Composition at Scale

Janos Sztipanovits
18 August, 2010

Software at Scale
UC Berkeley
Berkeley, CA
Outline

- Integration Challenge in Software Intensive Systems
- Pursuing Compositionality
- Model-Based System Integration
  - Approach
  - Model-Based Integration Process
  - Software/Systems “Wind Tunnel”
  - Challenges
Dimensions of System Integration

- Components
- Layers
- System of Systems
Component Integration

**Functional: E.g.: Dynamics**

- Composability and compositionality are key concepts
- Defined for carefully selected properties (stability, latency, power,..)
- Decomposed into structure, interaction and behavior

**Software: E.g. Timing**

- Challenges:
  - composition frameworks providing constructivity for essential properties
# Layered Systems: Vertical Integration

## Roles
- Cognitive processes
- Social interaction
- Command and control

## Coordination
- Data distribution

## Component interactions
- Component behaviors
- Architecture

## Resource management
- Scheduling
- Separation

## Timing/performance
- Fault management
- Power management

## Heat dissipation
- Crossover
- Radiation effects

## Layers
- **Human Organization**
- **System Operation Layer**
- **SW/Component Layer**
- **OS/Network Layer**
- **HW/Systems Layer**
- **Materials & Devices**

## Characteristics
- Inter-layer interactions
- Effects propagate across the layers
- Efficiency and optimization drives toward intractability
- Inter-layer relationship: mapping, refinement, synthesis
- **Challenges:** modeling, constraining, composing
System of System Integration

Future Combat System

- Heterogeneous
- Open Dynamic Architecture
  - heterogeneous networking
  - heterogeneous components
- Very high level concurrency with complex interactions

Challenges:
  - understanding and predicting behavior
Outline

- Integration Challenge in Software Intensive Systems
- Pursuing Compositionality
- Model-Based System Integration
  - Approach
  - Model-Based Integration Process
  - Software/Systems “Wind Tunnel”
  - Challenges
Integration Inside Abstraction Layers: Composition

**Plant Dynamics Models**  \( B(t) = \kappa_p(B_1(t),...,B_j(t)) \)
- **Properties**: stability, safety, performance
- **Abstractions**: continuous time, functions, signals, flows,…

**Controller Models**

**Physical design**

**Software Architecture Models**  \( B(i) = \kappa_c(B_1(i),...,B_k(i)) \)
- **Properties**: deadlock, invariants, security,…
- **Abstractions**: logical-time, concurrency, atomicity, ideal communication,…

**Software Component Code**

**Software design**

**System Architecture Models**

**Resource Management Models**  \( B(t_j) = \kappa_p(B_1(t_i),...,B_k(t_i)) \)
- **Properties**: timing, power, security, fault tolerance
- **Abstractions**: discrete-time, delays, resources, scheduling,
Controller dynamics is developed without considering implementation uncertainties (e.g. word length, clock accuracy) optimizing performance.

**Assumption:** Effects of digital implementation can be neglected

Software architecture models are developed without explicitly considering systems platform characteristics, even though key behavioral properties depend on it.

**Assumption:** Effects of platform properties can be neglected

System-level architecture defines implementation platform configuration. Scheduling, network uncertainties, etc. are introduce time variant delays that may require re-verification of key properties on all levels.
Challenge to Compositionality: Heterogeneity

• Consequence of the lack of composability across system layers
  – intractable interactions
  – unpredictable system level behavior
  – full-system verification does not scale

• Active research: simplification strategies
  – Decoupling: Use design concepts that decouple systems layers for selected properties
  – Cross-layer Abstractions: Develop methods that can handle effects of cross-layer interactions
Decoupling Example 1: Robust Implementation of R-T Systems

Abdellatif, Combaz, Sifakis [2010]: Model-Based Implementation of Real-Time Applications

- $M$: Based on Logical Execution Time (LET) Based on Timed Automata Actions are atomic and timeless They can be executed after release time and before the due time
- $M_{\phi}$: real-time system. Models the behavior of the software on a platform. Actions are assigned with WCET
Decoupling Example 1: Robust Implementation of R-T Systems

Abdellatif, Combaz, Sifakis [2010]: Model-Based Implementation of Real-Time Applications

However, essential system properties such as stability, safety, performance are expressed in terms of physical behavior.

