Challenges and Opportunities in Embedded Software Development

Keynote RTiS 2009
Hans Hansson
Prof. in Real-Time Systems
Director of Mälardalen Real-Time Research Centre
http://www.mrtc.mdh.se/han
hans.hansson@mdh.se

EMBEDDED SOFTWARE – WHY BOTHER?

Competitiveness by Software (2)

Importance for Sweden

Swedish product export 2007 (SCB)

Approximately 50% are software intensive products
EMBEDDED SOFTWARE: TRENDS

Some more insights

- 90% of new innovations is realized by software
- 23% of total cost, estimated to increase to 35% 2010

Complexity?

- Automotive: ~2 MLOC
- Automation: ~20 MLOC
- Telecomm: ~200 MLOC

Telecommunication Evolution

SonyEricsson:
Software doubled every 16 months

Industrial Automation Software Evolution

KLOC = 1000 lines of code

- >1000 KLOC: multiple development sites
- 200 - 300 KLOC: single development site
- ~100 - 300 KLOC: 2-3 development sites
- ~1‘500, 3‘000 KLOC: 3rd Party OS, Graphics, DB, Office Suite, Components

Automotive

- In 5 years, vehicles will contain as much memory as a PC of today
- Complexity is increasing
- Development and maintenance costs are increasing

History repeats itself...

Source: Audi
Historical development of complex systems

- ~1980 – Single company development at one site through research and development centres. E.g. Ericsson (Älvsjö), Asea (Västerås), etc.
- ~1990 – COTS. Buying standard components such as OS, Communication etc.
- ~2000 in time order
  - Outsourcing of simple tasks for cost reduction (testing, single service) and sometimes to be present (testing, simple service) and sometimes to be present in certain countries
  - Maintenance of products is outsourced to offshore development centres in low cost countries.
  - Subsystems outsourced
  - New development is outsourced (e.g. Iphone, ..)
- New skills required in integration, specification, architecture, and collaboration to be successful

Global competence

- How many engineers are educated each year?
  - In Sweden?
  - In the US?
  - In India?
  - In China?
- 2004: China 600,000, India 350,000 and US 70,000
- 2004: True numbers
  - Sweden: 4,000
  - US: 137,437
  - India: 112,000
  - China: 351,537

New development paradigm

Open innovation examples

Can open innovation be used in development of embedded systems?

- Can quality be guaranteed?
  - Can I trust components developed by “anyone”?
- Can confidentiality be guaranteed?
  - Can “anyone” develop my components without knowing my secrets?
How to meet the SW complexity challenge

- Partitioning and integration
- Abstraction
- Reuse
- Automation
- Standardisation

Methods and tools
(+architectures and platforms)

Moores curse for software development
(our ability to handle SW does not increase as fast)

“Moores law”
Doubled computing capacity every 18th month

Gordon Moore

– divide and conquer

Abstraction
Reuse
Automation
Standardisation

Methods and tools
(+architectures and platforms)

Embedded SW challenge

Specific requirements (extra functional properties)
- Systems are resource constrained (space, power, computation, ...)
- Real-time (sometimes very stringent)
- Dependability (sometimes safety-criticality)
- ...

Quality assurance at a reasonable effort is a constant challenge
- Fast may be more important than precise

Availability, Reliability, Maintainability

No uniform set of reqs. ⇒ Specialized domain specific methods are needed

Encapsulation

Software Component (given correct in isolation)

– Composability

\[ T = T_1 + T_2 \ ? \]
\[ M = M_1 + M_2 \ ? \]

Answer: NO (typically)

- Tradeoff between the simplicity and accuracy.
- Composability at the cost of reduced efficiency

Fundamental Scientific Challenges

- Composability

- Encapsulation

- Security

- ...
Observations of the practice of software engineering

- About 80% of software development deals with changing (adaptation, improvement) of existing software
- Time to market is an important competitive advantage:
  - Importance of incorporation of new innovations quickly
- System should be built to facilitate changes
  - Easy removal and addition of functionality
- Systems should be built to facilitate reuse
  - Easy integration of existing functions

Hypothesis

- By building embedded software (and systems) from reusable components
  - complexity,
  - integration, and
  - quality assurance
- can be handled in a more cost efficient and scalable way

Remains to be proven!
(at least for embedded software)

Approach: Component-Based Development

- Software built from larger reusable components

Software Component Definition

Szyperski (Component Software beyond OO programming)

- A software component is
  - a unit of composition
  - with contractually specified interfaces
  - and explicit context dependencies only.
- A software component
  - can be deployed independently
  - it is subject to composition by third party.

Component-based vs. Model-based

- **Model-based** is focusing on abstraction
  - The model is the system
  - Automatic translation from model to system
  - "Top down"
- **Component-based** is focusing on reuse
  - "Bottom-up"
  - Two independent processes
    - Development of components
    - Development of systems from components

Complementary approaches
Reusable components

"Model based Development"

Optimised implementations

- Components as design entities
  - Traditionally, SW components are binaries

**“Basic values” (Philosophy/approach)**

- Development of application SW for the vehicular, automation, and telecom domains
- First class citizens:
  - **Extra functional properties**
    - Timing, Resource usage, Reliability, and Life-cycle properties
  - **Reuse**
    - At multiple levels of abstraction/granularity
      - Small/Large SW components
    - Specifications (models) and analyses
  - **SW Design and Deployment**
    - Efficient development requires an interplay between these throughout the SW development (hypothesis)

**Final remarks**

- Embedded systems development is not static
  - System complexity is sky-rocketing
  - Global competence is growing (and the centre of mass is moving)
- Opportunity for new players
- Threat for the established
  - IBM, Microsoft, Google, …

**The applied (externally motivated) researcher’s opportunity and threat**

- Problems are constantly evolving (moving target)
- Solutions are becoming obsolete
  - (and possibly: obsolete solutions can be recycled?)
- Assumptions must constantly be questioned
- Research must target future industrial reality (2020?)
- … better stick to basic research ;-)
Those that master Software Intensive Embedded Systems are winners today… and tomorrow…

[Our mission is to help Swedish Industry stay on top!]

PROGRESS
A national Swedish Strategic Research Centre