Mobile and Ubiquitous Computing 2023-24
MEIC/METI - Alameda & Tagus

Adaptability

Contents

- Definition of adaptability
- Variable context/resources
- Trading-off adaptability and invisibility
- Measuring the effectiveness of adaptability mechanisms
- Adaption layer (operating system, middleware, application)
- Architecture for adaptability
- Adaptability policies

Definition of Adaptability

- **Adaptability** (Latin: *adaptō* "fit to, adjust")
  - is a feature of a system or of a process
  - quality of being adaptable; a quality that renders adaptable
  - variability in respect to, or under the influence of, external conditions

- A computer system is said to have the capability of being **adaptable**:
  - if it changes **automatically** to suit **variable** working conditions,
  - while offering the best **quality of service** possible,
  - that is acceptable by the user.

Variable Context/Resources (1/2)

- Due to their intrinsic nature, execution environments in mobile computing suffer from **great and diverse variations** during applications execution.
- These variations can be:
  - **qualitative** (e.g., network connection or disconnection, specific devices such as printers in the device neighborhood, consistency and security constrains), or
  - **quantitative** aspects (e.g., amount of usable bandwidth, memory available).
- Applications should be able to **automatically deal with this variability**
- Applications programmers should **not be forced to account for every possible scenario** in their coding as this would be:
  - inefficient, error-prone, and limited to situations accounted for *a priori*

Variable Context/Resources (2/2)

- Which resources should the system adapt to ?
  - **system**: network, energy, CPU, memory, etc.
  - **environment**: luminosity, temperature, etc.
  - etc.
- An adaptable system must take into account:
  - **all resources** whose quality/availability may change (with time/location/etc.) while the **application is running.**

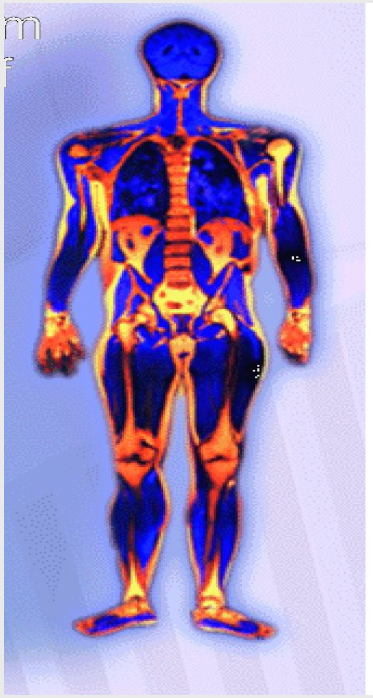
Trading-off Adaptability and Invisibility

- Correct **adaptability** is key for **transparent operation.**
- If a systems does not adapt at all:
  - **stops working**
- If a system adapts wrongly:
  - **It may malfunction**
- Both cases make the adverse conditions **visible**!
- Being adaptive implies:
  - **detecting changes** and **acting** accordingly.
  - **predicting changes** to allow **timely adaptation** (sometimes, in advance).
  - Ideally in an **autonomous** way...

Characteristics of an Autonomic System

- In the original IBM proposal ("An architectural blueprint for autonomic computing", a *de facto* standard):
  - Self-Configuration
  - Self-Optimization
  - Self-Repair
  - Self-Protection

