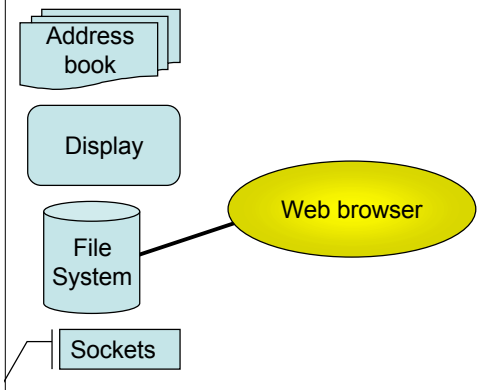
Networking styles
display



- Access: library calls
- Windows environment: Not distributed
- Unix (X-windows) environment: Distributed using sockets locally & remotely
- Application: nearly unaware

ENEA Embedded for leaders

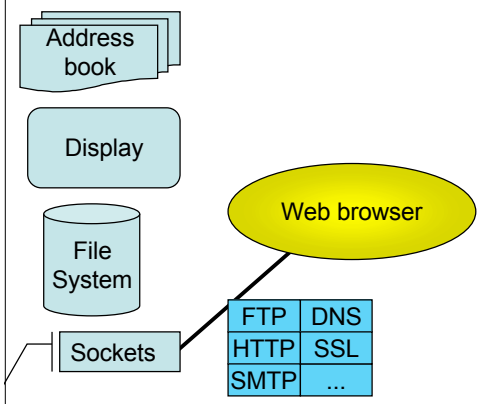
Networking styles
file system



- Access: library calls
- Access to local file systems is not distributed
- Access to remote file systems using a multitude of mechanisms, including sockets and SMB
- Application: totally unaware

ENEA Embedded for leaders

Networking styles
sockets



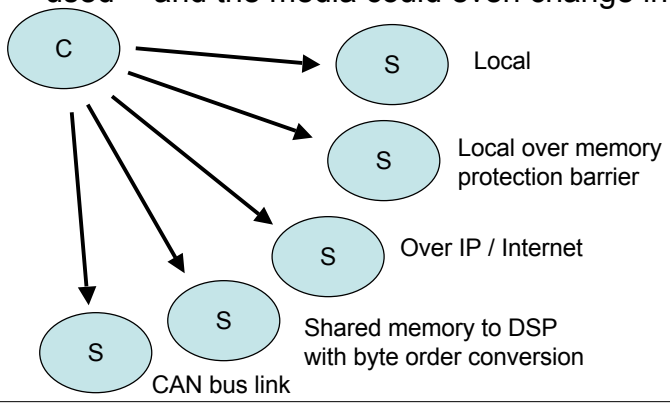
- Access: library calls
- Application sets up and manages connections and protocols
- Application: totally aware

Summary of networking styles

No distribution	direct function calls	**Shared library**
Explicit	applications fully aware of the network and have to handle many network issues. one distribution mechanism, which is used locally also	**Sockets**
Transparent unified	one distribution mechanism, which is used locally also	**X-windows**
Transparent diversified	several parallell distribution mechanisms, no distribution used locally	**File systems**

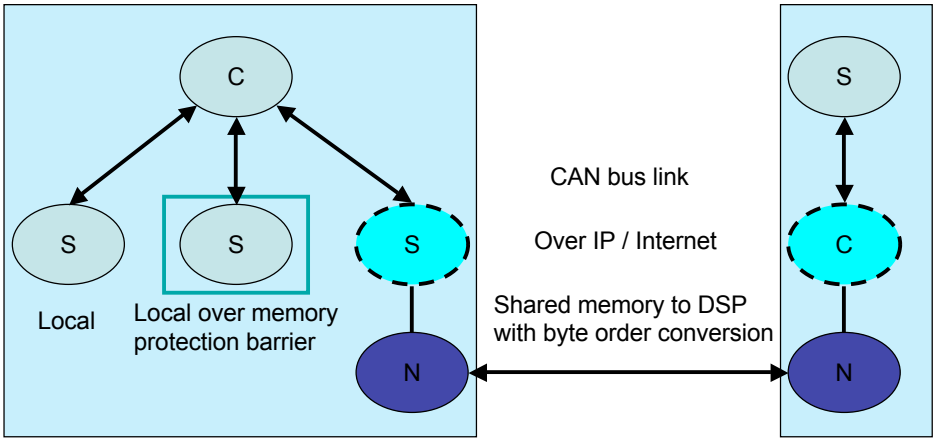
Embedded networking

Diversified transparent networking makes it possible for processes to communicate over a multitude of media, while staying unaware of what media is being used -- and the media could even change in runtime



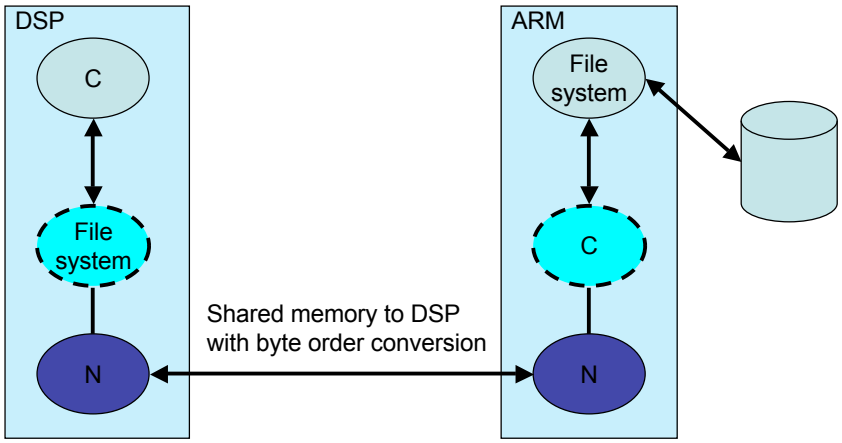
ENEA Embedded for leaders

Embedded networking



ENEA Embedded for leaders

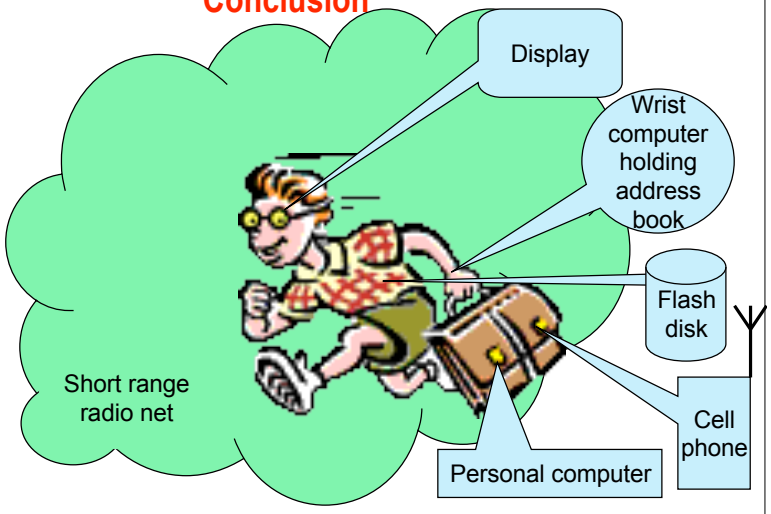
Embedded networking



ENEA

Embedded for leaders

Conclusion

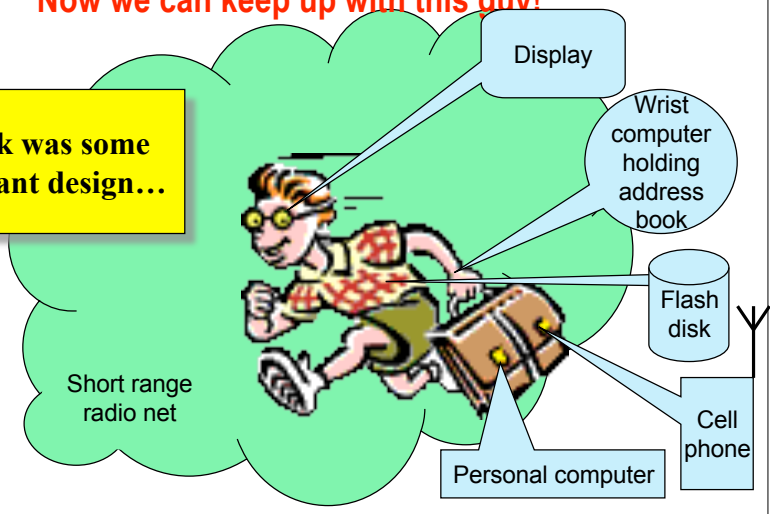


ENEA

Embedded for leaders

Now we can keep up with this guy!

All it took was some fault tolerant design…



ENEA

Overwhelming complexity

Jan Lindblad
Enea Embedded Technology

Introduction

Embedded software development cost is increasing steadily. Today, about half the cost consists of test and maintenance.

- Why is this?
  - Some important trends exposed
- What can be done about it?
  - Some suggestions to answers
  - Some suggestions for research

ENEA Embedded for leaders

Enea www.enea.com

- Founded 1968 by 4 KTH Students
- Headquarters in Kista, Stockholm
- Ca 500 employees, majority in Sweden
- Offices: 4 in Sweden, 4 in Europe, 5 in USA, 2 in Asia

- First Swedish UNIX system
- enea.se first domain in Sweden
- OSE family of operating systems
- Telecom dominates

ENEA Embedded for leaders

OSE www.enea.com/ose

- Operating System for Embedded applications
  - Microkernel, message passing, distributed systems
  - File systems, IP-stacks, Program loader, Memory management, Virtual machines, … as plug-ins
  - PowerPC, ARM, MIPS, DSPs, 16-bit CPUs, …

- OSE kernel safety certified to IEC 61508-3
- Thousands of concurrent OSE processes
- Interrupt latency < 1µs on many processors

Jan Lindblad jan . lindblad @ enea . se

- System Architect, System Manager at Enea Embedded Technology
- Built first computer at 12, designed & built sound card at 16, wrote first compiler at 17
- Married and three kids in villa outside Stockholm

ENEA Embedded for leaders

Real-time Research Projects

- Execution Time Analysis (ETA)
  - Measurement techniques
  - WCET by Abstract Interpretation
  - Evaluation of commercial WCET tools
- Automatic race condition detection
  - Whole system simulation in Simics
- Root cause analysis of reported bugs
  - Classification and countermeasures

ENEA

Software development cost increases

Why?

