Adaptive Distributed Software Systems

INF 5360 spring 2010

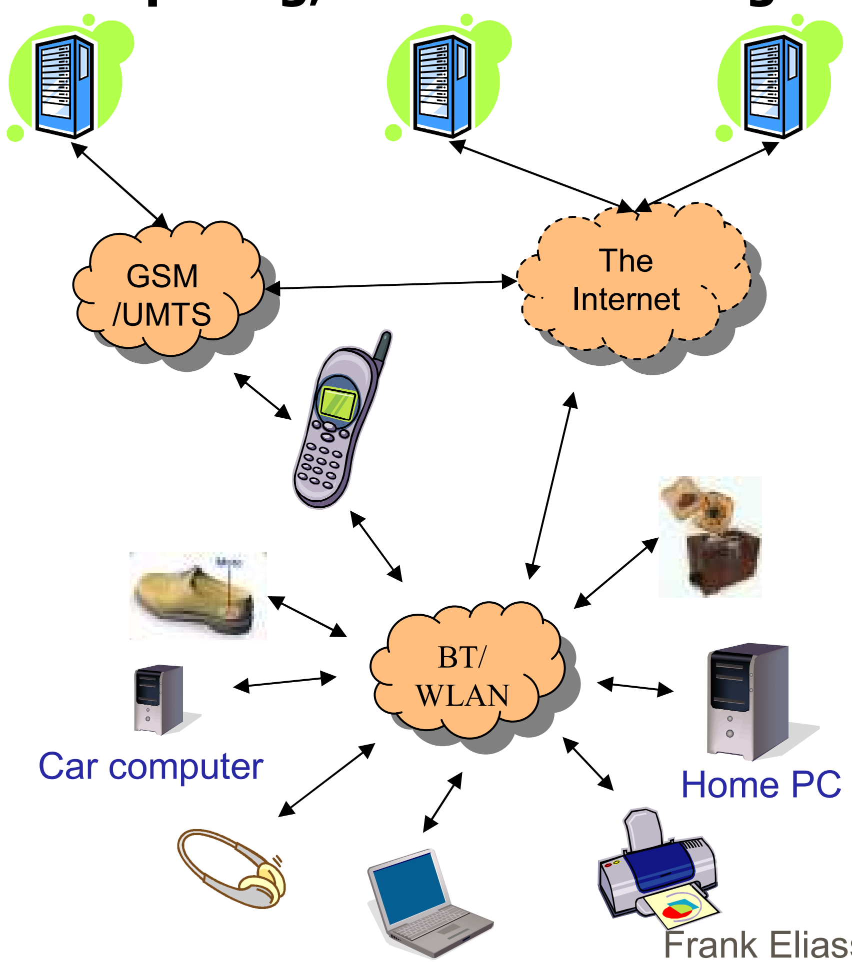
lecturer: Frank Eliassen

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**.
- **Achieving Self-Management via Utility Functions**, J.O. Kephart, R. Das, IEEE Internet Computing, **2007**
- **Self-Adaptive Software: Landscape and Research Challenges**, M. Salehie, L. Tahvildari, ACM TAAS, **2009**

Motivation

- **Mobility and ubiquitous computing, Internet of Things**



When handheld devices are carried by users moving around in ubiquitous computing environments

- ✓ devices/sensors come and go
- ✓ network connections come and go and QoS varies, and therefore
- ✓ services available for use come and go
- ✓ service quality varies
- ✓ user tasks vary and are interleaved with tasks related to movement and social interaction
- ✓ computing resources and power are limited

The need for self-adaptation

- In such environments applications and users will benefit a lot from context awareness and adaptivity
  - Adapt application to resource situation (battery, bandwidth, memory)
    - Example: Dynamically adapt media quality (e.g., video) to available bandwidth
  - Dynamically adapt user interface to situation of user
  - Adapt application to availability of devices and services in the environment (ubiquitous services)
- The demand for applications exhibiting such properties is accelerating
  - Mobile computing
  - Ubiquitous computing
  - Service oriented computing
  - Sensor networks
- Challenge for users, application and systems developers and managers
- Need for self-adaptation

