The Role of Software in Systems

by Gerrit Muller  University of South-Eastern Norway-NISE

e-mail: gaudisite@gmail.com

www.gaudisite.nl

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
September 9, 2018
RSWrelativeEffort
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

mismatch!

focus of software discipline
software technologies, such as:
  programming languages
  data bases
  operating systems
  component technologies
  engineering practices
Control Hierarchy along Technology axis

- Control
- Feedback

- 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

5 Gerrit Muller

version: 1.3
September 9, 2018
SWdisciplineCharacterization

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

dependable
- safety
- security
- reliability
- robustness
- integrity
- availability

efficient
- resource utilization
- cost of ownership

effective
- throughput or productivity

consistent
- reproducibility
- predictability

interoperable
- connectivity
- 3rd party extendible

liable
- liability
- testability
- traceability
- standards compliance

serviceable
- serviceability
- configurability
- installability

future proof
- evolvability
- portability
- upgradability
- extendibility
- maintainability

logistics friendly
- manufacturability
- logistics flexibility
- lead-time

logistics friendly
- manufacturability
- logistics flexibility
- lead-time

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

ecological
- ecological footprint
- contamination
- noise
- disposability

ecological
- ecological footprint
- contamination
- noise
- disposability

weak SW relation

weak SW relation

strong SW relation

strong SW relation

legend

legend

The Role of Software in Systems
6 Gerrit Muller

version: 1.3
September 9, 2018
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
- 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**

Gerrit Muller
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