Current trends

MP3 player – Mobile phone – DVD player – Pacemaker – Car – Watch – Landline phone – Stereo – TV – Photo album – Coffee machine – Dishwasher – Indoor climate system – Traffic light control system – Paper mill – National energy transmission system – Nuclear power plant – A380 airliner – Ariane 5 rocket

Which of these contain embedded software?

1990?

2005?

2020?

Current trends

MP3 player – Mobile phone – DVD player – Pacemaker – Car – Watch – Landline phone – Stereo – TV – Photo album – Coffee machine – Dishwasher – Indoor climate system – Traffic light control system – Paper mill – National energy transmission system – Nuclear power plant – A380 airliner – Ariane 5 rocket

- Integration ⇒ Increased
- Communication Complexity
- Functionality & Size ⇒ Bugs
- Ecosystem
- Dynamic & Configuration ⇒ Reset button

ENEA Embedded for leaders

Functionality & Size

- Embedded = small?
  - OSE 5.1: 2 million SLOC
  - Ericsson Mobile platform: 3 million SLOC
  - Ericsson Cello platform > 50 million SLOC equiv
- Compare with desktop systems
  - RedHat 7.1: 30 million SLOC
  - Windows NT 5.0 20 million SLOC
  - Windows XP 2002: 40 million SLOC

Ecosystem

Embedded = all sources available?

- OSE: 4 sites, 62 official partners
- EMP platform: 30 software suppliers

We are not software houses,
we are integration houses!

ENEA Embedded for leaders

Dynamic & Configuration

- Plug and play (or pray)
- Runtime upgrades
- Linux kernel configuration

- Complex configuration/reconfiguration

Increased Complexity

MP3 player – Mobile phone – DVD player – Pacemaker – Car – Watch – Landline phone – Stereo – TV – Photo album – Coffee machine – Dishwasher – Indoor climate system – Traffic light control system – Paper mill – National energy transmission system – Nuclear power plant – A380 airliner – Ariane 5 rocket

- Higher cost per bug(?)
- More bugs per system ⇒ Increased cost
- More systems

Most bugs are low cost – no inquiry board – but they are many ⇒ Salami trick

ENEA Embedded for leaders

The Problem

- Large systems written by many developers around the world at different points in time in multiple languages
- Partial access to source code
- Applications communicating with the outside world, millions of configuration combinations, dynamically reconfiguring

➢ Typical program analysis: compiler warnings

Test & maintenance is only 50% of the development cost??

This is the class of systems that really need analysis tools

ENEA

How can we defeat the complexity?

ENEA

Embedded for leaders

Suggested attack routes

- We need to raise the abstraction level for systems programmers
  - Languages
  - Intentional programming
  - Contracts
- We need methods to analyze and test large systems
- We need the systems to handle unforeseen situations autonomously

Languages

- Every API is a language
- C is everywhere because of
  - Huge ecosystem (tools, engineers, code, …)
  - High expressiveness
  - Up to the programmer [to be efficient]
- But C is not particularly analyzable ☹

- UML? Java? Erlang? Ada?
  Academic languages?

Languages

- UML? Java? Erlang? Ada?
  Academic languages?

Maybe, but it's not mainly about that…

We need higher level language concepts like

- Parallel processes, components & contracts,
  error handling, recovery,
  intentional programming, …

Intentional programming

dict_get(key, dict)

⇒

dict_lookup(key, dict)

dict_search(key, dict)

dict_is_key(key, dict)

Joe Armstrong, “Making reliable distributed systems in the presence of software errors”

Contracts

- Hardware components have data sheets
  - Lists max and typical values
- What about software components?
- Reuse ⇒ contracts (Ariane 5)
- Eclipse Java IDE

Suggested attack routes

- We need to raise the abstraction level for systems programmers
- ➢ We need methods to analyze and test large systems
  - Components
  - Specification & program analysis
  - Test & simulation
- We need the systems to handle unforeseen situations autonomously

ENEA Embedded for leaders

Components

- Divide and conquer, one piece at a time
- Components need isolation and specification
- Analyze one component,
  use specifications for rest of system

- Key to reuse
- Writing usable specifications is hard,
  tool support needed

Specification & program analysis

- Dereferencing NULL is not good
- Stack overflow is not good
- Indexing out of bounds is usually not good
- Division by zero is not good

⇒

```
/*  @exec_time: 120 < T < 200 + 20*x->maxlen
    @stack_memory: 88 < M < 1520
    @precond: 0 < y < x->maxlen
    @postcond: return 0..65535 */
uint16_t foo_lookup(FX *x, uint32_t y) { …
```


Test & simulation

- Cannot test every possible
  - Hardware platform
  - Configuration
  - Interleaving
  - Error situation
- Need tools that can handle abstract configurations, hardware, interleavings, …
  Like abstract values in abstract interpretation

Suggested attack routes

- We need to raise the abstraction level for systems programmers
- We need methods to analyze and test large systems
- ➢ We need the systems to handle unforeseen situations autonomously
  - Backwards recovery
  - Supervision
  - Seamless distribution

ENEA Embedded for leaders

Backwards recovery

When a state error has been detected, all you can be sure of (prove) is that something is wrong. There is no way to safely bring the current (unknown) state back to a known state.

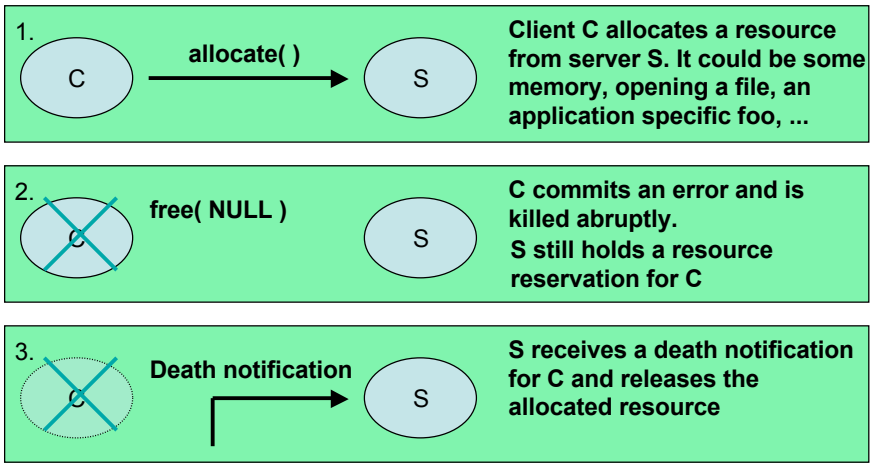
Therefore it's a good idea to "nuke" the whole context with the state problem, and all parts of the system that could be "infected" by this context. And do it fast, before the problem spreads.

Backwards recovery

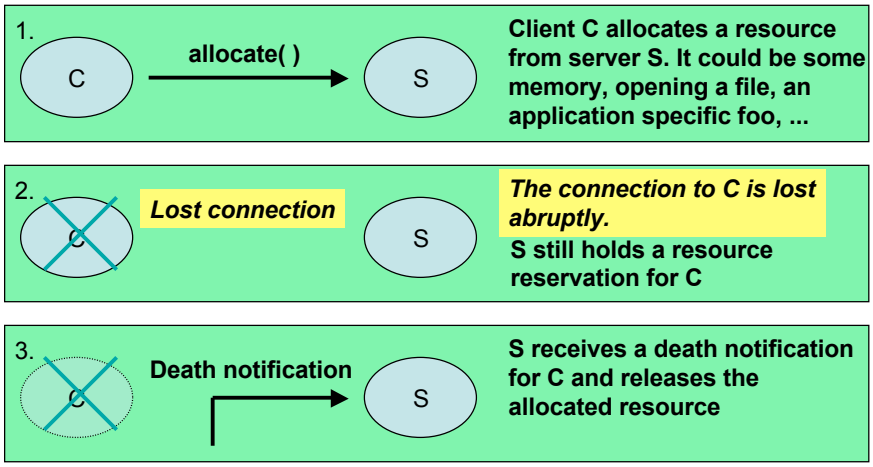
Most importantly: any cleanup that needs to be done when killing "infected" parts of the system must not be done by the "infected"!