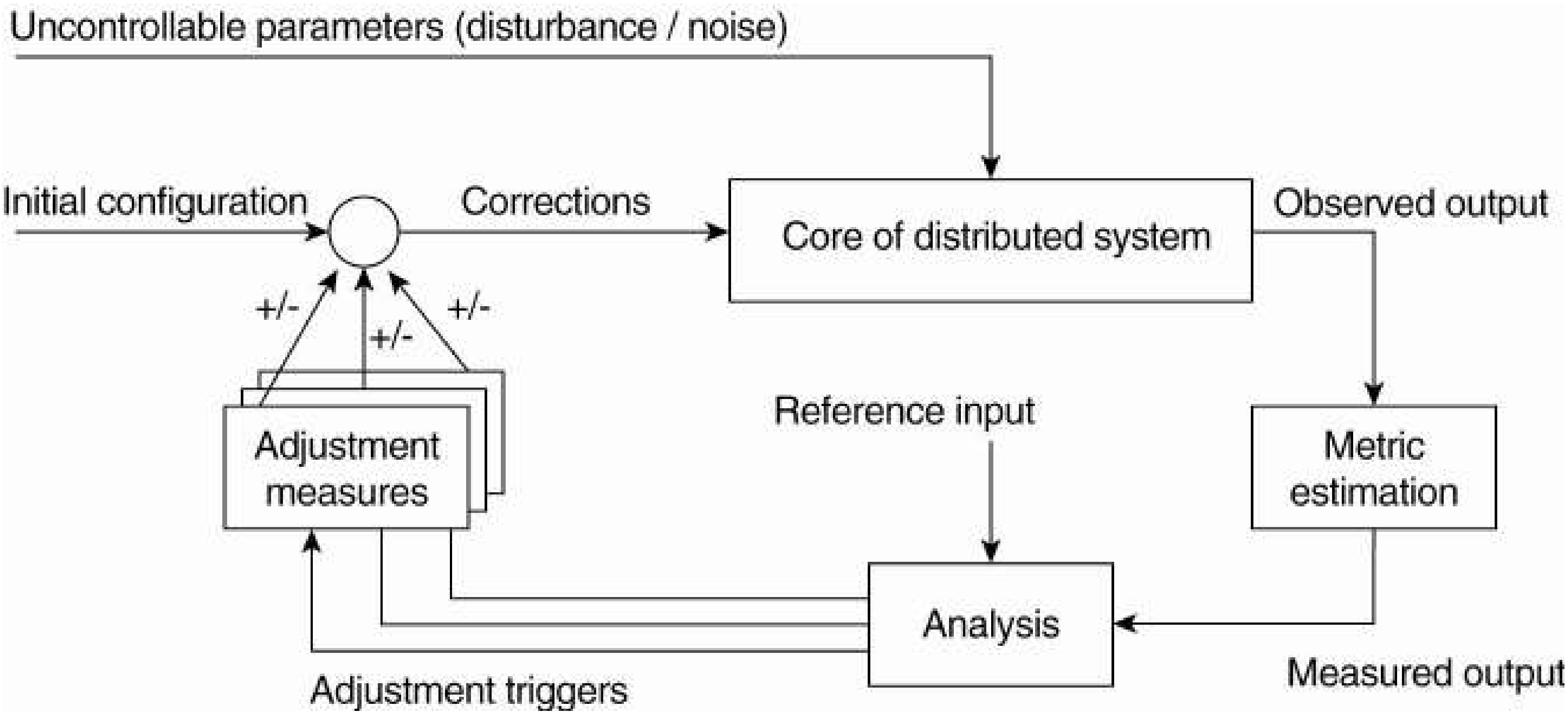
Preliminaries

- 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

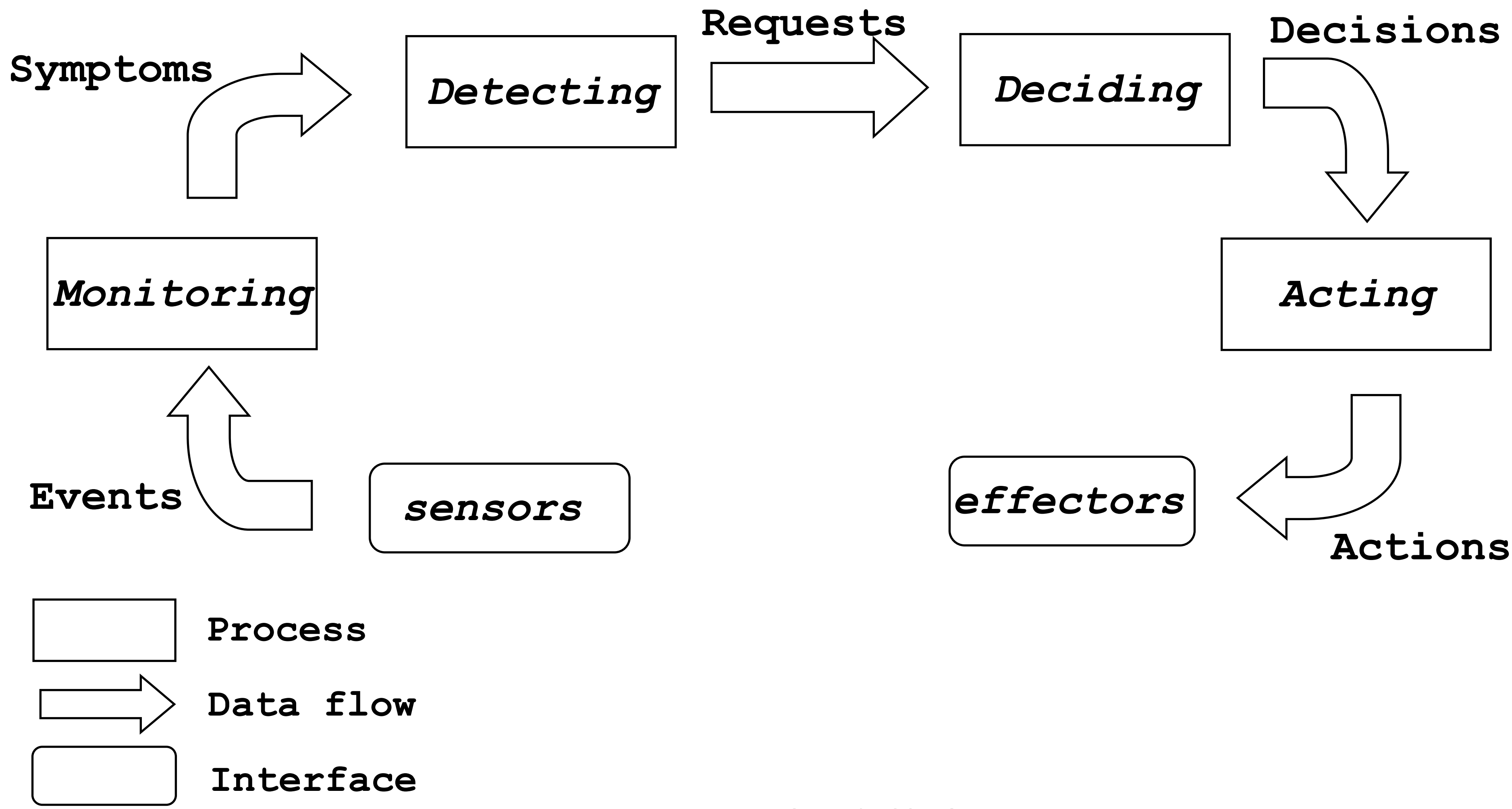
- Automatic adaptation requires strong interplay between system architectures and software architectures
  - how to organize the components of a distributed system such that monitoring and adjustments can be done?
  - where to execute the processes that handle the adaptation?
- A common approach is to 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

