Monterey Workshop 2006
October 16-18, 2006 – Paris, France

Reliability and Flexibility Properties of Models for Design and Run-time Analysis

Manuel Rodríguez
National Research Council

Jointly with **Luqi, V. Ivanchenko, V. Berzins**

Naval Postgraduate School
Monterey, CA
USA

Introduction

- Large variety of methods, models and assisting tools for software development and analysis
- Systems of Embedded Systems (SoES) require new capabilities
  - Shortcoming of existing tools: poor means for representing system interactions, concurrency, interoperability and scalability
  - SoES needs modeling of the system and the environment (e.g., physical devices)
- Review of modeling paradigms and how they affect properties of SoES
  - Trade-off between reliability and flexibility
  - Documentation Driven Development (DDD) & Agent Based System (ABS)

Outline

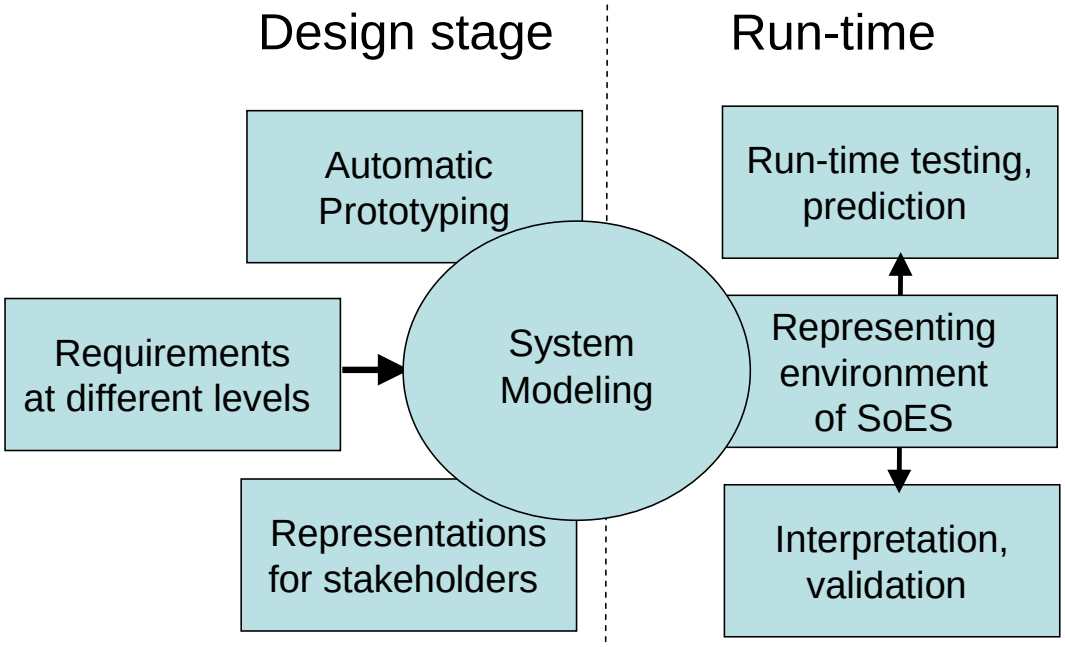
- Model-driven development & analysis
- Documentation driven approach
  - Documentation Driven Development (DDD)
- Agent Based Systems (ABS)
- Conclusions

Outline

- Model-driven development & analysis
- Documentation driven approach
  - Documentation Driven Development (DDD)
- Agent Based Systems (ABS)
- Conclusions

Model-driven development & analysis

- SoES needs a framework to connect constraints to a large variety of requirements
  - Task complexity, multiplicity of SW/HW platforms and protocols
- Model-driven development
  - Suitable paradigm applied to both design and run-time analysis



Model-driven development & analysis

- System design
  - Multi-level representation of requirements
  - Software elements for different stakeholders at different degrees of detail
  - Imposing constraints of various scopes and automatic prototyping
- Run-time analysis
  - E.g.: testing, prediction, interpretation of results
  - Simplified model of interactions to limit number of behaviors
    - Architectural models oriented to interactions and behaviors are preferred to structure-oriented architectures
    - Managing complexity facilitates dynamic analyses as testing