- Processes can die at any time ⇒
  - Semaphores or mailboxes cannot be used
  - Cleanup by resource manager (server), not by process itself (client)
  - Process supervision required

Supervision



Seamless distribution



Supervisors

Just as in a business organization, a supervisor:

- Has responsibility for a function
- Knows what should be done
  (may change over time, e.g. plug and play)
- Manages, but does not participate in, the actual work
- Has to deal with any failures of the workers
  (restart, service degradation, …)
- Is terminated if there are persistent problems with the function

Supervisors

Likewise, a worker:

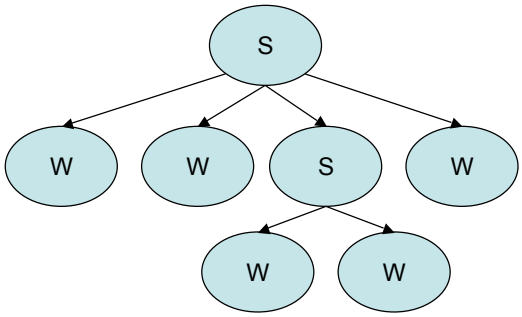
- Is hired/created by the supervisor
- Gets a task to execute and parameters for the task from the supervisor
- Reports problems with the task to the supervisor
- Is fired/terminated by the supervisor
- May be a supervisor itself with respect to its own workers

Supervision tree

Supervisors …

- (Re-)start
- (Re-)configure
- Recover from problems with
- Shutdown

… the workers



ENEA

Conclusions

Example 1: Telecom

- "The world's largest machine"
- Very distributed and dynamic
- Huge ecosystem
- Lot's of bugs remain in shipping systems
- Fault tolerant design

- Very few analysis tools used
- Real-time design by stomach or pen+paper

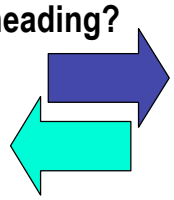
Example 2: Automotive

- A few dozen ECUs
- Centralized distribution and little dynamics
- Small centralized ecosystem
- Not fault tolerant – “all faults eliminated”
- Complete system design, analysis, testing

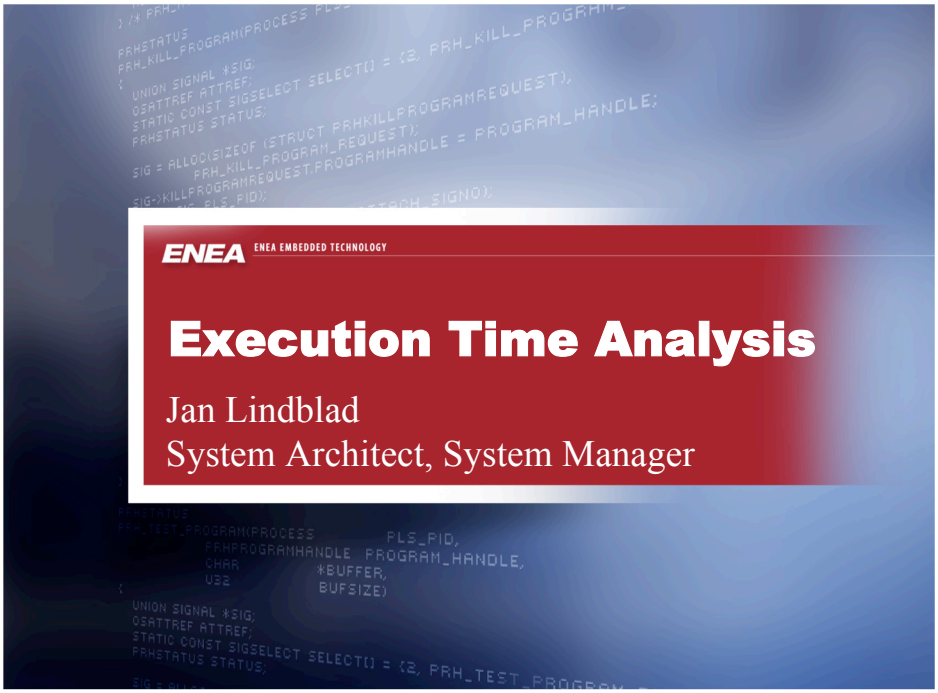
- Several different analysis tools actually used
- Real-time design by proven methods & tools

ENEA Embedded for leaders

Where are we heading?

Telecom  Automotive

- Very distributed
- Huge ecosystem
- Bugs remain in shipping systems
- Fault tolerant design
- Several different analysis tools actually used
- Real-time design by proven methods & tools
- Components



ENEA ENEA EMBEDDED TECHNOLOGY

AGENDA

- The Timing Challenge
  - The Challenge
  - Measurement or Calculation?
  - Wanted!
- Execution Time Analysis
- Program Analysis

ENEA ENEA EMBEDDED TECHNOLOGY

The Challenge

All of us are designing software with requirements on

- Parallelism
- Real-Time
- High Performance

Most of the difficult problems with our software concerns the timing behavior.

3

ENEA ENEA EMBEDDED TECHNOLOGY

The Challenge

How do we make sure that…

- Parallelism
  - … there are no race conditions or deadlocks?
- Real-Time
  - … we are always finished before every deadline?
- High Performance
  - … the average execution time is short enough?
  - … we optimize the code where it makes a difference?
  - … we use algorithms that scale well?

4

ENEA ENEA EMBEDDED TECHNOLOGY

Race conditions

Modern classification of bugs:

– Bohrbugs

– Heisenbugs

Most race conditions and deadlocks are never found

– Heisenbugs generally cannot be found by testing

5

ENEA ENEA EMBEDDED TECHNOLOGY

Performance

Numbers to search for:

– Execution time
  - System call, context switch times (max, average)
  - Interrupt latency (max, average, jitter)
  - Various application operations (max, average)
  - …

– Complexity analysis
  - Ordo classes

6

ENEA ENEA EMBEDDED TECHNOLOGY

Performance

Execution Time (ET) depends on
- compiler
- processor

Ok, just a matter of counting cycles in the binary code.

… or ?

7

ENEA ENEA EMBEDDED TECHNOLOGY

Performance

Execution Time (ET) depends on
- compiler
- processor
- pipeline
- cache (what is the worst realistic cache state?)
- input (what is the worst input?)
- network, bus, switch fabric delays

The worst case is not interesting anyway.

8

ENEA ENEA EMBEDDED TECHNOLOGY

Finding the Execution Time

In theory

The problem to find the execution time of a program is **undecidable**.

– If you could tell the execution time of every program, you would also have solved the halting problem.

In practice

Two main approaches are used:

– Measurement on a real implementation

– Calculation on a model of the system

9

ENEA ENEA EMBEDDED TECHNOLOGY

Measurement or Calculation?

- WCET = Worst Case Execution Time
- BCET = Best Case
- ACET = Average Case

10

ENEA ENEA EMBEDDED TECHNOLOGY

Measurement or Calculation?

MEASUREMENT	CALCULATION
– Measurement samples points inside the curve. – By definition unsafe. Need to add "enough" safety margin. – Input dependent. Model of typical or worst input needed.	– Calculation finds a point outside the curve. – By definition safe. Overestimates, but how much? – Model of system needed.

11

ENEA ENEA EMBEDDED TECHNOLOGY

Measurement or Calculation? Sources of Misestimation

MEASUREMENT	CALCULATION
Other input than used in test may generate much longer execution times. – The program may not even complete its execution.	Simplistic system model may give magnitudes of overestimation. – Pipeline: 2-8x – Cache: 10-40x

12

ENEA ENEA EMBEDDED TECHNOLOGY

Wanted!

Automatic tool that analyzes the timing aspects of my system, pointing out improvement areas:

- Deadline budget or system performance goals
  - Where to optimize?
  - What parts have large variance in execution time?
  - What parts scale badly?
  - What parts were too complex for high quality analysis?
  - How much better/worse did it get after the latest change?

13

ENEA ENEA EMBEDDED TECHNOLOGY

Wanted!

Automatic tool that analyzes the timing aspects of my system, pointing out timing related software errors:

- Locking errors
  - Possible race conditions, deadlock, live-lock
  - Failure to acquire or release a lock
- Control flow problems
  - Infinite loops
- Priority problems
  - Starvation, priority inversion

14

ENEA ENEA EMBEDDED TECHNOLOGY

Wanted!

Automatic tool that can work under real conditions:

- Multiple source languages
  - C, C++, assembler, UML, SDL, SCADE, ...
- Multiple source providers
  - Application developer, operating system, 3rd parties, ...
- Multiple tool chains
  - Each source provider may have own tool chain

15

ENEA ENEA EMBEDDED TECHNOLOGY

Wanted!

Automatic tool that can work under real conditions:

- Modern processors
  - Caches, superscalar pipelines, hierarchy of buses
- Optimizing compilers
  - Analyzing the actual running code
- Dynamic programs
  - Loops with data dependencies, function pointers, …

16

ENEA ENEA EMBEDDED TECHNOLOGY

AGENDA

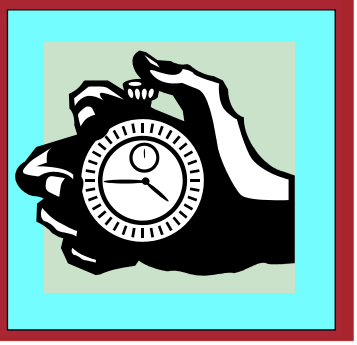
- ✓ The Timing Challenge
- Execution Time Analysis
  - Measurement
  - WCET
  - pWCET
- Program Analysis

17

ENEA ENEA EMBEDDED TECHNOLOGY

Measurement

- Observations on the actual system
  - Can be automated
- Generation of relevant input
  - Usually extremely large input domain
  - Plus all possible timings of input
  - Input include interrupts, clock signals, bus traffic, …
- Collection of data
  - Usually intrusive in some way



ENEA ENEA EMBEDDED TECHNOLOGY

Measurement Results

WCET=48.6us (in our test runs)
AverageET=12.8us (in our test runs)

+

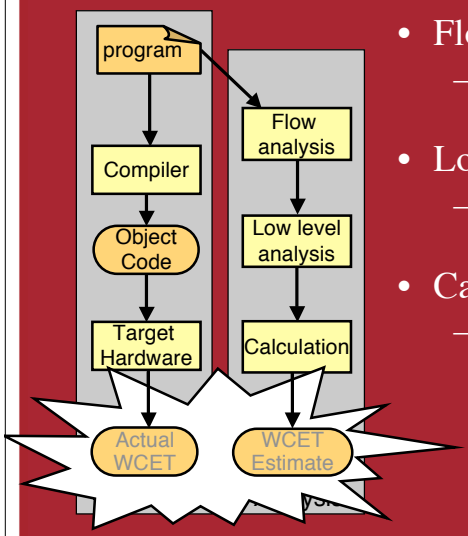
"We used an ARM920 processor running at 120 MHz, with a 32-bit bus at 33 MHz. The cache … The code was run 100 times, and the average and maximums were noted."

