Adaptive Distributed Systems

INF 5360 spring 2008

lecturer: Frank Eliassen

The need for adaptation

- Mobility and pervasive computing, Grids, P2P
  - Variability in operating environment
  - Context dependent user preferences
- Need for dynamic adaptation
  - Retain usability and reliability
- Complexity
  - heterogeneity of networks, nodes and end-user devices
  - personalization of services
- Challenge for users, application and systems developers and managers
- Need for self-adaptation

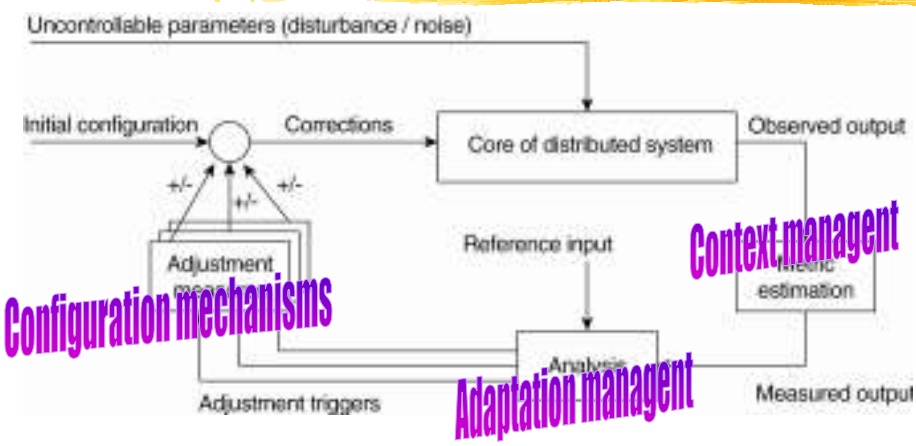
Preliminaries

- Definition
  - Self-adaptive software modifies its own behavior in response to changes in its operating environment [P. Oreizy et al, **1999]**
  - Self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible. [DARPA BAA]
- Assumptions
  - The software has multiple ways of accomplishing its purpose
  - The software has enough knowledge of its own construction
  - The software has the capability to make effective changes at runtime

Self-adaptive software systems

- Automatic adaptation requires strong interplay between system architectures and software architectures
  - how to organize the components of a distributed system such that monitoring and adjustments can be done?
  - where to execute the processes that handle the adaptation?
- A common approach is to organize distributed systems as **feed-back control systems** allowing automatic adaptations to changes
  - Known as **autonomic computing** or **self-* systems**
    - self-*: capturing automatic adaptation in a variety of ways
    - self-managing, self-healing, self-configuring, self-optimizing, etc

The feed-back control loop

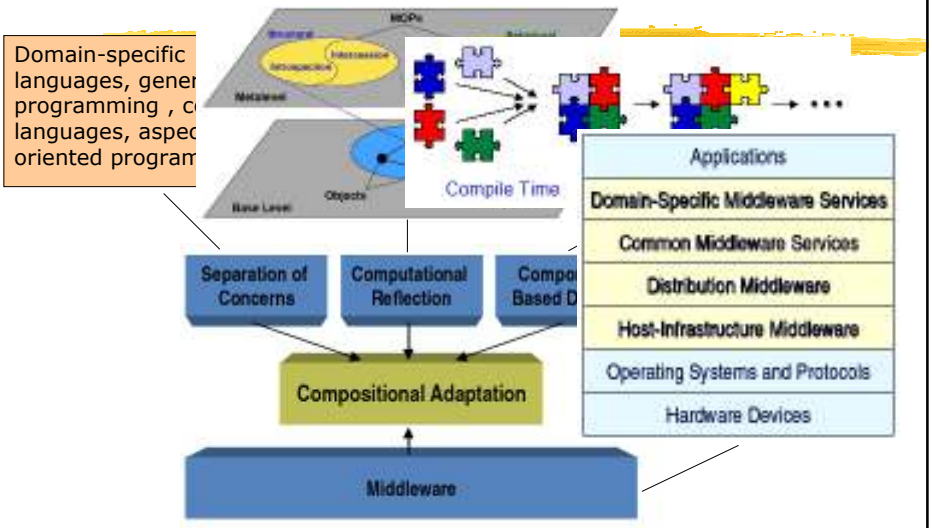


The logical organization of a feedback control system [TvS]

How to realize dynamic adaptations in software?

- Parameter adaptation
  - modification of variables that determine program behavior
    - tuning parameters
    - strategy selection
- Compositional adaptation
  - the exchange of algorithmic or structural parts of the system with ones that improve a program's fit to its current environment
    - enables adoption of new algorithms for addressing concerns unforeseen during original design and construction

Enabling Technologies



Main technologies supporting compositional adaptation.