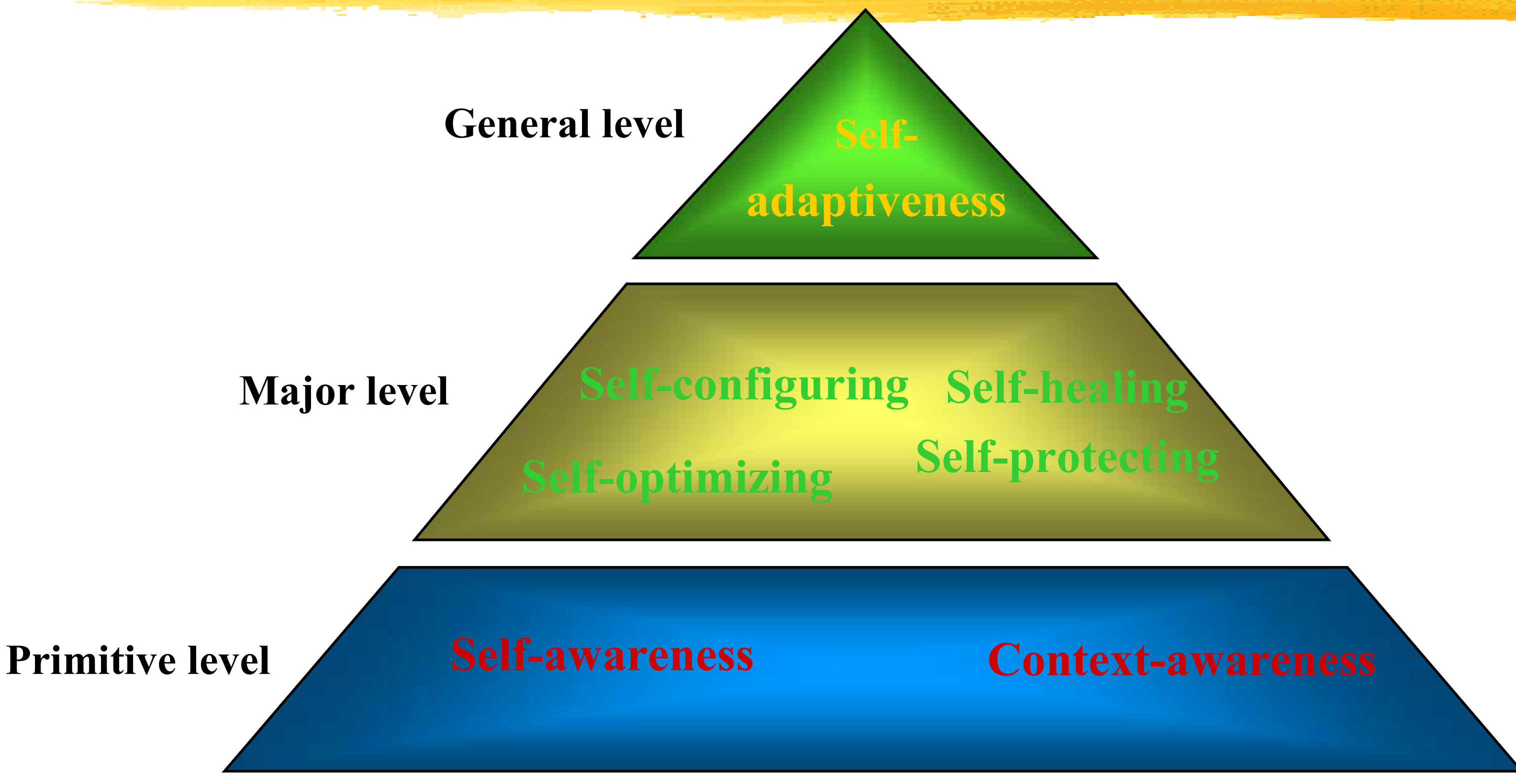


The logical organization of a feedback control system [TvS]

The adaptation loop
according to [Salehie, Tahvildari, 2009]



