Use-case driven Self-Management Framework

Reza Haydarlou, Michel Oey
Benno Overeinder, Frances Brazier
{rezahay,michel,bjo,frances}@cs.vu.nl

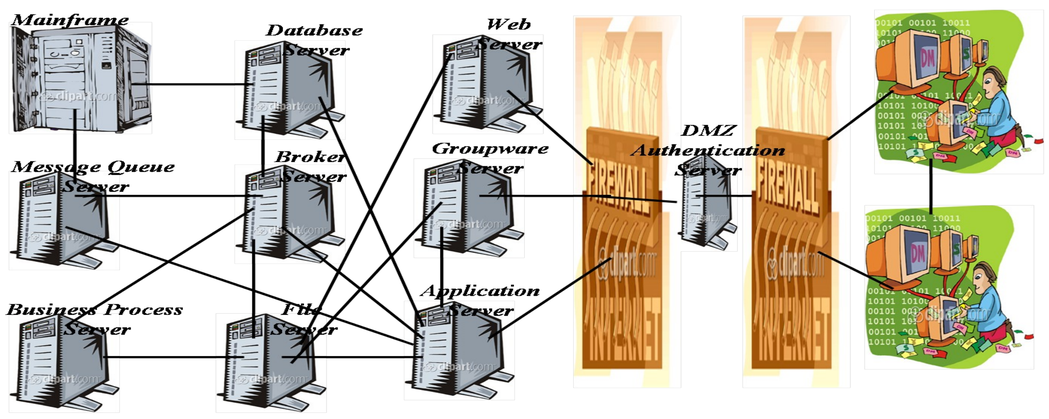
Intelligent Interactive Distributed Systems Group
Department of Computer Science
Vrije Universiteit Amsterdam

1. Overview

Problem

Systems are becoming more and more complex:

- composed of a variety of components
- operating in large-scale distributed heterogeneous environments
- require more human skills to install, configure, and maintain



Approach

Let systems manage themselves:

- define a *self-management* model
- make *use-cases* the unit of management
- recognise a *hierarchy of levels* among use-cases

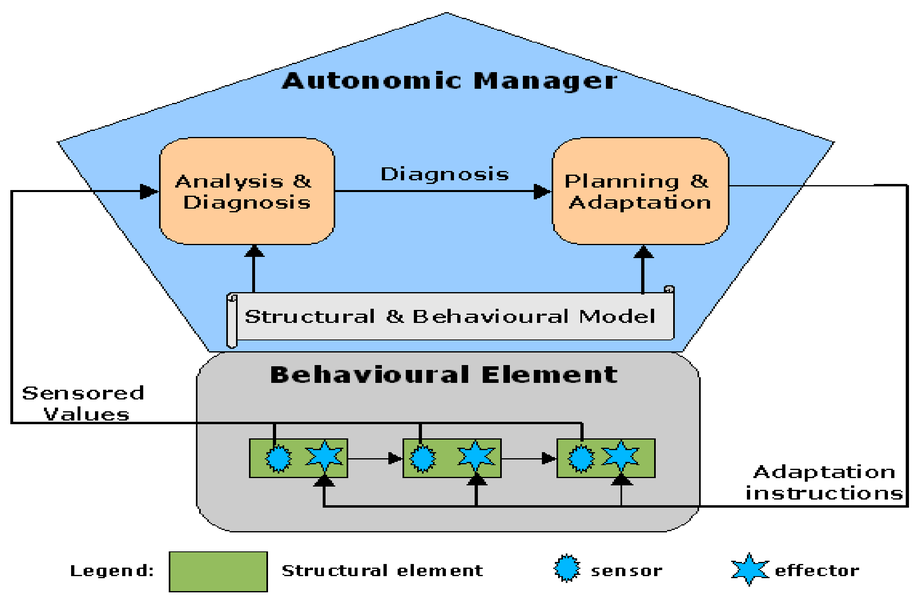
2. Self-Management Model

The model is based on the *feedback loop* created by IBM

1. *Sensors* in the managed unit are triggered
2. Autonomic manager *analyses* sensored values and determines a *diagnosis*
3. Autonomic manager makes a remedy *plan*
4. *Effectors* implement the adaptation instructions

What is the unit of management?