User always has other, different conditions

19

ENEA ENEA EMBEDDED TECHNOLOGY

Automatic WCET Estimation

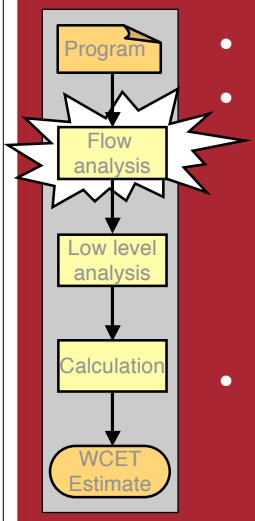


- Flow analysis
  - Determine the dynamic behavior of the program
- Low level analysis
  - Determine execution time for the program parts on the hardware
- Calculation
  - Combine flow and low-level times to give a WCET estimate

20

ENEA ENEA EMBEDDED TECHNOLOGY

WCET Flow Analysis



- Dynamic behaviour of program
- Example of info determined:
  - Number of loop iterations
  - Recursion depth
  - Input dependencies
  - Infeasible paths
  - Function instances
- Provided by static analysis and/or manual annotations

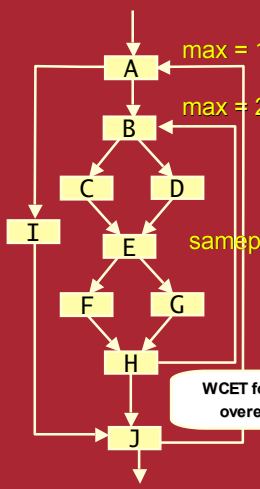
21

ENEA ENEA EMBEDDED TECHNOLOGY

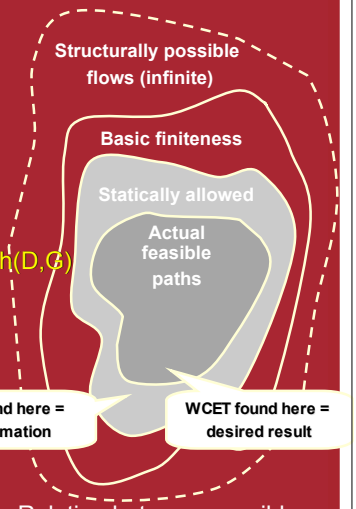
WCET Flow Analysis

```
do
{
   if(...)          // A
      do
      {
         if(...)          // B
            ...           // C
         else
            ...           // D
         if(...)          // E
            ...           // F
         else
            ...           // G
      } while(...)       // H
   else
      ...                // I
} while(...)             // J
...
```

Example program



Basic block graph

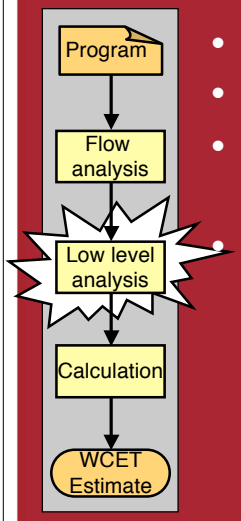


Relation between possible executions and flow info

22

ENEA ENEA EMBEDDED TECHNOLOGY

WCET Low Level Analysis



- Determine execution time for program parts
- Account for hardware effects
- Work on object code
  - The "real" program
- Two main issues:
  - Cache analysis (global)
  - Pipeline analysis (local)

23

ENEA ENEA EMBEDDED TECHNOLOGY

WCET Low Level Analysis

```
fib(int n)
{
  int
  i,Fnew,Fold,temp,
  ans;
  Fnew=1; Fold=0;
  for(i=2;i<=n;i++)
    {
      temp = Fnew;
      Fnew = Fnew
  + Fold;
      Fold = temp;
    }
    ans = Fnew;
  return ans;
```

Each block will run as a unit

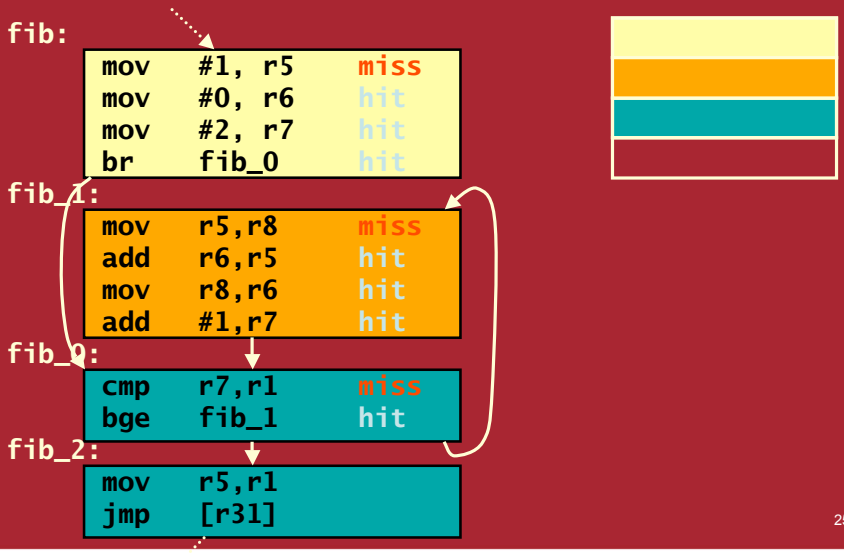
Flows as edges

```
fib:    mov   #1, r5
        mov   #0, r6
        mov   #2, r7
        br    fib_0
fib_1:  mov   r5,r8
        add   r6,r5
        mov   r8,r6
        add   #1,r7
fib_0:  cmp   r7,r1
        bge   fib_1
fib_2:  mov   r5,r1
        jmp   [r31]
```

24

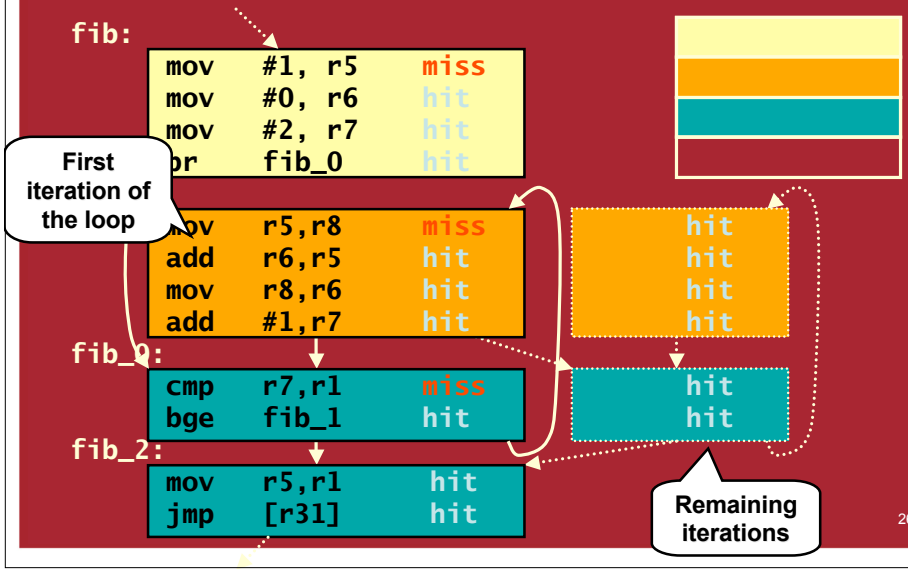
ENEA ENEA EMBEDDED TECHNOLOGY

WCET Cache Effects



ENEA ENEA EMBEDDED TECHNOLOGY

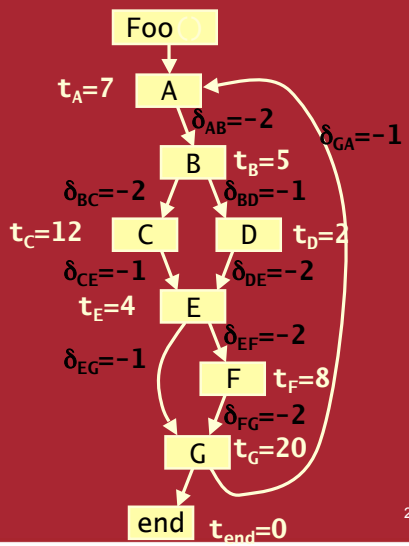
WCET Cache Effects



ENEA ENEA EMBEDDED TECHNOLOGY

WCET Pipeline Effects

```
    foo(x):
A:
  loop(i=1..100)
B:      if (x > 5)
  then
C:        x = x*2
        else
D:        x = x+2
        end
E:      if (x < 0)
  then
F:        b[i] =
  a[i];
        end
G:      bar (i)
```



27

ENEA ENEA EMBEDDED TECHNOLOGY

WCET Flow Information

- WCET= max $\Sigma(x_{entity} * t_{entity})$
  - Where x_{entity} satisfies all constraints
- Constraints:
  - Start condition
  - Program structure
  - Loop bounds
  - Other flow information

$X_{foo}=1$

$X_A=X_{fooA}+X_{GA}$

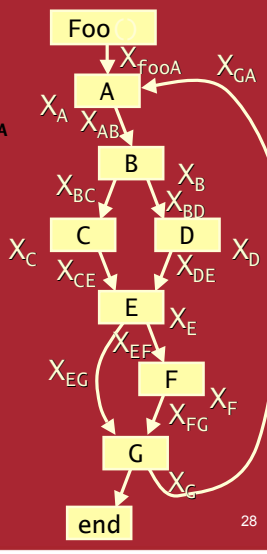
$X_{AB}=X_A$

$X_{BC}+X_{BD}=X_B$

$X_E=X_{CE}+X_{DE}$

$X_A<=100$

$X_C+X_F<=X_A$



28

ENEA ENEA EMBEDDED TECHNOLOGY

WCET Result

- Integer Linerar Programming equation solver:

WCET=4806

in the model

29

ENEA ENEA EMBEDDED TECHNOLOGY

WCET Tools

- Mälardalen University + Uppsala University
  - Flow analysis
  - Low level analysis
  - Integration
- AbsInt GmbH
  - AirBus is a customer
- Tidorum OY
  - BoundT tool

30

ENEA ENEA EMBEDDED TECHNOLOGY

Enea WCET Experience

```
Foo()
{
  alloc(...) // System call
       inside alloc()
       if(error_check_mode)
         do_lengthy_memory_check()
         if(corrupt_memory)
           error("memory_corrupt")
                inside error handler
                if(corrupt_memory)
                  reboot()
                else if(...)
                  ...
                return
```