Outline

- Model-driven development & analysis
- Documentation driven approach
  - Documentation Driven Development (DDD)
- Agent Based Systems (ABS)
- Conclusions

Documentation driven approach

- Documentation plays a key role in software development
  - Informal/formal representations
- Open challenges in documentation technology
  - Information consistency across development phases
  - Increase of intellectual burden on stakeholders
  - Need for transformations
  - Inefficient support for complex real-time systems
- Example: Sensor-network based systems
  - System requirements are constantly changing
  - Success depends on being able to accommodate requirements changes and system extensions to address emerging requirements
  - This flexibility should not compromise the system dependability
- Documentation Driven Development (DDD) addresses these problems

Documentation Driven Development (DDD)

- DDD features
  - Documentation structured into computational and design models
  - Models and simulations included in documentation
  - Automated decision support and representation in multiple formats
  - Computer-aided design tools driven by documentation
  - Promptly adaptation to new requirements and support for diverse stakeholders, while preserving high-confidence and timing constraints
  - Agility of software development and support for partial automation

Documentation Driven Development (DDD)

- Classification of information
  - For tools
    - E.g.: mathematical notations, design languages, programming languages, system models, requirements/design specifications, ontologies, source code, test cases, databases, etc.
  - For humans
    - E.g.: natural language text annotations, decision tables, spread sheets, computed attributes. Also video, audio/clips, live simulations, queries, etc.
- DDD divided into:
  - Document Management System (DMS)
  - Process Measurement System (PMS)

Documentation Driven Development (DDD)

- Documentation Management System
  - Create, organize, monitor, analyze, manipulate, and display docs
  - Record documentation (reqs. specs, models, design rationale, stakeholder inputs, project management information, etc.)
  - Extract relevant information from all development phases
  - Provides a Documentation Repository, Representation Converters, and Transition Drivers
- Process Measurement System
  - Obtain necessary information from the documentation repository
  - Metrics & measurement models
    - Measure system's high confidence properties
    - Assess the effort and success probability of the project
    - Monitor changes in system requirements
  - Analysis results presented to developers and users as feedback

