Embedded Systems – Challenges and Work Directions

EmSys Summer School
Salzburg
July 2, 2003

Joseph Sifakis
Verimag Laboratory
Outline

Embedded Systems - Scope, Stakes

System-centric Approach - Grand Challenges

Work Directions
Embedded Systems: Scope

An Embedded System integrates software and hardware jointly and specifically designed to provide given functionalities.
Embedded Systems: Main Characteristics

- **Criticality**
  - The degree of criticality depends on the consequences of deviation from nominal behavior on safety, security, mission, business …

- **Reactivity**
  - Embedded systems interact with some physical environment
  - They must meet real-time constraints relating their execution speed to the speed of their environment

- **Autonomy**
  - Embedded systems need to fulfill their functions without human intervention for extended periods of time
Embedded Systems: Needs

We need system development methodologies allowing to consider jointly

- Functionality (capacity to deliver a specific service)
- Quality
  - Performance properties (response time, throughput, jitter, latency)
  - Dependability properties (safety, security, availability, reparability, …)
- Physical implementation constraints
  - Use of resources (memory, energy, power ..)
  - Deployment context (weight, size, resistance to vibration …)
- Market constraints covering aspects related to the product quality and
time to market for given functionality and target consumer profile

**Building systems of guaranteed functionality and quality, at acceptable costs is a major technological and scientific challenge.**
Embedded Systems : Current Limitations

Two main technological approaches and related know-how :

- Hard real-time and safety critical systems
  - Automated aircraft landings
    High reactivity + High Dependability
  - Existing technologies are not mature enough for automobile

- Distributed soft real-time systems
  - 2nd generation mobile phones
    Distribution + Good reactivity + Good dependability
  - Existing technologies are not mature enough for UMTS

The vision :

Reactive, Distributed, Heterogeneous and Complex Systems

- Automated freeways
- Next generation Air Traffic Control
- « Ambient Intelligence »

Reactivity + Distribution + High Dependability + Autonomy + Security + Safety + …. + Intelligence
Air Traffic Control – the Next Generation
Is it … attainable?

Start of the project

1984

Air traffic takes another turn
FAA weighs fixes for automation project
By Gary H. Anthes - April 25th, 1994

The Federal Aviation Administration is considering major changes in its troubled Advanced Automation System (AAS) project, now billions of dollars over budget and years behind schedule.

FAA administrator David R. Hinson told a congressional panel that the agency might scrap the project entirely, although he said some scaling back was more likely. The FAA has spent about $1.5 billion to date on the estimated $6.9 billion air traffic control system.

The FAA's Course Correction
The Ugly History of Tool Development at the FAA
By David Carr and Edward Cone April 9th, 2002

Online exclusive: The agency wrote off $1.5 billion of its $2.6 billion investment to overhaul the nation's air traffic control computer systems. What went wrong? (Just about everything.)

One participant says, "It may have been the greatest failure in the history of organized work."

Certainly the Federal Aviation Administration's Advanced Automation System (AAS) project dwarfs even the largest corporate information technology fiascos in terms of dollars wasted. Kmart's $130 million write-off last year on its supply chain systems is chump change compared to the AAS. The FAA ultimately declared that $1.5 billion worth of hardware and software out of the $2.6 billion spent was useless.
Outline

Embedded Systems - Scope, Stakes

System-centric Approach - Grand Challenges

Work Directions
The main focus is the end result: a system as the combination of HW and SW in interaction with its physical environment

- **Joint Design** (Hardware, Software, Environment)

  - Determine tradeoffs between cost and quality by taking into account features of HW, SW, environment
System-centric Approach

Today:

Systems designs are validated experimentally:

- No unified theory for software and its dynamic properties on a given execution platform.
- Complex systems are built through a succession of incremental developments.
- Exploding validation costs!
Grand Challenge 1: Coping with Heterogeneity

Build complex systems by integrating heterogeneous components
Grand Challenge 1: Coping with Heterogeneity

Two sources of heterogeneity:

- **Interaction can be**
  - blocking (e.g. Rendezvous in CSP)
  - non blocking (e.g. sending a message)
  - atomic (e.g. synchronous broadcast)
  - non atomic (e.g. function call)

- **Execution can be**:
  - synchronous (e.g. VHDL, Esterel)
  - asynchronous (e.g. SDL)

We need a unifying theoretical framework for composing heterogeneous components - Standards are only the tip of the iceberg
Grand Challenge 1: Coping with Heterogeneity

A: Atomic interaction

B: Blocking interaction

Asynchronous Computation

Synchronous Computation

Lotos

CSP

Java

UML

SDL

UML

Esterel

VHDL

Statecharts 1

Statecharts 2

nonA B

A nonB

donA nonB
Grand Challenge 2: Correctness by construction

The validation effort grows exponentially with the number of components integrated.