Our results were

- Completely correct
- Completely irrelevant

⇒ Flow analysis required

Program flow should end here, but tool follows a longer path

ENEA ENEA EMBEDDED TECHNOLOGY

pWCET

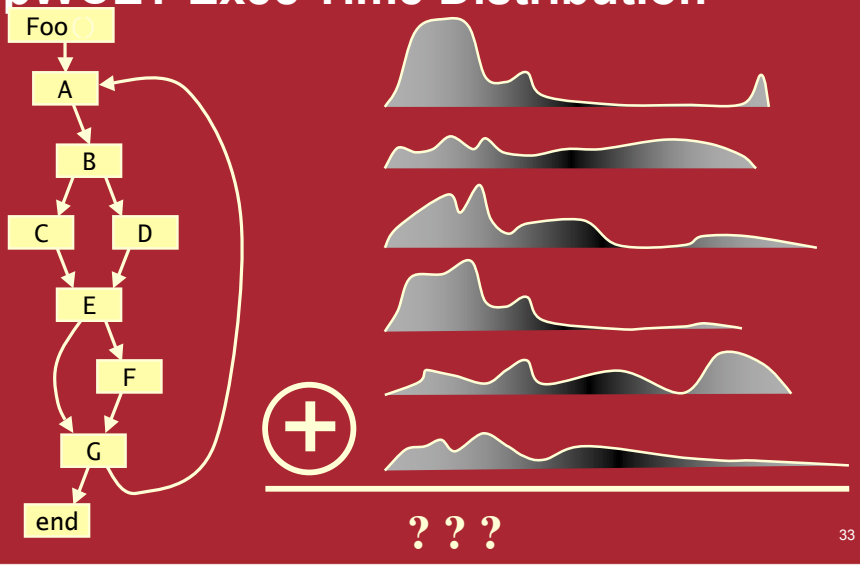
” The best model of the processor is the processor itself ”

- Let’s *measure* the execution time
  - But for small, small simple pieces separately
  - Gives distribution of execution times
  - Combine the small pieces!

32

ENEA ENEA EMBEDDED TECHNOLOGY

pWCET Exec Time Distribution

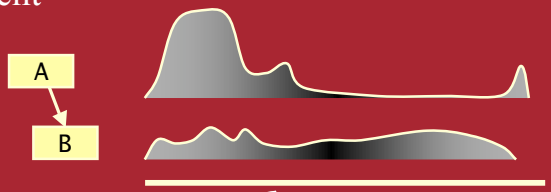


33

ENEA ENEA EMBEDDED TECHNOLOGY

pWCET Sequence

Case 1: Assume execution times of A and B independent



Convolution $H(z) = \int_x F(x)\, G(z - x)\, dx$

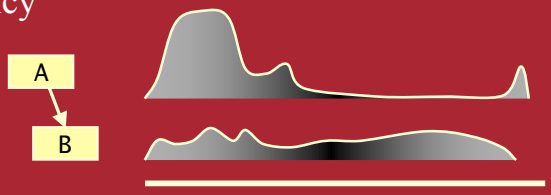
Usually this is not a feasible assumption

34

ENEA ENEA EMBEDDED TECHNOLOGY

pWCET Sequence

Case 2: Assume A and B dependent, known dependency



$$H(z) = \int_{x+y=z} j(x,y)$$

$$J(x,y) = p[X \leq x,\ Y \leq y]$$

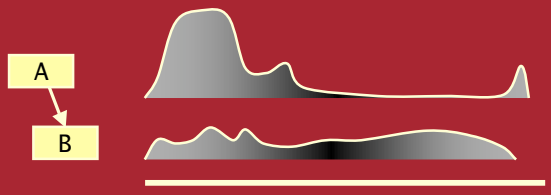
Usually the dependency is not known

35

ENEA ENEA EMBEDDED TECHNOLOGY

pWCET Sequence

Case 3: Assume A and B dependent, unknown dependency



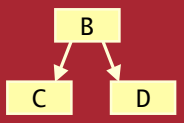
$$H(z) = \int_{x+y=z} \frac{\partial^2 \min(F(x), G(y))}{\partial x\, \partial y}$$

Assumes the worst possible dependency (conmonotonic)

36

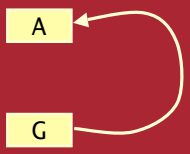
ENEA ENEA EMBEDDED TECHNOLOGY

pWCET Control Flow



If B then C else D

$Z = B + \max(C, D)$



For i=1..N

$Z = G + n(A..G)$

Single basic block

Z = Measure!

37

ENEA ENEA EMBEDDED TECHNOLOGY

pWCET Result

38

ENEA ENEA EMBEDDED TECHNOLOGY

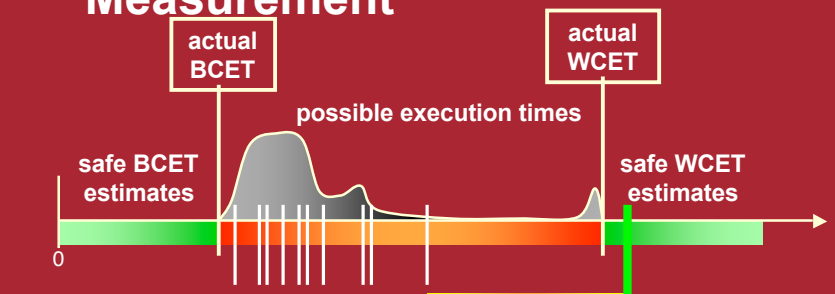
pWCET Tools

- University of York
  - www.rapita.com
- Enea Epact

39

ENEA ENEA EMBEDDED TECHNOLOGY

Execution Time Analysis Measurement



10 observations

How much margin?

40

ENEA ENEA EMBEDDED TECHNOLOGY

Execution Time Analysis WCET

How to build a model?

(and is the model correct?)

41

ENEA ENEA EMBEDDED TECHNOLOGY

Execution Time Analysis pWCET

10,000 micro observations

Measure what input??

Select your level of confidence

42

ENEA ENEA EMBEDDED TECHNOLOGY

Future Tools

Many research groups are working on execution time analysis in EU, and Sweden especially

- Current effort to standardize interfaces between execution time analysis tools
  - Different analysis methods will help each other

43

ENEA ENEA EMBEDDED TECHNOLOGY

AGENDA

- ✓ The Timing Challenge
- ✓ Execution Time Analysis
- Program Analysis
  - Stack Usage
  - Reverse Engineerning the Design
  - Simulation

44

ENEA ENEA EMBEDDED TECHNOLOGY

Stack Usage

aiCall from AbsInt is a call graph analyzer

- See the call graph
- See the maximum stack depth used
  - Use this information to configure OSE processes

aiPop from AbsInt is a code compressor

- Rewrites object code into functionally equivalent, but more compact object code

45

ENEA ENEA EMBEDDED TECHNOLOGY

Reverse Engineering the Design

The ASTEC remodelling project works on a tool to

- Extract the design of a system from code.
  Use this information to
  - Understand the system
  - Check that the design is what you think it is
  - Verify that changes to the code don't change the design in negative ways

46

ENEA ENEA EMBEDDED TECHNOLOGY

Simulation

Simics from Virtutech is a CPU+System simulator

- Provides a deterministic and non-intrusive observation environment. Use it to
  - Measure execution time
  - Find race conditions and deadlocks
    - Current project to intelligently insert delays/breakpoints to change the timing behaviour at sensitive spots.
  - Find reason for 'glitches'
    - Put a breakpoint on missed frame/lost call.

47

ENEA ENEA EMBEDDED TECHNOLOGY

AGENDA

- ✓ The Timing Challenge
- ✓ Execution Time Analysis
- ✓ Program Analysis

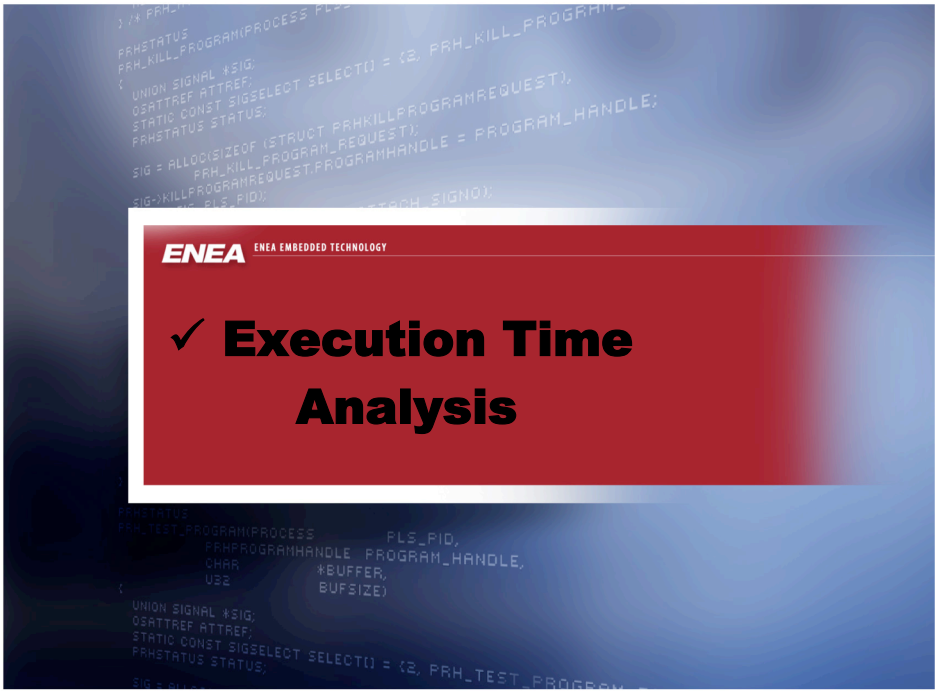
48

ENEA ENEA EMBEDDED TECHNOLOGY

Information Sources

- Mälardalen University, M. Real-Time Research Center
- University of York, Real-Time Systems Research Group
- Uppsala University, Advanced Software Technology Group
- Swedish institute of Computer Science (SiCS)
- Virtutech AB
- Rapita Systems Ltd
- AbsInt GmbH
- Tidorum OY, BoundT
- Enea Epact, Embedded Technology AB

49




ARTES
SUMMER SCHOOL 2005

ARTES summerschool 2005

Invitation, Programme, Skövde information, SNART, Poster instruction, Travel

Programme

- 15-16 Bus to Läckö castel
- 16-17 Free strolling around and possibility to see exhibitions.
- 17-17.40 Guided tour of the castel
- 18-22 Boat to Navens Light house inkl. BBQ (Bring your swimssuit in case the weather is varm.)
- 22-23 Bus to Skövde

Updated: 09-Aug-2005 15:42
Location: http://www.artes.uu.se/events/summer05/social_activity.shtml


ARTES
SUMMER SCHOOL 2005

Tutorial: Experimental Sensor Networking Research

Thiemo Voigt and Joakim Eriksson
Swedish Institute of Computer Science
{thiemo,joakim}@sics.se

August 9, 2005

Wireless sensor networks (WSN) consist of a potentially large number of tiny nodes equipped with sensing facilities, a low power processor, limited memory, and a radio module that enables the nodes to form networks through which sensed data can be transported to a base station. Applications range from tornado detection and environmental science, to industrial process monitoring, ventilation control and intrusion detection systems.