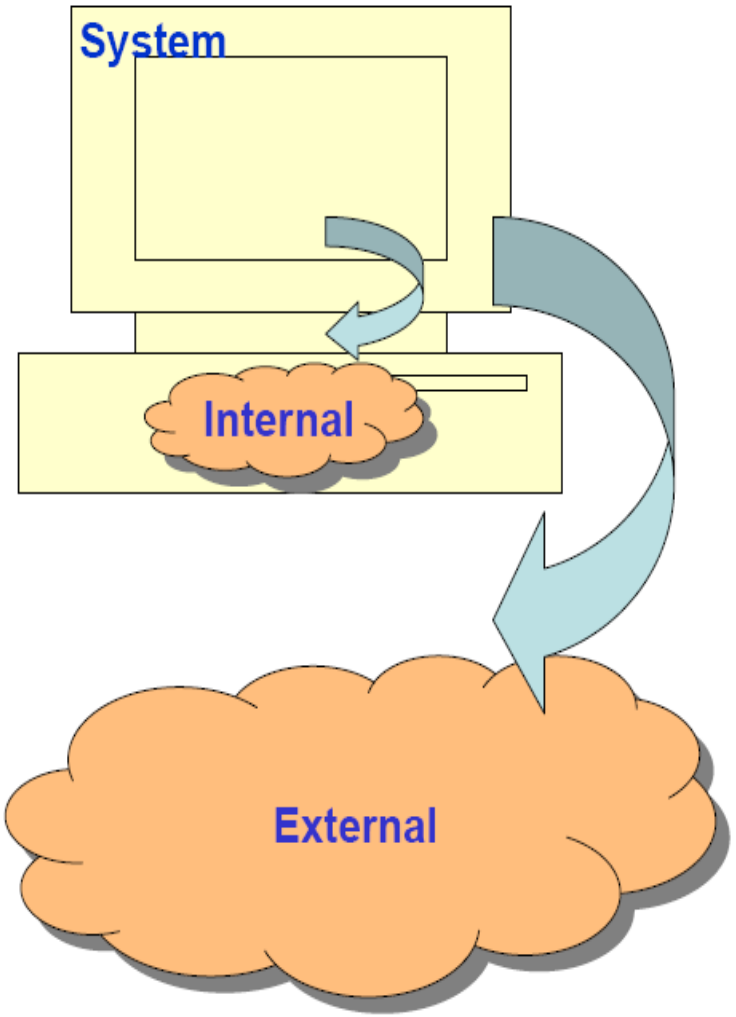
Self-* or Self-X Properties



Self-Configuration

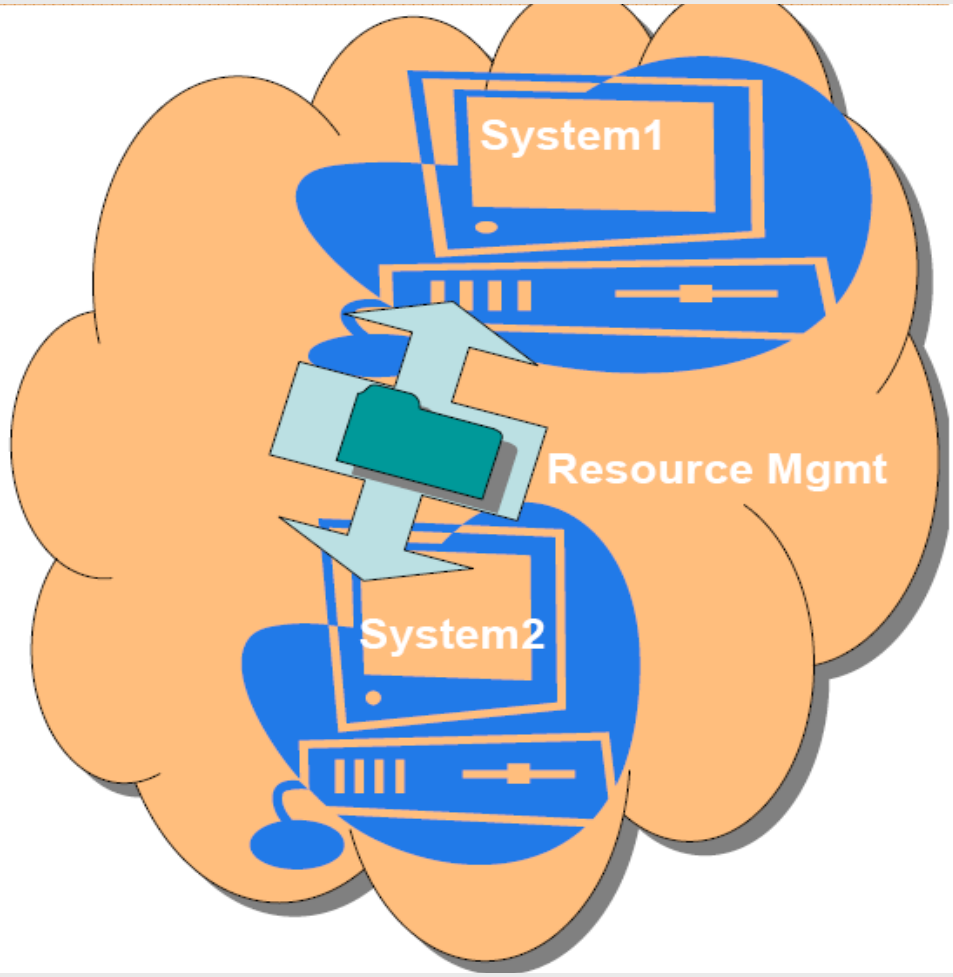
- Adapt automatically to the dynamically changing environment
- Internal adaptation
  - *Add/remove new components (software)*
  - *configures itself on the fly*
- External adaptation

