Abstract

Software is a dominating factor in the development of complex systems. It plays a crucial role in the performance of the final product at the one hand, while it contributes significant to the development cost and elapsed time of development. This paper will discuss the role of software in the broader system context. An improved understanding of the role of software enables the system architect, and the other stakeholders of the product creation process, to integrate the software development better. In this way hardware-software tradeoffs can be made, balancing performance, costs and risks.
Relative Contribution of SW

- Electronics
- Mechanics
- Physics/Chemistry, etc.

1970 to 2000:
- Electronics: Decreasing effort
- Mechanics: Increasing effort
- Physics/Chemistry, etc.: Constant effort
Mismatch between Role and Discipline

**role of software**

integration technology
captures *application* functionality
defines lot of *system* behavior
determines how much of potential *system* performance is achieved
acts as director

**focus of software discipline**

software technologies, such as:
- programming languages
- data bases
- operating systems
- component technologies
- engineering practices

mismatch!
Control Hierarchy along Technology axis

- Human user
- Application SW
- Control SW
- Digital electronics
- Analog or power electronics
- Mechanical device
- Optical device
- Sensor

Legend:
- Local automation or safety
The Role of Software in Systems

Gerrit Muller

version: 1.3
August 31, 2020
SWdisciplineCharacterization
Quality Attributes annotated with SW relation

- **Usable**
  - usability
  - attractiveness
  - responsiveness
  - image quality
  - wearability
  - storability
  - transportability

- **Dependable**
  - safety
  - security
  - reliability
  - robustness
  - integrity
  - availability

- **Liable**
  - liability
  - testability
  - traceability
  - standards compliance

- **Serviceable**
  - serviceability
  - configurability
  - installability

- **Ecological**
  - ecological footprint
  - contamination
  - noise
  - disposability

- **Effective**
  - throughput or productivity

- **Consistent**
  - reproducibility
  - predictability

- **Efficient**
  - resource utilization
  - cost of ownership

- **Future-proof**
  - evolvability
  - portability
  - upgradability
  - extendibility
  - maintainability

- **Logistics Friendly**
  - manufacturability
  - logistics flexibility
  - lead-time

- **Interoperable**
  - connectivity
  - 3rd party extendible

- **Legend**
  - weak SW relation
  - strong SW relation
Design Aspects related to SW

- **Customer Objectives**
- **Application**
- **Functional**
- **Conceptual**
- **Realization**

- Design philosophy per quality attribute: performance, safety, security, ...
- Granularity, scoping, containment, cohesion, coupling
- Interfaces, allocation, budgets
- Information model (entities, relations, operations)
- Identification, naming: HAL_message_acknowledge_status versus ACK
- Static characteristics, dynamic behavior
- System-level infrastructure
- Software development process, environment, repository, and tools
- Life cycle, configuration management, upgrades, obsolescence
- Feedback tools, for instance monitoring, statistics, and analysis
- Persistence
- Licensing, SW-keys
- Setup sequence, initialization, start-up, shutdown
- Technology choices
- Make, outsource, buy, or interoperability decisions
SW Mechanisms

- **Customer objectives**
- **Application**
- **Functional**
- **Conceptual**
- **Realization**

error handling, exception handling, logging
processes, tasks, threads
configuration management; packages, components, files, objects, modules, interfaces
automated testing: special methods, harness, suites
signaling, messaging, callback scheduling, notification, active data, watchdogs, timeouts
locking, semaphores, transactions, checkpoints, deadlock detection, rollback
identification, naming, data model, registry, configuration database, inheritance, scoping
resource management, allocation, fragmentation prevention, garbage collection
persistence, caching, versioning, prefetching, lazy evaluation
licensing, SW-keys
bootstrap, discovery, negotiation, introspection
call graphs, message tracing, object tracing, etc.
distribution, allocation, transparency; component, client/server, multitier model