Replace *a posteriori* validation methods with incremental validation.

- **Composability**: Preserve functionality and component quality across the integration process

- **Compositionality**: Infer the system’s properties from its components’ properties

We need tools and techniques for building systems that are *correct by construction*
Grand Challenge 3: Intelligence

The means for improving quality (dependability and performance) of systems:

- **Reflexivity**: capacity to analyze, auto-diagnose its own state
- **Adaptability**: capacity to adapt (auto-organize+plan) behavior according to dependability and performance objectives

Develop theoretical tools and practices for engineering intelligent systems
Outline

Embedded Systems - Scope, Stakes

System-centric Approach - Grand Challenges

Work Directions
Model-based Development

Trends

- Move from physical prototypes to virtual prototypes (models)
- Advantages: minimize costs, flexibility, genericity, possibility of verification

Directions

- Modeling and validation environments for complex real-time systems
  
  ➢ Libraries of Components
    eg. HW, SW, Models of continuous dynamic systems
  
  ➢ Languages and tools for assembling components
  
- Synthesizing embedded software from domain-specific models
  
  e.g. from Matlab/Simulink
Programming Technologies

Directions:

- Expressive programming languages with semantically aware programming/analysis tools – detect as many errors as possible by static analysis, abstract interpretation and model checking

- Extend programming languages to model system features – Bridging the gap between programming and modeling

- Compilation used to synthesize code that meets given extra-functional requirements
Programming Technologies

Compiler

Application SW | Architecture model | Timing QoS | Security

Scheduler

Event Handler | Task1 | Task2 | Task3 | Task4

Synchronization and resource management

Platform
Operating Systems

Operating systems are often, more complex than necessary, undependable, with hidden functionalities, difficult to manage and use efficiently

Directions

• Modularity to use minimal OS configurations - incremental composition of features

• Adaptability to optimize the use of available resources and adapt to dynamically changing requirements
  - Give up control to the application – move resource management outside the kernel
  - Supply and allow adaptive scheduling policies

• Application specific operating systems ex. OSEK, ARINC, JavaCard, TinyOS with features ensuring safety and security
Control for Embedded Systems

Automated control systems are central to embedded technologies

- They are used in typical control applications such as flight control, unmanned vehicles, process control for manufacturing
- But also, for network traffic control, adaptive scheduling, for applications where adaptability is sought

Directions

- Hybrid systems - combine continuous and discrete dynamics
  - Need for theory integrating multi-disciplinary aspects from control theory, numerical analysis, and computer science
  - Modeling and verification techniques

- Implementation techniques
  - Code generation techniques e.g. from Matlab/simulating descriptions
  - Distributed implementation, taking into account the influence of: communication delays, jitter, clock drift, aperiodic sampling …
Dependability

- Embedded technologies are poorly suited to traditional techniques based on redundancy
- Dependability should be a guiding concern from the very start of development. This applies to programming style, architectures, validation techniques, fault-tolerance mechanisms, traceability …

Directions

- Methodologies for domain-specific standards, such as:
  - DO178B Process Control Software Safety Certification (real-time avionics software)
  - Common Criteria for Information Technology Security Evaluation

- Verification Technology (resistance to certain classes of errors and attacks) – certification technology

- New approaches for fault tolerance: active redundancy, control techniques, verification techniques
Challenges, Stakes and Opportunities

Embedded Technologies are of strategic importance

Industry

• Europe has leading positions in sectors where embedded technologies are central to growth (avionics, automotive, space, consumer electronics, telecom devices, energy distribution, railway transport, …)

• Potential impact on IT industry

Research and Education

• Embedded systems is a new multi-disciplinary research area requiring combination of competencies in software, control, networks, electronics, man-machine interfaces

• An opportunity for CS