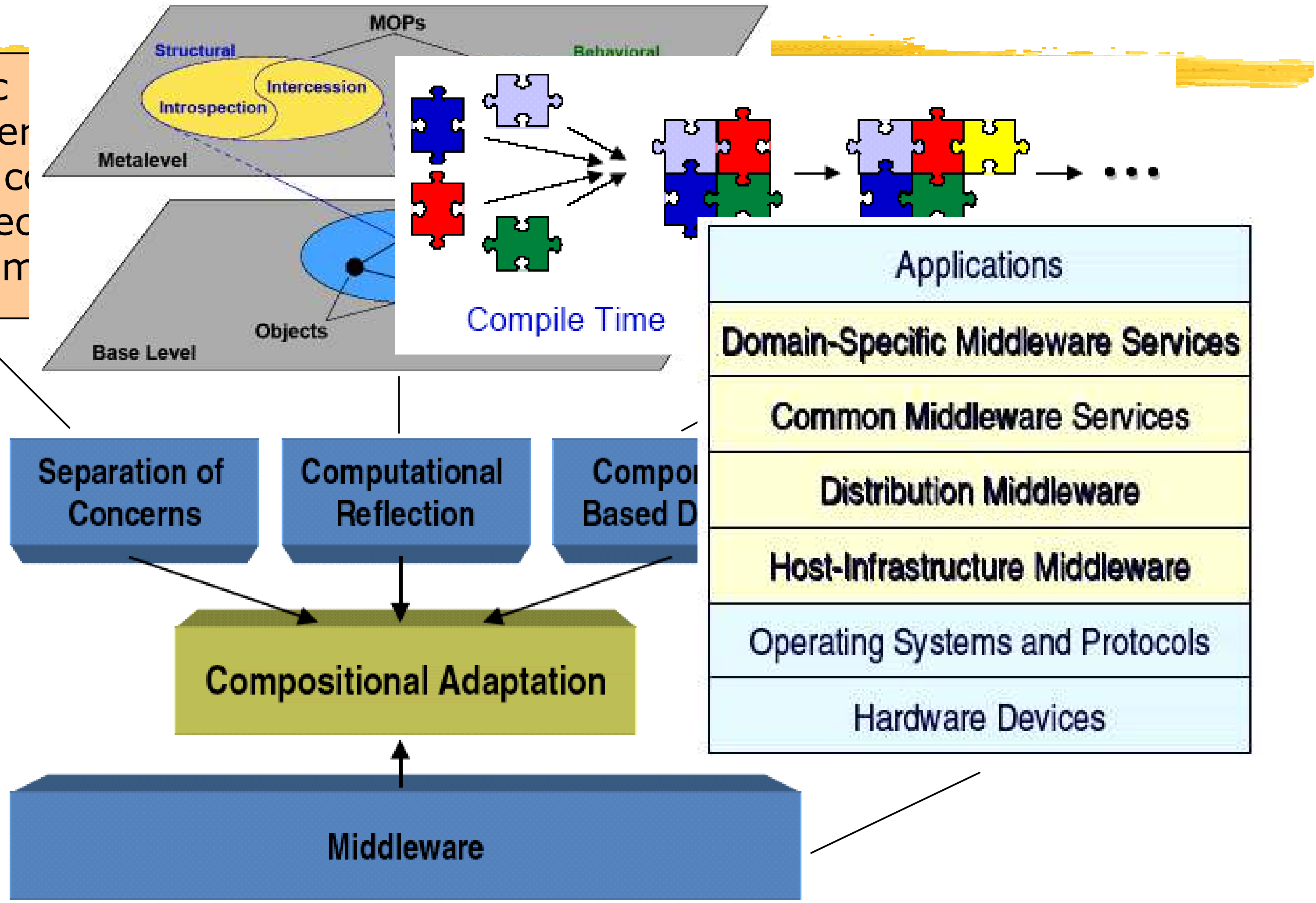
Hierarchy of self-* properties



After: [Salehie, Tahvildari, 2009]

Enabling Technologies

Domain-specific
languages, gener
programming , c
languages, aspec
oriented program