- *Structural* elements – Sub-systems, components, classes, methods
- *Behavioural* elements – Use-cases

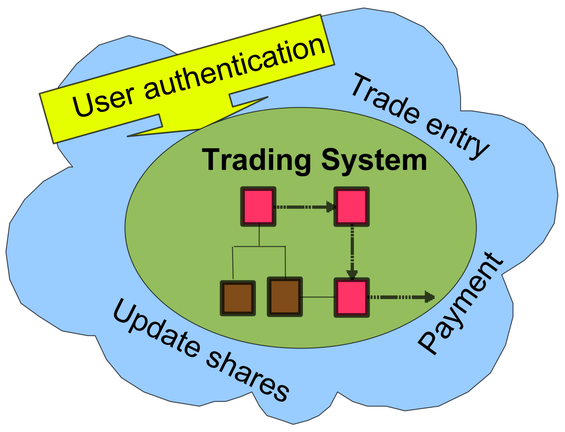


Self-Management feed-back loop

3. Use-Case as Unit of Management

A use-case (*behaviour*) is a description of a process in which a system:

1. receives a request
2. executes the request in one or more *structural elements*
3. produces a response



Choosing the use-case as the unit of management, solves the following problems:

- *Acquiring domain knowledge is known to be difficult.*
  Use-cases are familiar to developers -- *the domain experts* -- who provide the self-management knowledge.

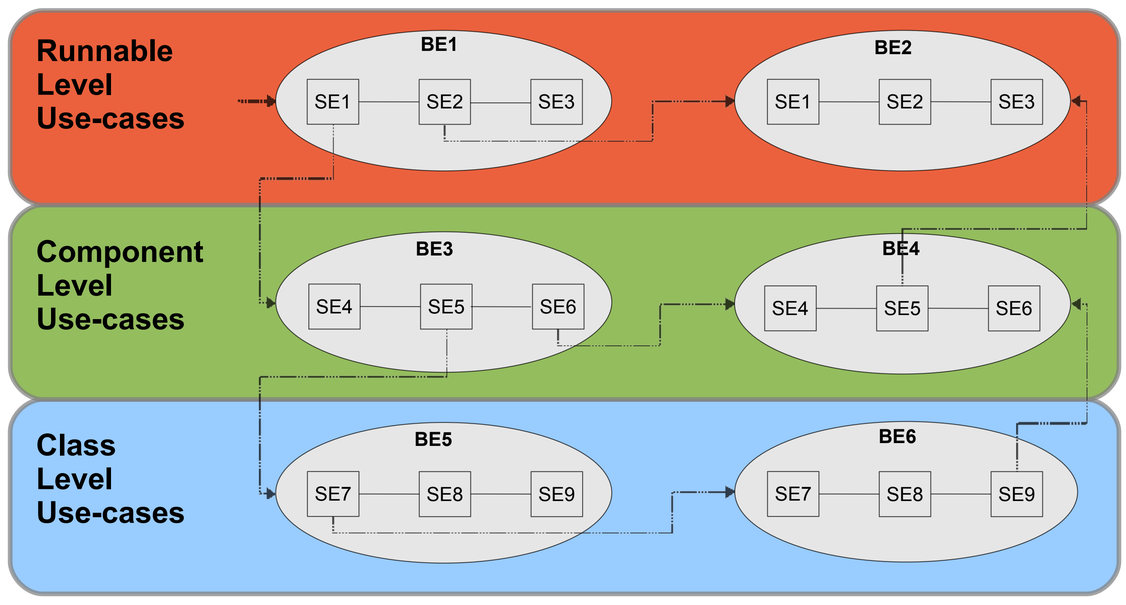
- *What information from the system is necessary for self-management and where to get it?*
  Use-cases guide which structural elements to monitor and where to place sensors.

- *Correct behaviour of structural elements depends on context. How is this dealt with?*
  A use-case provides the *context* that determines correct behaviour of structural elements
  - Each structural element can participate in multiple use-case realisations
  - Analysis of sensored values from monitored structural elements depends on the active use-case

4. Hierarchy of Levels

The self-management framework distinguishes a hierarchy of levels in the structure and the behaviour of a system on the basis of use-case descriptions.

1. *Runnable* level: view of *System Administrators*
2. *Component* level: view of *Functional Analysts*
3. *Class* level: view of *System Developers*



Advantages of multi-levels

- Domain knowledge is acquired from domain experts, each at his/her own level

- Levels divide the problem space into subspaces, each with its own characteristics:
  - *Runnable* level: broken connections, incorrect startup sequence, etc.
  - *Component* level: incompatible component versions, etc.
  - *Class* level: incorrect parameters, uninitialised class members, etc.

- Ability to 'zoom in' on particular areas during the analysis of a problem

IIDS Intelligent Interactive Distributed Systems
http://www.iids.org
vrije Universiteit amsterdam
Supported by:
FORTIS BANK
NL STICHTING