While analysis and simulation of sensor networks are indispensable, experimental research and the deployment of prototype or real sensor networks are at least of equal importance. In this tutorial, we present some of the research conducted at the Swedish Institute of Computer Science in the area of wireless sensor networks. Our work includes some theoretical aspects but the main focus is on experimental work in the areas of experimental evaluation of lifetime bounds for WSNs, TCP/IP support for sensor networks, the Contiki operating system and the development of prototype networks with industry partners.

More information about the WSN research at SICS in general can be found at http://www.sics.se/sensornets/. Most of the work presented in this tutorial has been conducted within the VINNOVA-sponsored DTNSN-project, see http://www.sics.se/cna/dtnsn/. The demonstrator shown as part of the tutorial has mostly been developed within the EFFWSN project financed by FMV.


ARTES
SUMMER SCHOOL 2005

The future for IT,

Karl-Einar Sjödin,
Vinnovas unit for information and communication research,

08-473 31 13
Karl-Einar.Sjodin@VINNOVA.se

Teknisk Framsyn är ett nationellt projekt som syftar till att skapa insikt och visioner om teknikutvecklingen på lång sikt. Rapporterna kan hämtas i digital form på projektets hemsida: www.tekniskframsyn.nu.

Inspiration till innovation, 2004
Vägval för Sverige, 2004

❞ *Vårt senaste århundrade åstadkom fler förändringar än under de föregående ett tusen åren tillsammans. Det kommande århundradet kommer att överträffa vårt föregående århundrade.* ❞

HG Wells vid en föreläsning över temat upptäckten av framtiden i London 1902.

http://www.cordis.lu/ist

http://www.cordis.lu/ist/grids

Sverige måste våga välja väg!

Karl-Einar Sjödin, VINNOVA
Teknisk Framsyn

Skövde 2005-08-19

Teknisk Framsyn vill ...

- Stärka det framtidsinriktade arbetet i företag och organisationer
- Främja hållbar tillväxt och förnyelse i Sverige
- Skapa underlag och processer för att prioritera områden som bygger upp svensk kompetens. Vad är vi egentligen riktigt bra på i Sverige?

Fokus: 15–20 år framåt

Projektet Teknisk Framsyn

- Nationellt projekt som analyserar svenska förutsättningar för teknisk och ekonomisk tillväxt
- Ungefär hundra experter från olika delar av samhället pekar på de viktigaste framtidsfrågorna – vägval för Sverige
- Projektet visar på förväntade teknologiska genombrott
- Projektet lyfter fram samspelet mellan teknik, samhälle och olika institutioner

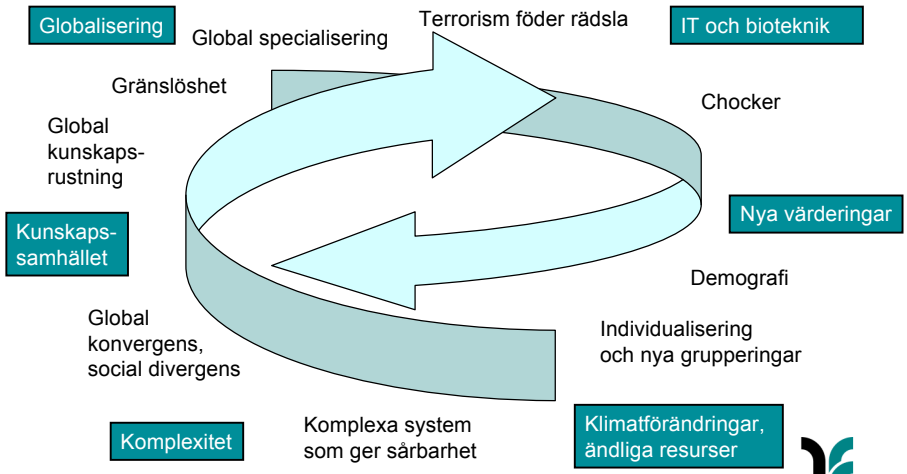
Huvudbudskap

- Välj väg – vi står inför sex vägval där beslut krävs *nu* för vår framtida konkurrenskraft
- Prioritera inom forskning och utveckling – vi föreslår 11 mångvetenskapliga teknikområden
- Utveckla för framtiden – även i små företag
- Samarbeta – Sverige står inte ensamt
- Förankra en vision som styr besluten i både näringsliv, forskning och politik

Starka, globala drivkrafter tränger sig på

antingen vi vill eller inte!

Drivkrafternas virvelvind



Våra viktigaste val:
vi vill ...

- att Sverige ska vara en del av världen
- kraftsamla för framtidens infrastruktur
- ta vara på människors resurser

och vi måste våga ...

- modernisera det offentliga åtagandet
- ta steget mot det hållbara samhället
- prioritera och fokusera

Teknisk Framsyn

6. Vi måste våga prioritera och fokusera, genom att ...

- Satsa på FoU, men satsa smart – inom 11 mångvetenskapliga teknikområden
- Skapa mer förädling – prioritera verksamhet långt fram i värdekedjan
- Specialisera regionalt
- Skapa mylla för nya företag
- Våga välja bort!

Teknisk Framsyn

Vad har vi valt bort?

Exempel:

- Optiska datorer
- Glasmetall
- Kiselkarbid
- Samhällsplanering
- Människans åldrandeprocess
- Funktionella textilier
- Fiberoptik
- Högspänning
- Nätverksdatorer
- Akustik
- Funktionell mat

Teknisk Framsyn

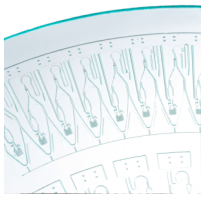
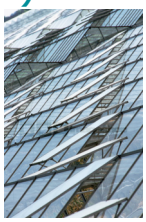
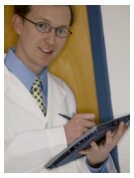
Ett nytt synsätt:

11 mångvetenskapliga teknikområden

Teknisk Framsyn

Mångvetenskapliga teknikområden med stor svensk potential (1)

- Säkrare komplexa system
- Mekaniska system och strukturer
- Interaktiv teknik
- Funktionella material

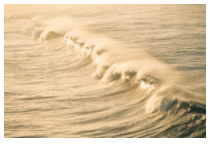





Teknisk Framsyn

Mångvetenskapliga teknikområden med stor svensk potential (2)

- Miljö- och livscykelteknologi
- Rörlig energiförsörjning
- Fasta energisystem





Teknisk Framsyn

Mångvetenskapliga teknikområden med stor svensk potential (3)

- Säkerhet och skydd
- Hållbar livsmedelsproduktion
- Tillgänglig IT
- Hälsa och sjukvårdsteknik




Teknisk Framsyn

Säkrare komplexa system

Exempel:

- Öppna standarder
- Simuleringsteknik
- Teknik för trafiksäkerhet
- Automatisering
- Datasäkerhet, inkl. programvaruteknik

Teknisk Framsyn

Våga fatta beslut!

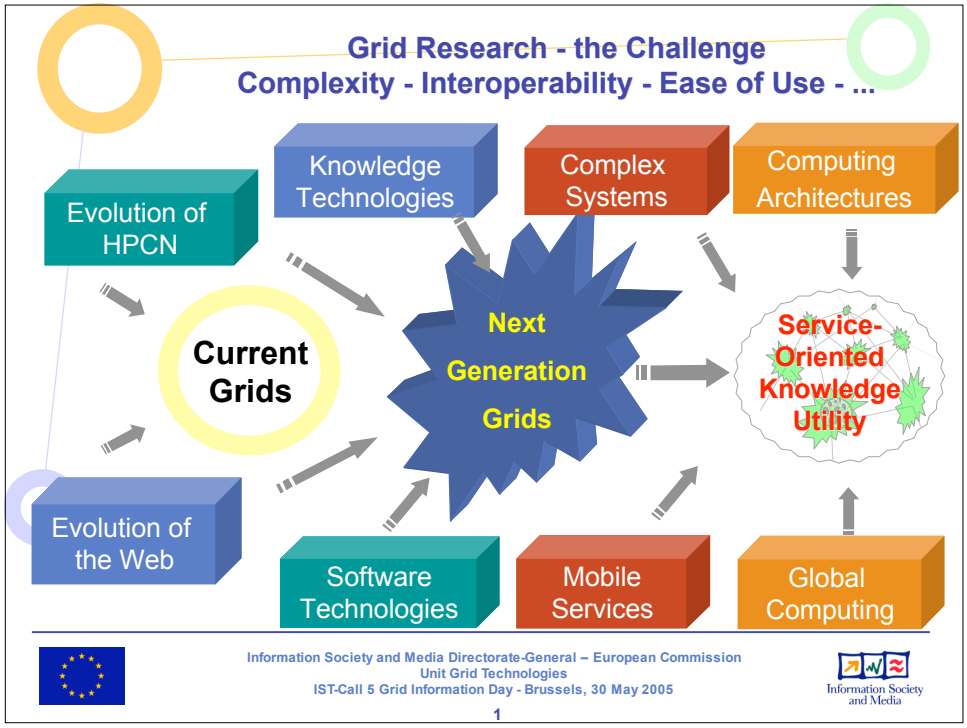
Att inte besluta får lika stora konsekvenser som aktiva beslut – men de är oftast värre!

