Self-Awareness and Self-Consciousness via Software: An Engineering Perspective

Expert Talk, First CHIST-ERA conference
EUR - Rome, 28 May 2010

Holger Giese
System Analysis & Modeling Group, Hasso Plattner Institute for Software Systems Engineering at the University of Potsdam, Germany
holger.giese@hpi.uni-potsdam.de
My Terminology

Self-awareness and self-consciousness via software will lead to a new generation of adaptive and evolving systems of systems.

**BUT we do know yet**

- How to cost effectively engineer this capabilities?
- How to engineer meaningful and trustworthy systems of systems where the subsystem have this capability?

**Disclaimer:** *Self* and *consciousness* are considered only in a weak partial sense.
Adaptation Loop & Self-Consciousness

Self-Awareness:
- Includes collect
- Enables analyze
- Basis for act

Self-Consciousness:
- Includes collect
- Enables analyze
- Basis for decide (alternatives as well as comparison)
- Basis for act

Roadmap:
- Requirements
- Modeling
- Feedback loops
- Assurance
Engineer Monolithic Self-Awareness

Development Time:

Adaption || Software || Context \( \Rightarrow \) Requirements

Runtime:

Benefit:
Adaption based on self-awareness (and context-awareness) enables to fulfill the requirements despite dynamic changes of the context (reflected in the software) and software.

Challenge: engineer adaptation loop
Assumption: known context and software where only parameters change

Legend:
Model
Runtime
System
Software-system

Self-Awareness and Self-Consciousness via Software: An Engineering Perspective
$H(z) = \text{Closed Loop Transfer Function}$

Integral Control Law

$u(t) = u(t-1) + Ke(t)$

Transfer Function

$N(z) = \frac{b_1}{z - a_1}$

Choose a good $K$

Poles of $H(z)$

Beyond Parameter Changes …

**Example:** “Ambient Intelligence (AmI): …

By adding intelligent user interfaces and integrating sensing devices, it should be possible to **identify and model** user activities, preferences and behaviours, and create individualised profiles.”

Gasson, Mark; Warwick, Kevin (2007), "D12.2: Study on Emerging AmI Technologies", FIDIS Deliverables 12 (2)
Engineer Monolithic Self-Consciousness

**Development Time:**

- Software
- Requirement
- Context
- Adaption

- Software
- Context

**Runtime:**

Adaption:

- Requirements
- Software
- Context

Software = f(Context, Software, Requirements) with

**Benefits:**

- Covers more possible changes of the software and context
- Possible changes of the software and context can be analyzed at runtime (by f)
- Changes of the software could also be triggered by changes of the requirements

**Challenges:** engineer + find reasonable f + assurance

**Assumption:** stable context and requirements
Framework: Reuse Adaptation Infrastructure

Decides on the best adaptation

Carries out that adaptation

Bridges abstraction (aggregate info up, map action down)

Changes state in target system

Architecture Layer

Architecture Evaluator

Detects problem in system

Manages arch & env’t model

Relates system info to model

Extracts system information

Model Manager

Gauges

Translation Infrastructure

System API

Resource Discovery

Probes

System Layer

Target System

System API

Effectors

Strategy Executor

Adaptation Manager

Requirements and Models at Runtime

- Requirements are monitored at runtime
- Software is represented by a runtime model (synchronization)
- Generate system specific parts for collect, analyze, decide and act

Prognoses:

“In the near future, software-intensive systems will exhibit adaptive and anticipatory behavior; they will process knowledge and not only data, and change their structure dynamically. Software-intensive systems will act as global computers in highly dynamic environments and will be based on and integrated with service-oriented and pervasive computing.”


“The sheer scale of ultra large scale systems will change everything. ULS systems will necessarily be decentralized in a variety of ways, developed and used by a wide variety of stakeholders with conflicting needs, evolving continuously, and constructed from heterogeneous parts.

Adaptation is needed to compensate for changes in the mission requirements (...) and operating environments (..)

Example: Shuttle System

A shuttle system that builds convoys to optimize the energy consumption

http://www.railcab.de/
Example for Distributed Self-Consciousness/Adaptation

- Distributed learning of a model of the track (context)
- Local learning of a model of the shuttle (self!)
- Planning an adaptation in form of an optimal trajectory
- Trajectory synthesis establishes required assurance

**Engineer Distributed Self-Consciousness**

**Open Problems:**

1. Multiple possibly conflicting adaptation loops
2. Decentralized organization required => self-organization
Proposed Priorities for the Call

- **Cost-Effective Engineering of Systems with Self-Awareness & Self-Consciousness**
  - Forward Engineering via frameworks and models at runtime
  - Reengineering with black-box and white-box models
  - Co-existence of different levels (online application management ➔ self-awareness ➔ self-consciousness)
- **Assurance**
- **Engineering of Systems of Systems with Self-Awareness & Self-Consciousness**
  - Decentralized self-consciousness & adaptation
  - Overcoming the heterogeneity
  - Decentralized assurance