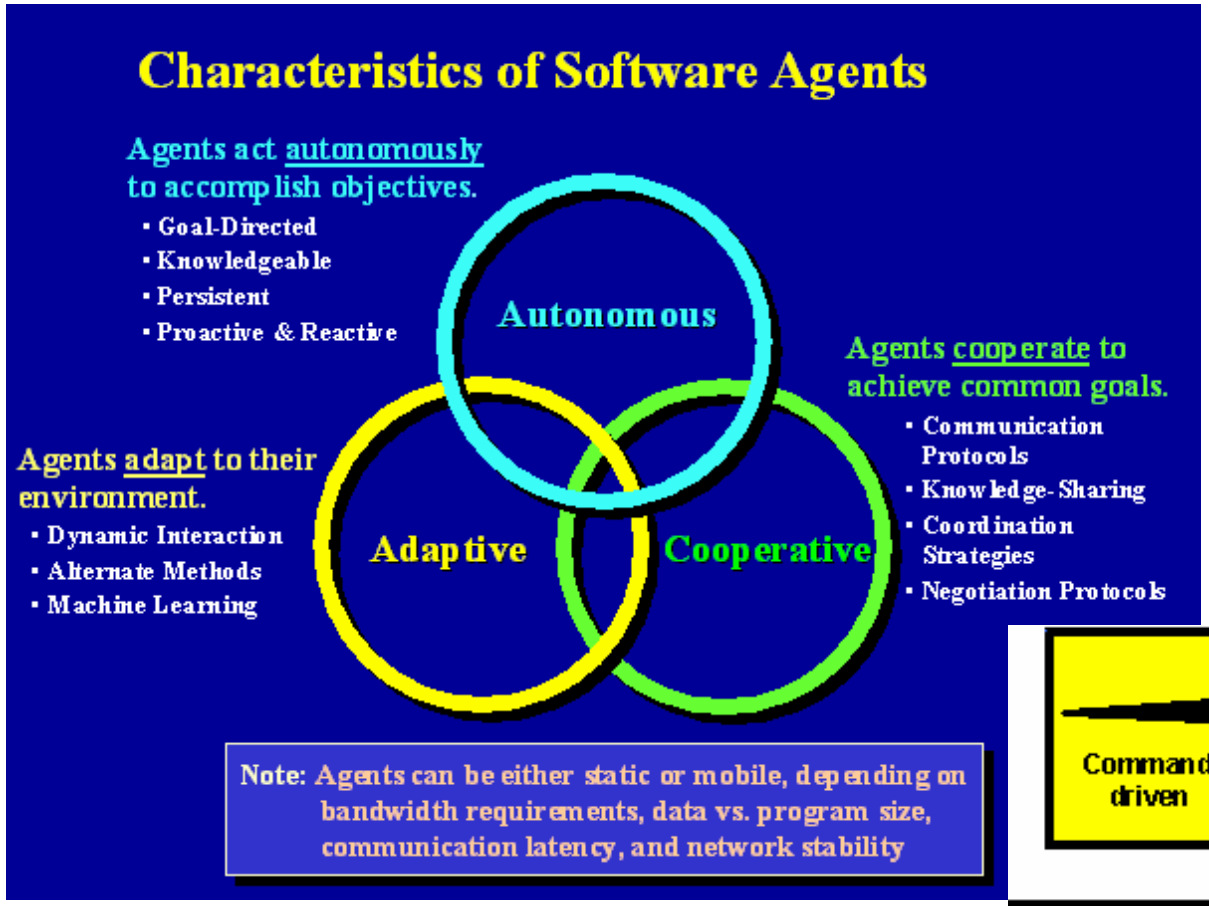
Outline

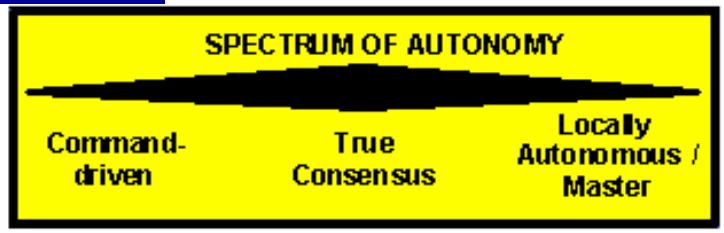
- Model-driven development & analysis
- Documentation driven approach
  - Documentation Driven Development (DDD)
- Agent Based Systems (ABS)
- Conclusions

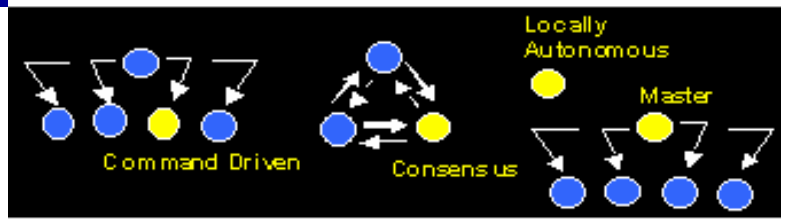
Agent Based Systems (ABS)

- Different definitions
  - Behavioral vs. structural based
  - Structural (IMPACT)
    - Set of data types, action constraints and integrity constraints
    - Set of API functions and actions implemented in any language
    - Notion of concurrency and program
  - Behavioral (DARPA)
    - Autonomously accomplish objectives
    - Adapt to the environment
    - Cooperate to achieve common goals

Agent Based Systems (ABS)







Agent Based Systems (ABS)