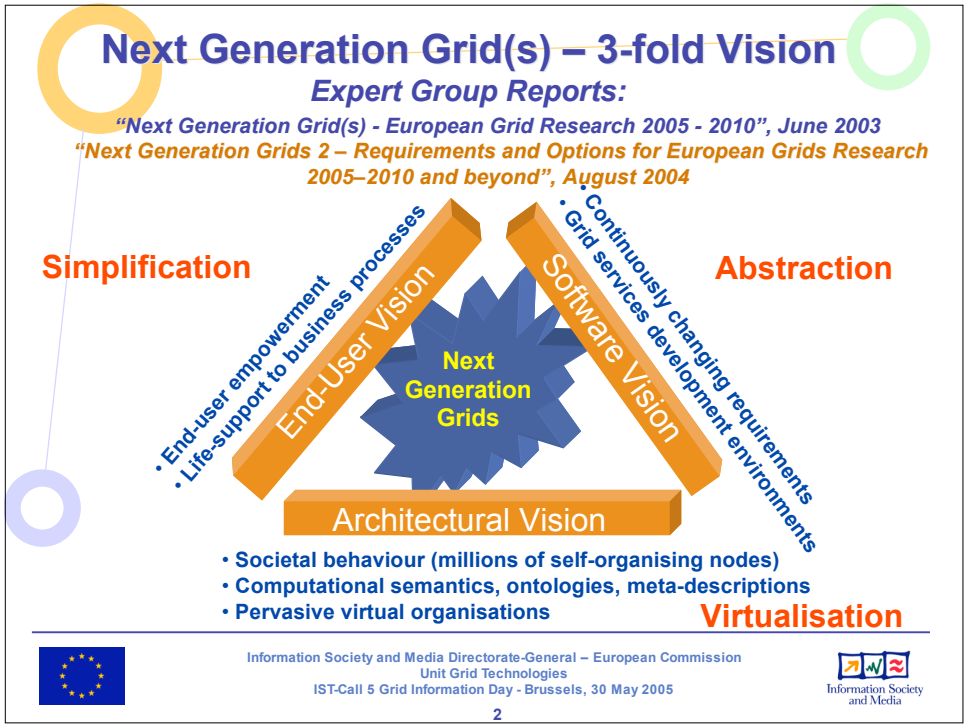
Det behövs åtgärder *nu* för att Sverige ska vara framgångsrikt om 15–20 år!

Teknisk Framsyn

www.tekniskframsyn.nu

Teknisk Framsyn

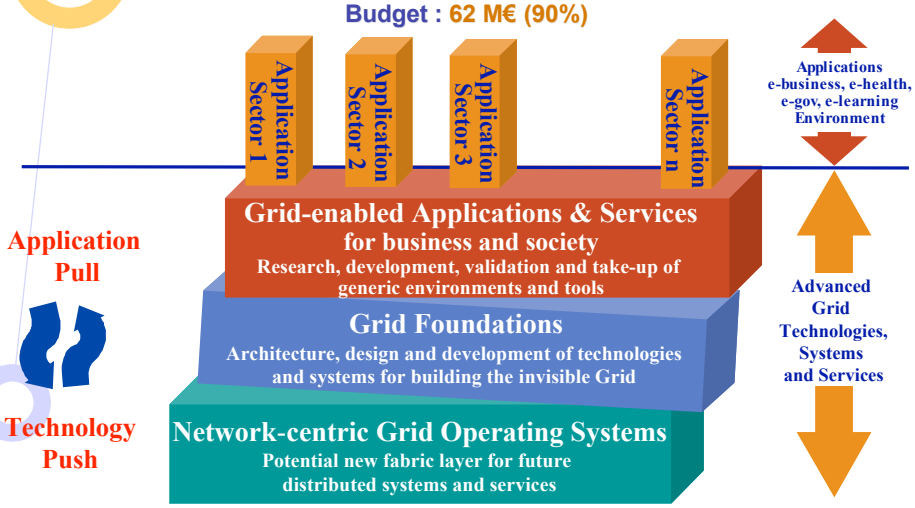

Grid Research - the Challenge
Complexity - Interoperability - Ease of Use - ...
Evolution of HPCN
Knowledge Technologies
Complex Systems
Computing Architectures
Current Grids
Next Generation Grids
Service-Oriented Knowledge Utility
Evolution of the Web
Software Technologies
Mobile Services
Global Computing
Information Society and Media Directorate-General – European Commission
Unit Grid Technologies
IST-Call 5 Grid Information Day - Brussels, 30 May 2005
1
Information Society and Media


Next Generation Grid(s) – 3-fold Vision
Expert Group Reports:
"Next Generation Grid(s) - European Grid Research 2005 - 2010", June 2003
"Next Generation Grids 2 – Requirements and Options for European Grids Research 2005–2010 and beyond", August 2004
Simplification
Abstraction
• End-user empowerment
• Life-support to business processes
End-User Vision
Software Vision
• Continuously changing requirements
• Grid services development environments
Next Generation Grids
Architectural Vision
• Societal behaviour (millions of self-organising nodes)
• Computational semantics, ontologies, meta-descriptions
• Pervasive virtual organisations
Virtualisation
Information Society and Media Directorate-General – European Commission
Unit Grid Technologies
IST-Call 5 Grid Information Day - Brussels, 30 May 2005
2
Information Society and Media

WP 2005-2006

Advanced Grid Technologies, Systems and Services

Budget : 62 M€ (90%)



Further Info on Grid Research

- Brochure: "*Building Grids for Europe*"
  - ⇨ FP6 Grid Project Fact Sheets, FP5 Grid Project Result Sheets
- Expert Group Reports
  - ⇨ "Next Generation Grid(s) – European Grid Research 2005 - 2010", 2003
  - ⇨ "Next Generation Grids 2 – Requirements and Options for European Grids Research 2005–2010 and beyond", August 2004
- FP5 Gridstart "*IST Grid Projects Inventory and Roadmap*"
- Brochure: "*Achievements of EU Grid Projects*"
- IST Work Programme 2005-2006

www.cordis.lu/ist

and more:
www.cordis.lu/ist/grids

ARTEMIS

The European Technology Platform for Embedded Systems

Introduction to the ARTEMIS Strategic Research Agenda

Eric Schutz
Vice President External Technology Coordination
STMicroelectronics

(ftp://ftp.cordis.lu/pub/ist/docs/artemis/eric_schutz.pdf)

Utdrag för presentation 2005-08-19 av K-E Sjödin

The ARTEMIS Technology Platform

Advanced research and technology in embedded intelligence and systems

Aim and scope

- Develop and drive a joint European vision and strategy on Embedded Systems
  - R&D and educational challenges
  - structural challenges: IPR, open source software, standards, research infrastructure,…
- Align fragmented R&D efforts in the ERA along a common strategic agenda at Community, intergovernmental and national levels

Embedded Systems - what they are

- **Intelligent electronic devices**
  - contain a computing device, transparent to the user
  - combine HW & SW
  - part of larger non-computing system(s)
  - resource constrained
- **Reactive to their environment**
  - “real-world” systems for control, perception and cognition
- **Networked**
  - Ad-hoc collaboration

Connecting the physical to the virtual world

Embedded Systems

- Embedded Systems are everywhere
  - cars, roads, bridges, tunnels, medical instruments, surgical robots, homes, offices, factories, aeroplanes, airports, mobile phones, phone networks, virtual reality glasses, clothes, …
- Interconnected into networks of many devices
  - … but are not general-purpose PC's, servers, etc…
- Used in all market sectors
  - automotive, aerospace, medical, environment, communications, entertainment, textiles, transport, logistics, printing, chemicals, food & drink, timber, materials, agriculture, …

The ARTEMIS Vision

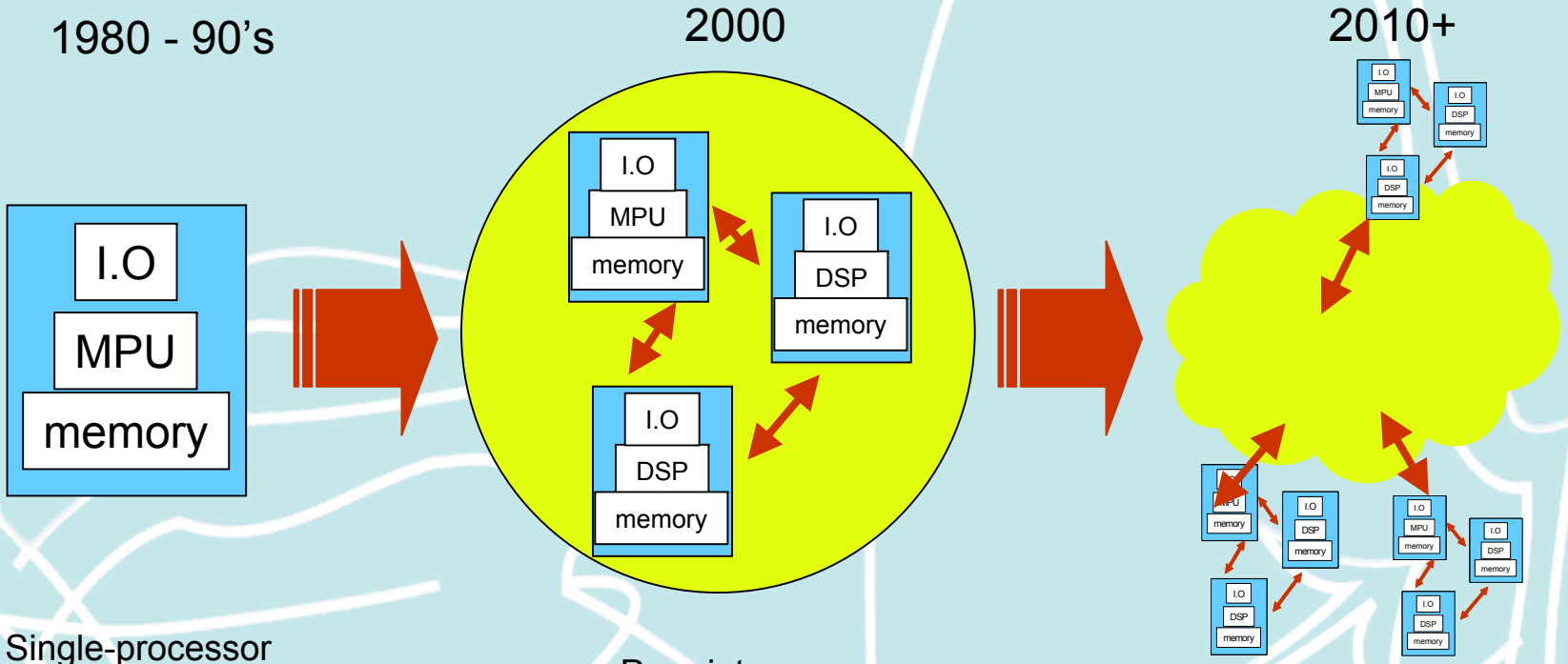
- … An ongoing, major evolution of our society in which all systems, machines and objects will become digital, communicating and self-managed