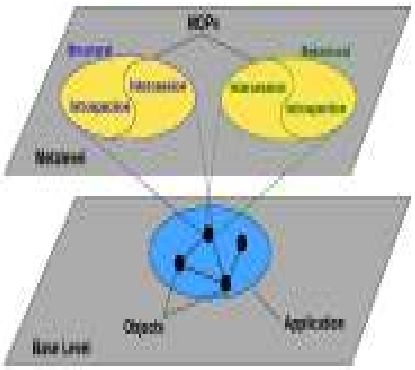
Adaptation mechanisms

- Parameter tuning
- Code (agent or component) migration
- Compositional adaptation
  - aspect weaving (e.g., intercept calls)
  - reflection
  - component-based
    - add, remove, replace, recompose, redeploy, tune (through parameters)

Aspect weaving

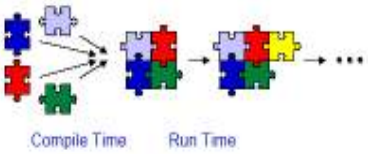
- Aspect oriented programm(AOP)
  - widely used approach for handling **cross cutting conerns** in modularized software
  - cross cutting concerns spans across many modules (QoS, security, fault tolerance)
  - AOP enables separation of cross-cutting concerns into **aspects**
  - aspects are developed separately and woven into the system during compile time (more recent approaches allows weaving during runtime)
  - **pointcuts** define locations in the program where aspect code can be woven

Reflection



- The ability of a system to reason about itself, and possibly, alter its own behaviour
- **Introspection**: the ability to observe its own behavior
- **Intersession**: enables a system or application to act on observation from introspection and modify its own behaviour
- **Base level**: the system itself (code)
- **Meta-level**: self-representation (model) of the system
- **Meta-object protocol (MOP):** interface that enables *systematic* introspection and intersession

Component based design



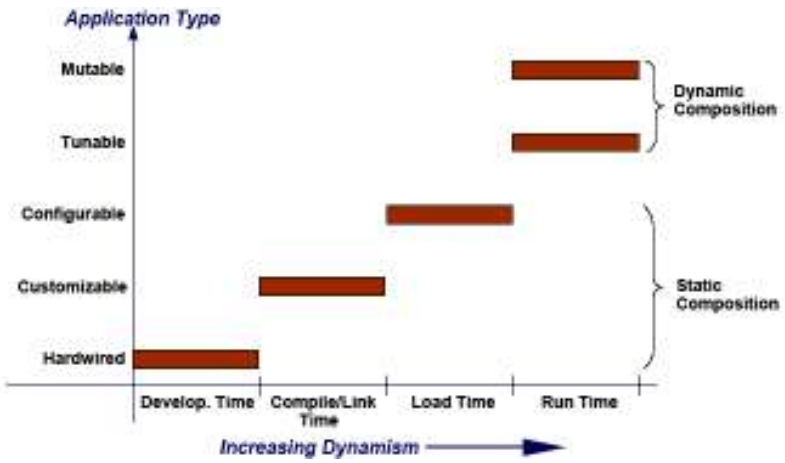
- Perspective: A software system is a network of concurrent components bound together by connectors
- Focus on coarse-grained components and their interactions, and not on the source code level
- Allows run-time rearrangement and replacement of components and connectors

Necessary (NOT sufficient) mechanism for self-adaptation

Where to compose?

- Middleware layers
  - Adaptable comm services
  - Intercept/redirect function calls
  - Open, component based middleware, reflection
  - Aspect middleware, reflection
- Application code
  - Domain-specific languages (entagled code)
  - aspect weaving, composition filters
  - component-based, reflection

Classification for Software Composition (when to compose?)



Adaptation policies

- Techniques for selecting, calculating or deriving the new configuration that fits the current system state and/or context
- Situation-action rules
  - Specifies exactly what to do
  - IF (RT>100msec) THEN (increase CPU by 5%)
- Goal-based
  - specify desired state(s): RT < 100msec
  - system responsible for calculating actions to bring system to desired state
- Utility-based
  - utility-function: ranks all feasible system states
  - $U(CPU)=U(f_{RT}(CPU))$
    - f_{RT} predicts RT from CPU value
  - Adaptation becomes an optimization problem: determine the feasible values of CPU for which U is maximized

Key challenges (1/2)

- Assurance
  - Automated checking of both functional and non-functional properties of the system
  - How to ensure that the system continues to execute in an acceptable, or *safe* manner during the adaptation process?
- Security
  - Protecting the system from malicious entities
  - How to adapt to security regimes that are part of the context, and how to prevent the adaptation mechanism from being exploited by would-be attackers?

Key challenges (2/2)

- Interoperability
  - Coordinated adaptation across system layers and across platforms
  - How to integrate separately developed adaptation mechanisms?
- Decision making (when and how to adapt)
  - Must adapt software while preventing damage or loss of service
  - Learn about and adapt to user behaviour
  - Decision making approaches including their scalability and general applicability
  - Decentralized decision making

Summary

- The need for adaptation is motivated by continuously **changing environment** and **user needs**
- Complexity motivates the need for **self-adaptation**
- Organizing distributed systems as **feed-back control systems** allow automatic adaptations to changes
- **Compositional adaptation** is the main enabling technology
- **Key challenges**: assurance, security, interoperability, decision making

Literature

- **Distributed Systems – Principles and Design,** A. Tanenbaum, M. Van Steen, Prentice-Hall, **2007** (chap 2.2, 2.3)
- **An Architecture Based Approach to Self-Adaptive Software,** P. Oreizy et al, IEEE Intelligent Systems, **1999**
- **Composing Adaptive Software**, P. K. McKinley et al, IEEE Computer, **2004**
- **A Taxonomy of Compositional Adaptation,** P. K. McKinley et al, TR MSU-CSE-04-17, Michigan State University, **2004**.
- **Achieving Self-Management via Utility Functions**, J.O. Kephart, R. Das, IEEE Internet Computing, **2007**