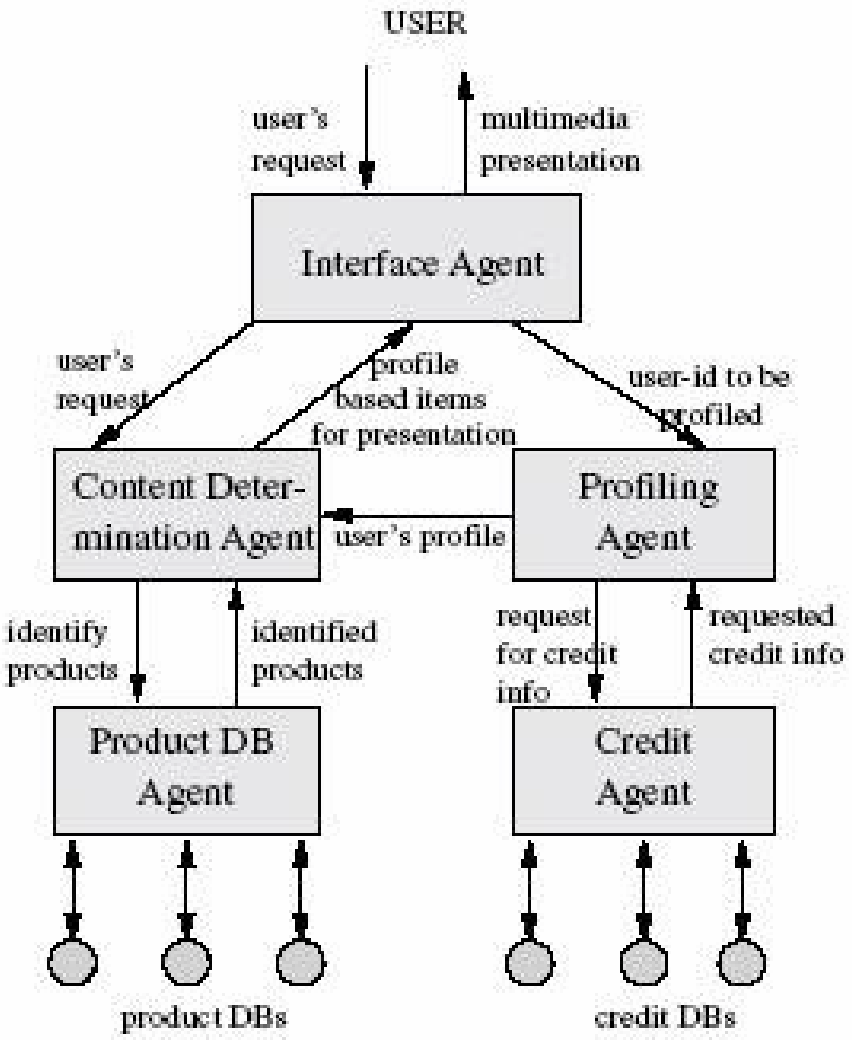
- Advantages for SoES
  - Clear mapping between agents and physical entities/concepts
  - Simple and well defined information exchange
  - Universal character of agent services
- Example of application: software testing
  - Usually done in multidimensional space and thus computationally difficult
  - Agents can help by autonomously performing well defined and simple actions and tasks to effectively cope with the system dimensionality

Agent Based Systems (ABS)

- ABS achievements
  - Rich mathematical foundations
  - Can be built on legacy data/code and specialized data structures
  - Dynamic (can couple arbitrary actions)
  - Open (can interact with other agent platforms)
  - Security (can make other applications more secure)
  - Intelligence
    - Collaboration with other agents, creation of plans, reasoning about time and uncertainty, decision-making, etc.
  - Heterogeneous information integration
  - Rapid creation/deployment

Agent Based Systems (ABS)

- Examples of agents
  - Meta agent programs
    - Reasoning about other agents states and future actions
  - Temporal agent programs
    - Commitments over time (may reason about the past)
  - Probabilistic agent programs
    - Decisions in the presence of uncertainty
  - Secure agent programs
    - Information exchange between secured parties



Conclusions

- Reliability and flexibility properties of DDD and ABS
  - DDD oriented to design with emphasis on hierarchical representations
  - ABS oriented to model performance with emphasis on interactions and behaviors
- Benefits
  - DDD posses some redundancy that makes it robust in case of requirements conflict or uncertainty/incomplete information
  - ABS handles uncertainty as incomplete information (probabilistic reasoning, system of beliefs)
  - Suitable for upgrading through automatic prototyping (DDD) and rewriting of agents (ABS) without changing the overall system
- All these properties make DDD and ABS well suited for SoES design and analysis