Developing highly complex distributed systems: A software engineering perspective

Marco Autili, Paola Inverardi, Patrizio Pelliccione, Massimo Tivoli

Software Engineering and Architecture Group
Computer Science Department, University of L’Aquila, Italy

13 December 2011, FOME Middleware 2011, Lisbon
What is a **HCDS**?

*Highly Complex Distributed Systems*

HCDS are paradigmatic of new developing trends

3 domains:
*Context Aware Systems, Service Oriented Systems, Cloud Systems*

3 questions:
What are the needs that may drive the development of such systems?
What is their life cycle?
Is there any new challenge for Software Engineering and Middleware?
Driving needs for developing HDCS

Distinctive feature: *variability of execution context*

**Context**: the combination of user-centric data (e.g., information of interest for users like requested QoS) + resource/computer-centric data (e.g., status of devices and network like availability of resources)

Variability cannot in general be predicted at development time thus calling for *(dependable)* **adaptiveness** at execution time.

This moves developing activities to run time and asks for new and more efficient techniques to support run-time activities.
Driving Needs in developing HDCS

Users express different needs and expectations. This may result in dynamic requirements changes.

HDCS contexts are multi-purpose, highly heterogeneous and fluctuant.

Software has to be able to evolve, react and adapt to a continuously changing environment, while guaranteeing dependability.
The **process view**: the focus is on the set of activities that characterize the production and the operation of a software service/system

Growing **Complexity** of Software has exacerbated the dichotomy *development/static* time vs *execution/dynamic* time

↓

Despite complexity HDCS life cycle requires a radically new approach
**HDCS Life Cycle**

*Evolutionary perspective:* perpetually (life long) adapting systems

*Not a priori fixed distinction among development and execution time.* Accordingly, system artifacts and related tools/techniques need to live at run time.

Run time evolutionary activities *meet* middleware

*Efficiency concerns* ask for minimizing run time support

The amount of required run time support *depends on the application*
Moving to run time

\[ P_S = \text{Standard Engineering Process} \]
\[ P_{EV} = \text{Perpetual Engineering Process} \]

- \( P_S = \) Standard Engineering Process
- \( P_{EV} = \) Perpetual Engineering Process

Diagram shows a transition from a frozen system to a running system, with models from 1 to \( N \). The diagram illustrates the process of compiling and running the system, with cores and adaptive middleware.
HCDS Life Cycle

Develop
- System synthesis starting from user centric requirements for both application and middleware

Analyze
- Producing observation models of the system with feedbacks concerns driving evolution

Observe
- Observe the execution environment and the application behavior and propagate to the models
The perpetual engineering process model
Many approaches to the development of CAS from programming languages to middleware

Vast heterogeneity induced by the mobile devices and the ubiquitous environment

Hybrid approach that allows the choice of the right trade off for each application between application specific adaptation concerns, parametric adaptation patterns and general adaptation mechanisms.
SOComputing promotes a development paradigm based on the dynamic integration of loosely-coupled autonomous services.

- Services are autonomously developed → heterogeneous technologies and middlewares.
- Choreographies emerge as the dynamic composition framework for large scale service-based HDCS, only roles are fixed at development time.
- Impossible to assume once and for all middleware-interoperable framework.
- Both the application and the middleware are required to evolve
Everything as a service modular and distributed virtualization of a system at different levels IAAS, PAAS, SAAS.

Middleware for CS follows the layer-oriented separation-of-concerns

However HDCS development requires a cross-cutting approach to (best) tackle the dynamicity of the application.
What are the most related research areas?

- Support to (dynamic) system configuration (costs/benefit analysis)
- Models@runtime (from req to code to process)
- Dynamic Software Architectures, integration paradigms and connector synthesis
- Functional and non-functional dynamic verification and validation
What are the challenges for middleware?

Adaptive, scalable, malleable Middleware

Building blocks for:
- context management
- adaptiveness (control loop, feedback management, code synthesis, ...)
- configuration management
- life cycle support
- ...

Strong multi-disciplinary cooperation
Questions?

Developing highly complex distributed systems: A software engineering perspective

Marco Autili, Paola Inverardi, Patrizio Pelliccione, Massimo Tivoli

Software Engineering and Architecture Group
Computer Science Department, University of L’Aquila, Italy

13 December 2011, FOME Middleware 2011, Lisbon