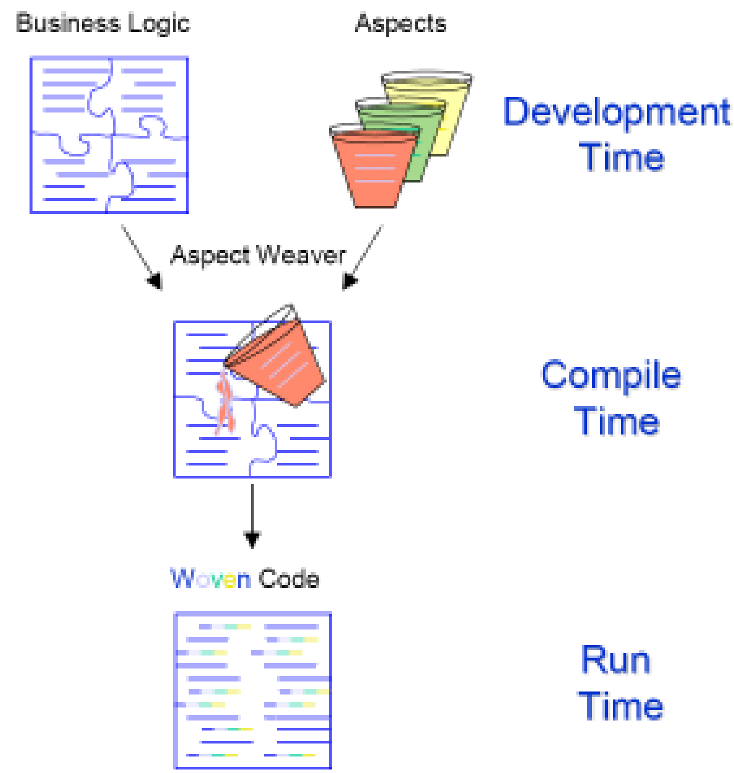
Main technologies supporting compositional adaptation.

Some adaptation mechanisms

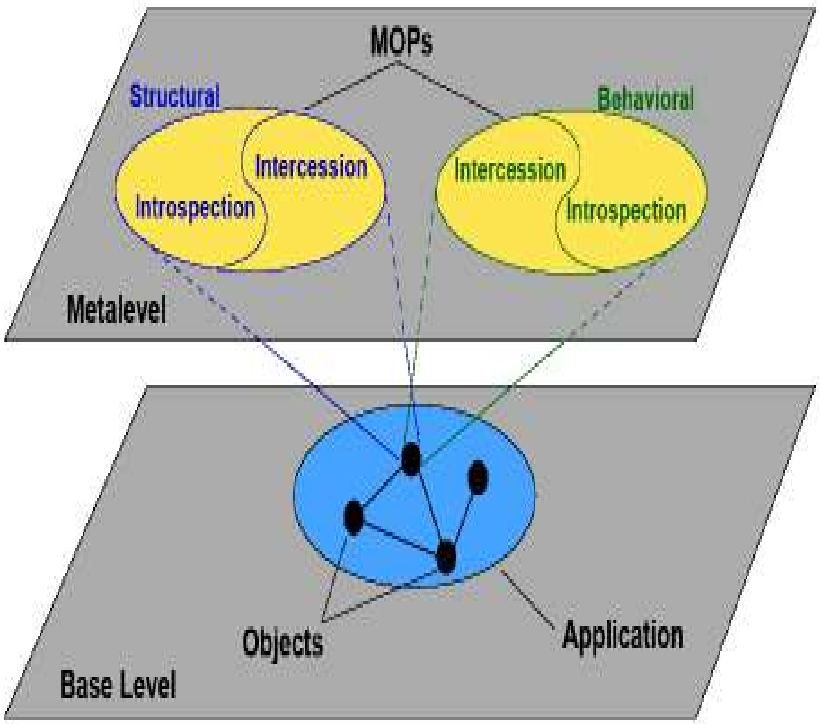
- Parameter tuning
  - modification of variables that determine program behavior (tuning parameters, strategy selection)
- Code (agent or component) migration
- Compositional adaptation
  - the exchange of algorithmic or structural parts of the system with ones that improve a program's fit to its current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

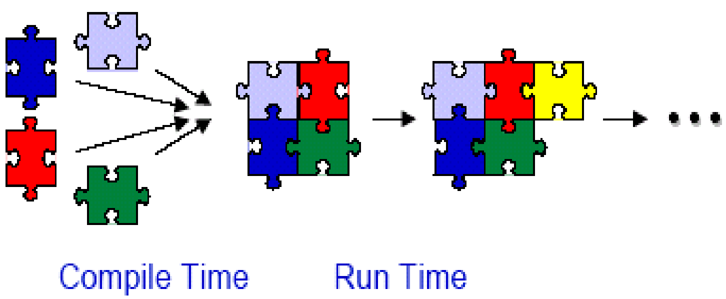


Reflection



- The ability of a system to reason about itself, and possibly, alter its own behaviour
- **Introspection**: the ability to observe its own behavior
- **Intercession**: 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