- Societal and economical consequences:
  - Competitiveness of most industry sectors will rely on ES innovation capability
  - ES technologies critically important in rebalancing Productivity Growth between Europe and the US and Asia
  - Security, Safety and Quality of life in our society will increasingly depend on ES technologies

The ARTEMIS Vision

- **Europe takes a leading role in ES development**
  - Requires committed investment in research and development
- **ARTEMIS will facilitate and stimulate success by:**
  - Avoiding fragmentation, ensure effective use of resources
    - Focused research and development
  - Establishing an environment supportive of innovation
    - Cooperation and competition in technological development
  - Proactively stimulating the emergence of a new supply industry
    - Components, tools, design methodologies

Embedded Systems
New Paradigm, New Challenges



- = Major technological complexity challenge
  - **How to maintain Europe's lead in meeting this challenge?**

The ARTEMIS SRA

- **The ARTEMIS platform elaborates a Strategic Research Agenda (SRA) around the vision to:**
  - Make Ambient Intelligence a reality
    - Address the complexity problem
  - Ensure competitiveness of European industries
  - Create opportunities (knowledge, ecosystem, …) for a New Industry to flourish

- **Subscribed to by Europe's leading technology companies and institutes**

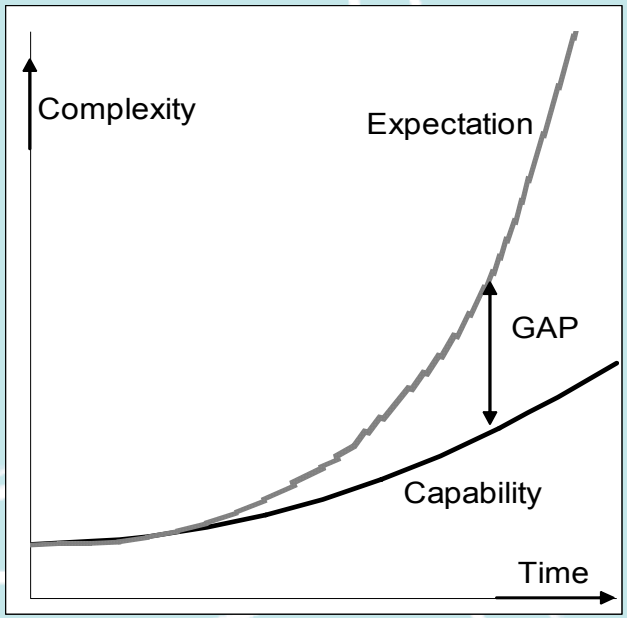
The ARTEMIS SRA Targets

- **To define a focused strategy, ARTEMIS set High Level targets to be attained by 2016**
  - 50% of ES deployed worldwide based on ARTEMIS results
  - Twice as many SMEs within the aegis of ARTEMIS engaged in the ES supply chain
  - Realise an integrated chain of European-sourced tools, to support development of ES
  - Achieve seamless interoperability between the envisaged Ambient Intelligence environments
    - At home, travelling, at work, in public spaces, …
  - Generate at least 5 ‘radical innovations’
    - Comparable to e.g. µ-processor, DSP, software radio, …
    - The number of relevant patents will have doubled

The ARTEMIS SRA Targets

- **To meet these targets, ARTEMIS must aim to close the design productivity gap**

  - Reduce cost of system design by 50%
  - Achieve 50% reduction in development cycles
  - Manage complexity increase of 100% with 20% effort reduction
  - Reduce by 50% the effort for re-validation and re-certification
  - Achieve cross-sectoral reuse
    - E.g. automotive, aerospace and manufacturing



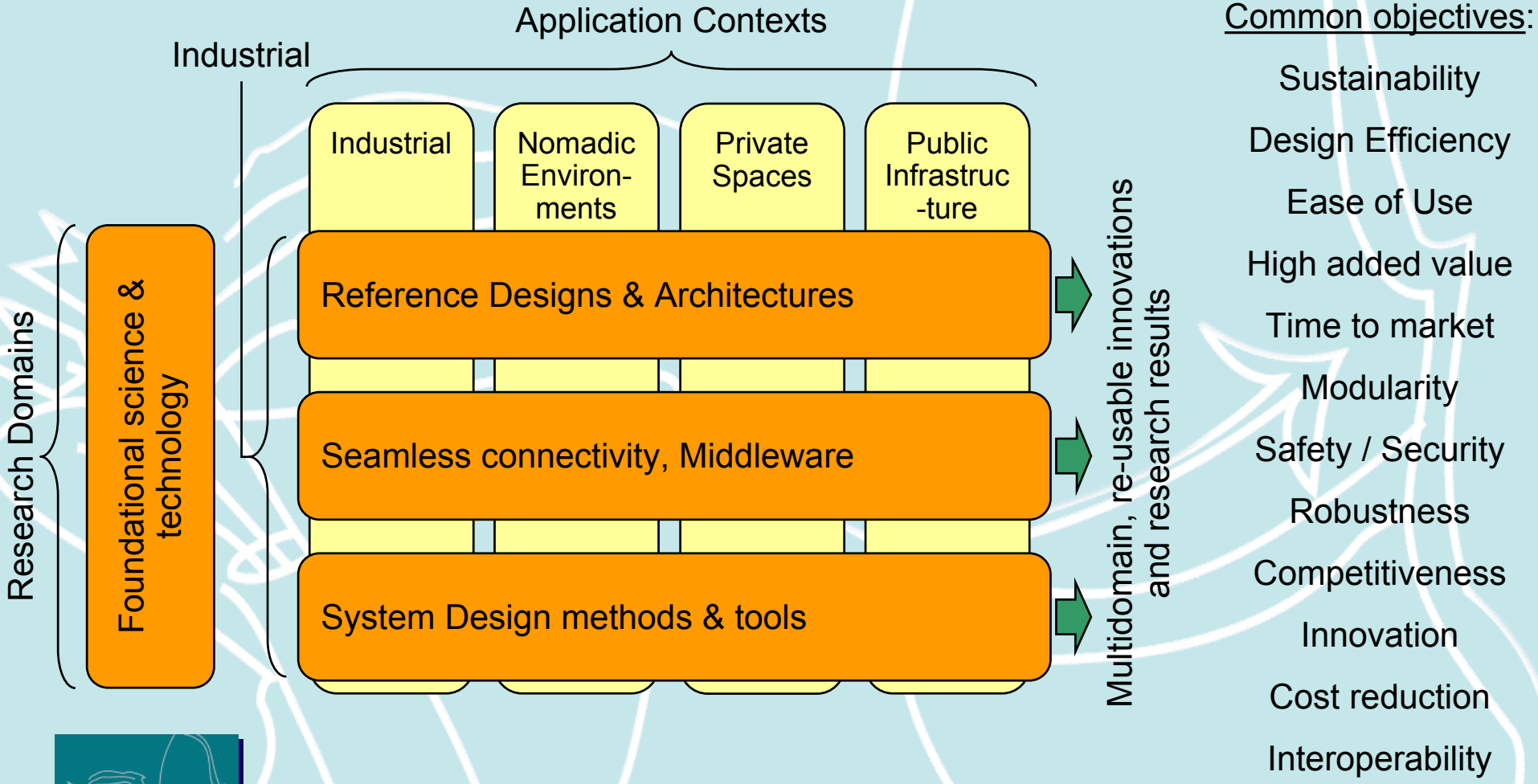
ARTEMIS Application Contexts

Focus research on technologies with high re-usability

- Identified four, strategically significant “Application Contexts”:
  - **Industrial systems**
    - Automotive: “Frugal, safe car”
    - Aerospace: “Customisable, efficient, safe air transport ”
    - Manufacturing & process Industries: “Efficient, flexible manufacturing”
  - **Private spaces:** “Efficiency, safety and pleasure in the home”
    - Includes Medical sector
  - **Nomadic Environments:** “Walk, Talk, Hear, See”
  - **Public Infrastructure:** “Secure and dependable environment”

ARTEMIS' Pan-Application-Context approach

- **ARTEMIS approach cuts barriers between application sectors, stimulating creativity and yielding multi-domain, re-usable results**



Identified RESEARCH DOMAINS

- **Reference designs and architectures:**
  - Create a generic platform and a suite of abstract components
- **Seamless connectivity and middleware**
  - Interoperable link to the physical world
- **Design methods and tools:** The ARTEMIS method
  - Design efficiency, systematic design, productivity and quality
- **Scientific foundational research**
  - provides the essential breakthrough ideas driving future innovation