  Systems configure themselves into a global infrastructure



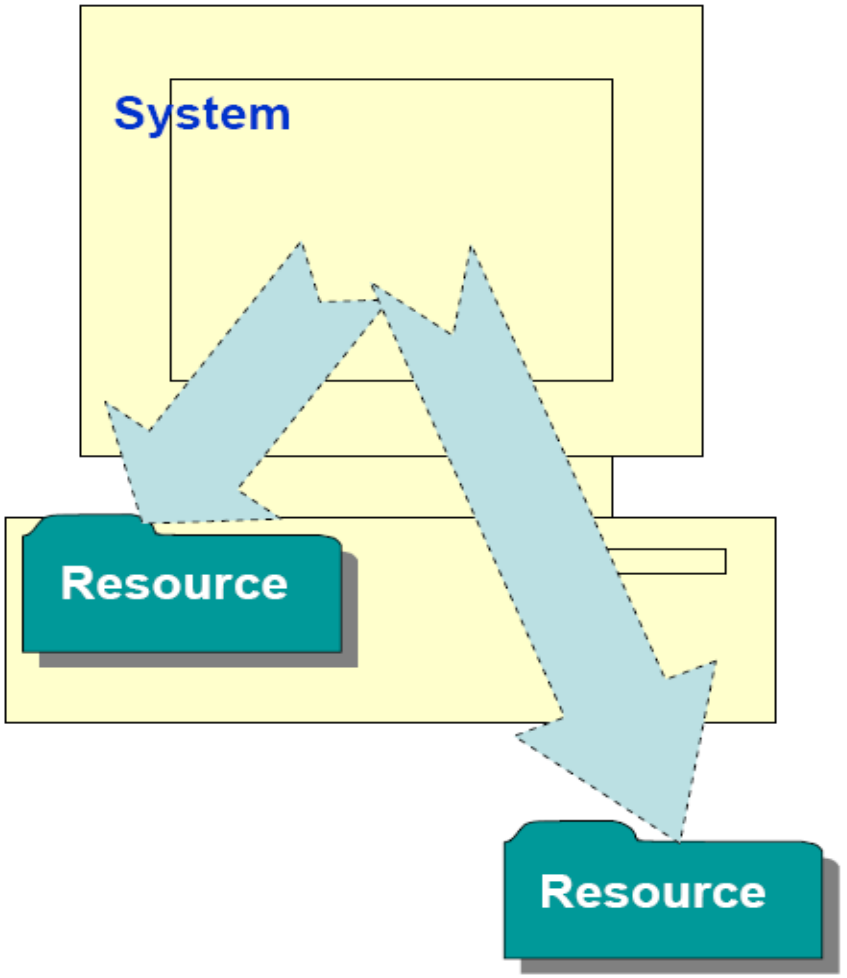
Self-Optimization

- Monitor and tune resources automatically
  - *Support operating in unpredictable environment*
  - *Efficiently maximization of resource utilization without human intervention*
- Dynamic resource allocation and workload management.
  - *Resource: Storage, databases, networks*
  - *For example, Dynamic server clustering*



