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.
The Role of Software in Systems

Gerrit Muller

version: 1.3
June 21, 2020
RSWRelativeEffort

Relative Contribution of SW

100% time relative effort

physics/chemistry, etc.

mechanics

electronics

1970 2000

100%

relative effort

time
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
Characterization of disciplines

- Concrete
- Tangible
- Mature
- Production lead-time
- Material cost
- Abstract
- Intangible
- Immature
- Flexible?
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

**efficient**
- resource utilization
- cost of ownership

**effective**
- throughput or productivity

**consistent**
- reproducibility
- predictability

**interoperable**
- connectivity
- 3rd party extendible

**serviceable**
- serviceability
- configurability
- installability

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

**down-to-earth attributes**
- cost price
- power consumption
- consumption rate
- (water, air, chemicals, etc.)
- size, weight
- accuracy

**logistics friendly**
- manufacturability
- logistics flexibility
- lead-time

**legends**
- weak SW relation
- strong SW relation

The Role of Software in Systems
6 Gerrit Muller

version: 1.3
June 21, 2020
RSWqualities
Design Aspects related to SW

Customer objectives
Application
Functional
Conceptual
Realization

design philosophy per quality attribute
performance, safety, security, ...
granularity, scoping, containment, cohesion, coupling
e.g., distributed or centralized control
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

The Role of Software in Systems
7 Gerrit Muller
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