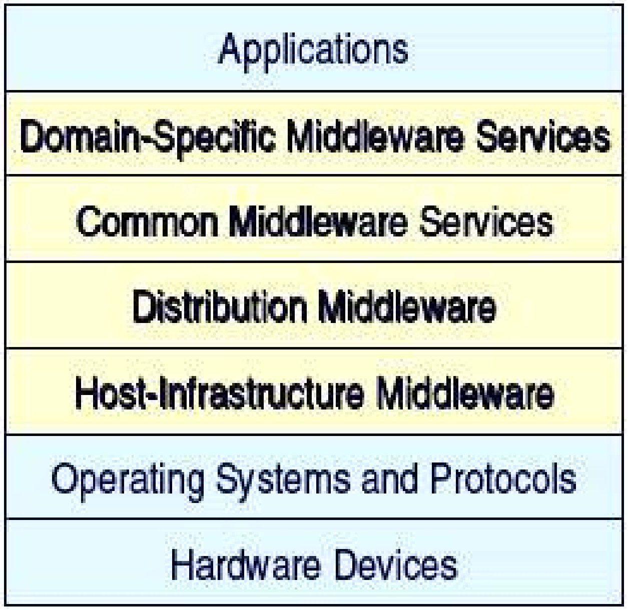


- Perspective: A software system is a network of concurrent components bound together by connectors
- Focus on coarse-grained components and their interactions, and not on the source code level
- Allows run-time rearrangement and replacement of components and connectors

Necessary (NOT sufficient) mechanism for self-adaptation

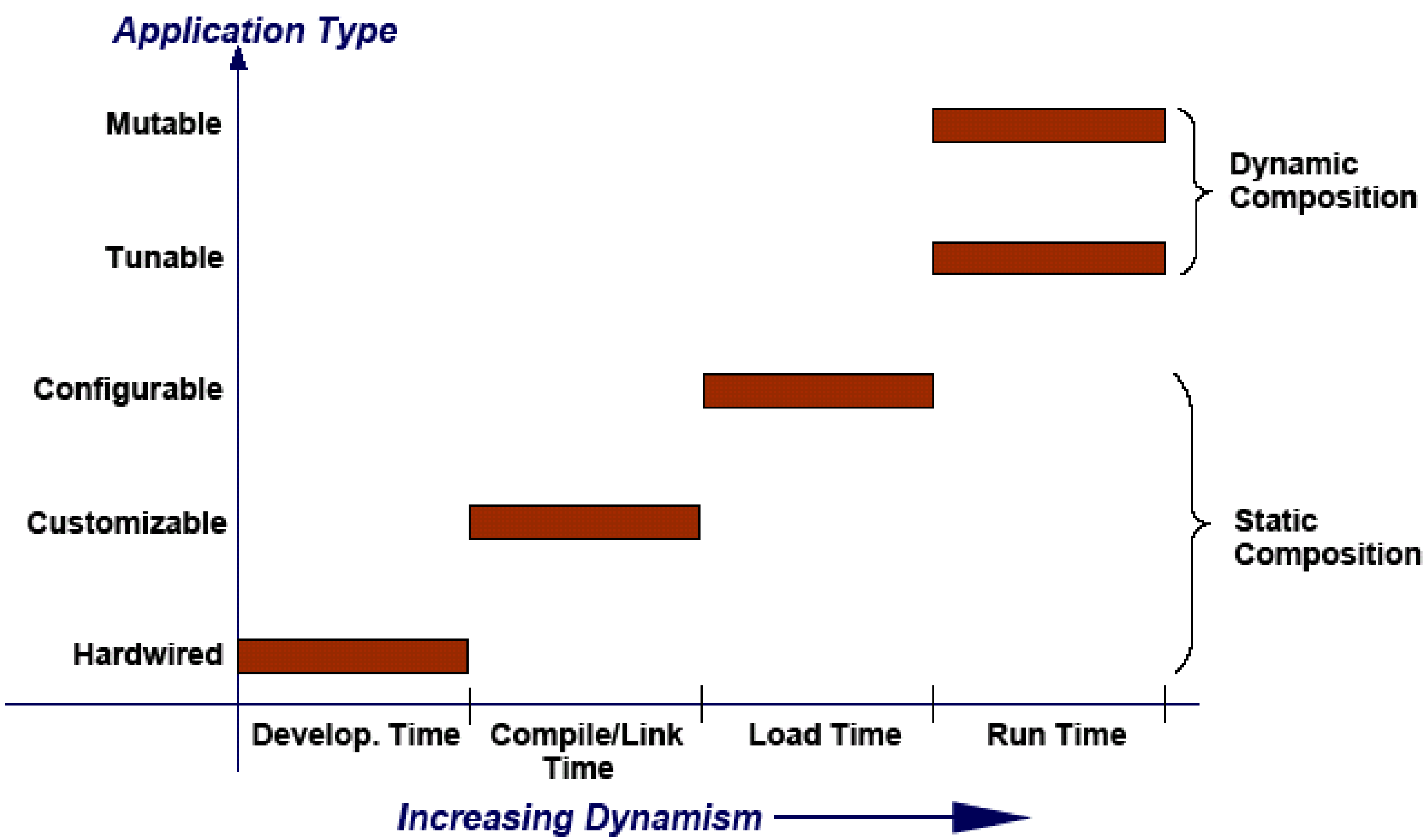
Where to compose?