Self-Protection

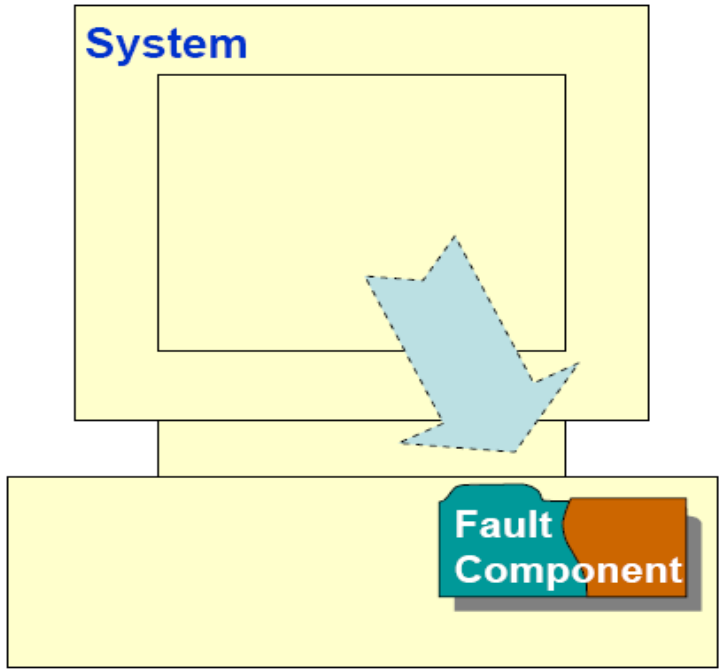
- Anticipate, detect, identify and protect against attacks from anywhere
  - *Defining and managing user access to all computing resources*
  - *Protecting against unauthorized resource access, e.g. SSL*
  - *Detecting intrusions and reporting as they occur*



Self-Repair

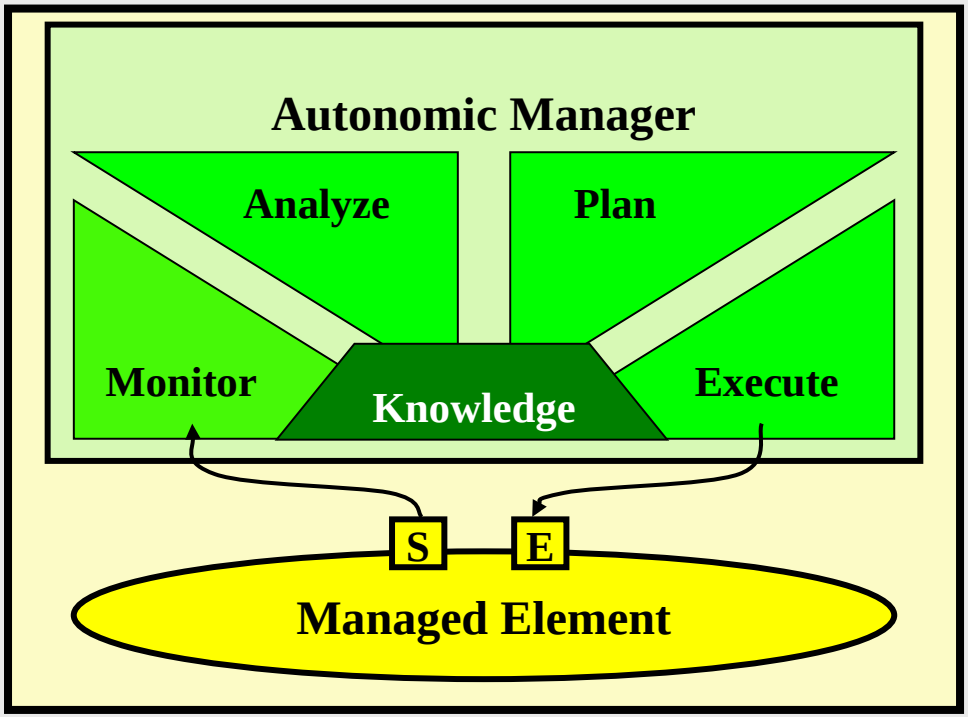
- Discover, diagnose and react to disruptions without disrupting the service environment

- Fault components should be
  - *detected*
  - *Isolated*
  - *Fixed*
  - *reintegrated*



Autonomic Elements: Structure

- Fundamental atom of the architecture:
  - Managed element(s):
    - Database, storage system, server, software app, etc.
  - Plus *one* autonomic manager.
  - The **MAPE-K** model.
- Responsible for:
  - Providing its service.
  - Managing its own behavior in accordance with policies.
  - Interacting with other autonomic elements.



An Autonomic Element

Monitoring

- Capture environment properties (physical or virtual) that are relevant for decision.
- Highly implementation dependent component.
- Sensors read properties from the managed component: requests per second, power consumption,...
- Passive monitoring (no changes to system): e.g. /proc folder in Unix.
- Active monitoring (with changes to the system):
  - e.g. ProbeMeister, Pin (code injection for monitoring).
  - May require adaptive monitoring so as not to influence performance.

