Strategic Directions in Real-Time and Embedded Systems
John A. Stankovic ET AL.
**Real-Time System:** "The correctness of the system depends not only on the logical results, but also on the time at which the results are produced."

**Embedded Systems:** "Components of a larger system."

- Definition better suited for a Subsystem.
- Does not address standalone (non-hosted systems), i.e., medical monitors, satellites, personal computing devices (cell phones, PDAs, portable MP3 players). These qualify as real-time and/or embedded systems; yet are self contained and can perform their intended task independently - they are full systems in their own right.
- The paper pays no attention to these standalone embedded systems, for the most part. Nevertheless, the majority of the discussions still apply.
- The paper seems to be geared more towards complex, compound, industrial systems, which better fit their Embedded System definition.
- The figures regarding the size and importance of the real-time embedded market are somewhat old by now. The paper was published in late 1996.
Future Challenges and Research

1 System Evolution

Problems:

- Process improvement  Flexibility, versatility and adaptability of the system’s components (both hard and soft) to enhancements in its tasks.

  - Problems described were not due to real-time or embedded systems technologies. They were caused mainly by human factors: lack of documentation, lack of upgrade paths and procedures, lack of proper timing analysis, etc.

- Equipment upgrades  Smooth transitions. Newer equipment replacing old.

  - Not always possible, i.e., equipment in Mars, hostile industrial or natural environments.
  
  - Required time and/or resources may be prohibitive, even if the technology allows for a smooth upgrade path, i.e., Hubble telescope.

Approaches & Challenges:

- Extensive use of open-standard-based components (both soft & hard)

- Coherent set of interfaces for control, scheduling and management among individual (sub)systems and plant-wide.

- Development of schemes with demonstrable predictability.

- Development of schemes for (sub)system composition.
2 Open Real-Time Systems

Challenges:

- General-purpose, open real-time systems and applications that permit a dynamic mix of multiple, independently developed real-time applications to coexist.
  - Standardized APIs
  - OS-independent
  - Architecture-independent

- "Perfect a priori schedulability analysis is effectively impossible." But then again, "perfect execution may not be the most important criterion for open, consumer real-time systems."
  - Authors suggest adequate performance as the most cost-effective execution as seen by the user.
  - This may not always be the case, i.e., when a (sub)system interacts with others there may be unforeseen inherited performance requirements.
3 Composibility

- Modules forming a system, across three interacting domains:
  1. Function
  2. Time
  3. Fault tolerance

- **Goal:** Adaptive high-performance fault-tolerant embedded systems that dynamically address real-time constraints and provide:
  - *a priori* acceptable system-level performance guarantees, and
  - graceful degradation in the presence of failures and time constraints
4 Software Engineering

Problem:

- *Since retrofitted tools are never quite satisfactory, real-time system engineers resort to developing their own tools specialized for the current project.*

Challenges:

- *Software engineering will need to undergo a radical shift in perspective and approach:*
  
  - *Time, dependability and QoS constraints must become first-class concerns.*
  - *Evolvability must be ensured.*
  - *Software must be structured into composable modules.*
  - *Software must be adaptive and configurable.*
  - *Timing constraints must be dynamically derived and imposed on the basis of end-to-end requirements.*
5 The Science of Performance Guarantees

- What is required is a science of performance guarantees that can provide a more formal analysis of dynamic real-time systems when:
  - the environment is not fully predictable or controllable, and
  - failures occur

- A science of performance guarantees needs to be developed that permits accurate analysis of meeting probabilistic requirements of various timing and delay specifications.

Problems:

- Existing schedulability conditions and validation algorithms produce unacceptably low degrees of success for applications with sporadic processing requirements.

- Many existing validation algorithms are based on deterministic workload and resource models and work for deterministic timing constraints.

- Most existing validation algorithms are for statically configured systems.
6 Reliability and Formal Verification

"Instead of performing a mapping from high-level abstraction of a system to its
detailed implementation, what is needed is a multilevel specification mechanism to
establish the preservation of real-time requirements in terms of conditions on the
low-level implementation. Such technology should ensure the correct functioning
of the system through monitoring and checking at runtime."

- Expensive proposition:
  - Can lead to serious tradeoffs between overhead and useful work done
    by the system. Can deadlines still be met?
  - Smaller embedded systems often do not have resources nor time to
    spare.
  - More horsepower for smaller systems may be out of the question:
    cost, weight, power requirements, additional components, over-complexity
    of a simpler system.
  - Legacy systems Sometimes there is no way to upgrade a platform
    in order to accommodate these new system requirements.

7 General System Issues

A good collection of questions Food for Thought and research areas.
8 Real-Time Multimedia

Challenges:

- *Precise specification of the predictability requirements.* Separating end-to-end from internal (sub)system-level predictability requirements.

- *Schemes for mapping predictability requirements into mechanisms that can demonstrably meet the requirements.*

- *Development of an integrated set of analysis tools that combines application QoS satisfaction with the ability to handle a wide range of stochastic application and system behavior.*

9 Programming Languages

Problem:

- *In general, they lack the expressive power to deal with temporal requirements and implementation strategies.*

Approach:

- *Developments in the OOP paradigm have drawn a distinction between functional and non-functional behaviors.* The functional behavior is expressed within a computational model, but the computational model itself can be altered by programming at the metalevel. *This process, known as reflection, will enable behavioral aspects to be addressed.*
10  Education

- Teaching real-time systems is teaching the science of performance guarantees.
- Two distinct aspects on teaching it:
  1. Understanding the execution of a (sequential) program in time
     - Interactions with:
       * hardware
       * OS
       * other components
       * other systems
  2. Management of logical and physical concurrency in time.
     - study of cooperating sequential processes (traditional)
     - study of the effects of competition from shared resources among processes (more specialized)
Ten-Year Vision

- Real-time to superset a vast number of the current running systems of the time,
  
  - No mention of standalone, roaming, personal or pervasive real-time computing.

- Open standards and systems presented as widespread, key infrastructure elements of the real-time embedded future,
  
  - If they are so fundamental, where are they? Why aren’t they more popular?
Summary

Real-time research has yet to grapple with three major realities concerning real-time applications:

1. Real-world real-time systems are expected to survive and continue to operate even when not all timing constraints are met or when components fail.
   - An early goal which is still relevant

2. Due to economic and portability considerations, the tendency towards the use of off-the-shelf hardware and software components to build real-time systems is increasing
   - They are not necessarily standards-compliant, open, nor de facto industry-standard, but it's a start.

3. The prohibitive cost of modernizing an industrial real-time computing system often results from the down time and risks associated with inserting time-dependent new technologies into a functioning industrial system,
   - These are not technological problems. They stem from bad planning and execution by humans. There are ways to smooth the roll out of updated systems in concert with existing ones.
   - The underlying assumption is that all real-time systems are either hard or impossible to upgrade and something needs to be done about it.
   - No mention of hard-to-access or expensive-to-upgrade currently running systems. Incomplete view.
   - No consideration for initial equipment cost, equipment maintenance, replacement cost nor amortization over time.