- 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 (deciding process)

- 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 in each situation
  - 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
- Many others from the AI community

Research challenges (1/4) (Salehie, 2009)

- Engineering
  - Requirements analysis
    - How to capture stakeholders' expectations?
    - How to map from expectations to adaptation requirements and goals to be used at runtime?
  - Design issues
    - How to design self-adaptive software to fullfil adaptation requirements? Which architecture styles? Which component models? How to reengineer legacy systems into adaptive ones?
  - Implementation languages, tools, and frameworks
    - Extending existing programming languages or defining new adaptation languages?
    - Adding, removing and modifying software entities at runtime (compositional adaptation)
  - Testing and assurance
    - Validation of adaptive behaviour
  - Evaluation and quality of adaptation
    - Criteria and metrics for self-adaptive software (safety, security, cost ....), comparing adaptation solutions

Research challenges (2/4)
(Salehie, 2009)

- Self-* properties
  - Individual self-* properties
    - self-protecting and self-healing needs more attention
  - Building multy-property self-adaptive software
    - coordinating and orchestrating more than one self-* property in a single adaptation loop

Research challenges (3/4) (Salehie, 2009)

- Interactions
  - Policy management
    - Policy translation: translate high-level goals into lower level/local ones understandable by the system elements
    - Dynamic policies and goals
      - policies and goals that can be changed during the operating phase
  - Building trust
    - Self-adaptive system are harder to trace for users and stakeholders
  - Interoperability
    - Coordinating and orchestrating self-adaptation behaviour of several subsystems ("systems of systems")

Research challenges (4/4)
(Salehie, 2009)

- Adaptation process
  - Monitoring challenges
    - reduce cost/load of sensors
    - make monitoring process adaptive: only load sensors that are needed and unload when not needed
  - Detecting challenges
    - deciding which behaviours/states that are "unhealthy" and that requires adaptation to be considered
  - Deciding challenges
    - dynamic adapation policies
    - finding approximate or suboptimal solutions (scalability/effciendcy)
    - dealing with uncertainty and incompleteness of events/information from *self* and *context*
    - correlating local and global decision making
    - scalability of decision making
  - Acting challenges
    - satisfy constraints, safety/integrity and fault-tolerance

Key challenges (1/2)
(McKinley 2004)

- 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)
(McKinley 2004)

- 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
- **Research challenges (Salehie)**: in the areas of engineering, self-* properties, interactions, adaptation process
- **Key challenges (McKinley)**: assurance, security, interoperability, decision making