Analysis

- Combine monitoring symptoms into higher level descriptions.
- Process event streams according to policies:
  - Which sequences are to be interpreted?
  - Which sequences are discarded?
- May include the ECA (event-condition-action) component.
- Borders between Monitoring/Analysis/Planning are not rigid.

Planning

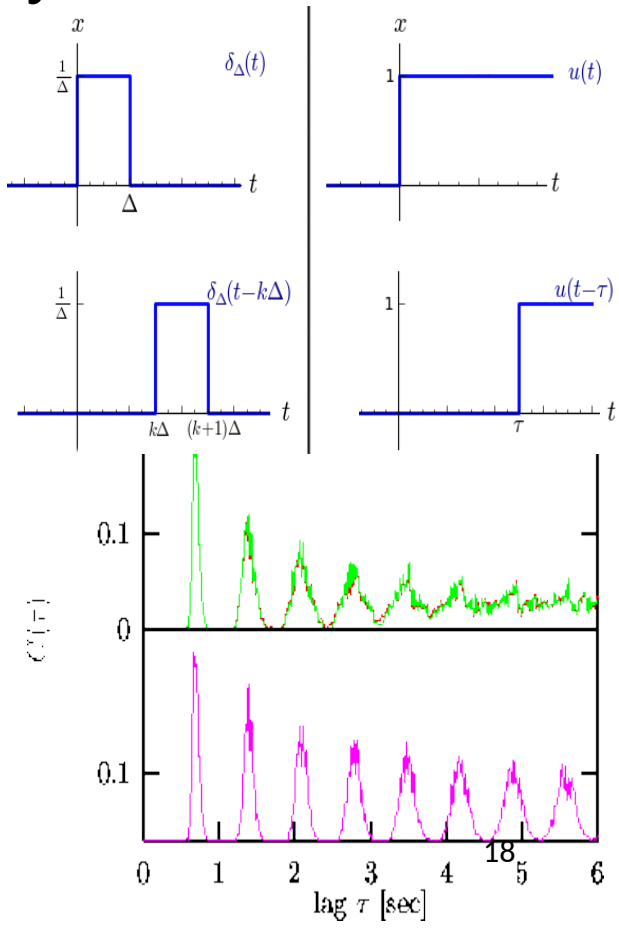
- Based on monitoring data decide changes to apply to the managed element.
- Two approaches:
  - ECA: a set of stateless rules to decide what actions to take. Often require conflict resolution mechanisms.
  - Model based approach: represent the managed element in a model. Actions are applied first on the model in order to detect problems and inconsistencies.

Execution

- Execution of the planning decisions.
- Based on dependencies between actions and intervention opportunities:
  - Convert the plan into a workflow.
  - Schedule executable actions.
  - Execute local and eventual remote actions.

Criteria for the Effectiveness of Adaptability Mechanisms

- User satisfaction
- “Dirac/Spike” changes on resources availability
- Reaction speed
- System stability
- Portability of the solutions
- Resource **usage** of the adaptability solutions



Adaptability Policies

- Adaptability can be provided through the enforcement of **declaratively-defined** policies supported by the middleware:
  - policies are not hard-coded in applications, and
  - can be deployed, enforced, and updated at any time
- Such middleware relies strongly on the following features:
  - the extensible capability to support the **specification and enforcement of runtime management policies**,
  - a set of **pre-defined policies** to control the mechanisms previously mentioned
  - for example: a pluggable set of basic mechanisms supporting object replication

Adaption Layer (operating system, middleware, application)

- At what level should the adaptability solution be implemented?

- Operating system:
  - oblivious to application semantics
  - global/equal solutions to all
  - implications with OS portability
  - no application awareness

- Middleware:
  - common mechanisms can be reused
  - interaction with applications to be semantic aware
  - portable

- Application:
  - semantic aware
  - specific for each application
  - not portable for different applications
  - With the issue of conflicting choices between applications

Conclusion

- What is adaptability ?
- Measuring how effective the system is w.r.t. automatic adaptability
- The autonomic computing model
- Adaption layer (operating system, middleware, application)
- A software architecture for adaptability
- Adaptability policies
- Access to objects (replication, consistency, transactions/sessions,failure)