- $M$: Based on Logical Execution Time (LET)
  - Based on Timed Automata
  - Actions are atomic and timeless
  - They can be executed after release time and before the due time

- $M_r$: real-time system. Models the behavior of the software on a platform. Actions are assigned with WCET
Decoupling Example 2: Passive Dynamics

**Goals:**

- Compositional verification of essential dynamic properties
  - stability
  - safety
- Passivity guarantees stability independently from implementation induced uncertainties
  - time varying delays
  - network uncertainties (packet drops, delays)
- Decreased verification complexity

---

Kottenstette, N., J. Hall, X. Koutsoukos, P. J. Antsaklis, and J. Sztipanovits, "Digital Control of Multiple Discrete Passive Plants Over Networks", *Int. J. of Systems, Control and Communications* 2010
Illustration of Passive Dynamics

Experimental Setup

Joint Angle and Reference

- Two CrustCrawler robotic arms
  - 4 DOF with AX-12 smart servos at each joint
- Novint haptic paddle
- Five networked Windows PCs with Matlab/Simulink

Time delay (Robot 2 and PJ)
Open Research Problems

- Extend theory for decoupling
- Develop theory of compositionality among system layers (vertical composition)
- Extend compositionality for multiple properties, e.g. stability, safety and invariants
- Exploit compositionality in software synthesis
Outline

• Integration Challenge in Software Intensive Systems
• Pursuing Compositionality
• Model-Based System Integration
  – Approach
  – Model-Based Integration Process
  – Software/Systems “Wind Tunnel”
  – Challenges
System of System Integration

Future Combat System

- Heterogeneous
- Open Dynamic Architecture
  - heterogeneous networking
  - heterogeneous components
- Very high level concurrency with complex interactions

**Challenges:**
- understanding and
- predicting behavior

How to achieve predictability with limited/partial composability?
Real-Life SoS Development

• All integration categories are present (component, layer, SoS)

• Systems are evolving along “spiral-outs”

• New technical challenges are emerging and potential solutions need to be rapidly explored

• All layers of the system are subject to modifications, there are no well defined synchronization points in the development process

• Integration is inherently incremental; deployed systems need to be integrated with components on different level of maturity: prototypical and with simulated systems/components.
How Is It Solved Today?

- Systems are integrated when all components are delivered
  - Acquisition pushes in this direction
- Integration means: “Make it working somehow”
- System Integration Labs do not offer support for spiral development
- There is no approach to deal with incomplete specifications and components

System Integration is the highest risk, most expensive, least predictable step in SoS development
Emerging Solution: Model-Based Integration

- Apply Models Early
- Apply Models Often
- Use Every Opportunity
  - Requirements/Architecture Integration
  - Architecture/Design Integration
  - Design Assessment/Verification
  - Prototyping/Scaling
  - Implementation
  - Scaling
  - Testing
Tool Chain for Architecture Exploration in FCS

- ADeVS, IONS
- Simulation
- Lab Integration Segment
- Certification & Accreditation
- RELEX
- GReAT Transform
- System Integration
- System Model Segment
- Document Integration Segment
- Excel
- GME Architecture Model
- XML
- Fault Trees
- BON Extraction
- XLS
- Interface Attributes
- Message Content
- Doc Xpress
- IIR
- RTF
- Rose CAT file
- Component Adapters
- System Integration Test Harness
- Integrated Model
- UML
Risk Mitigation: Surrogate Modeling and Synthesis

Deployment
Instance Topology
Networks

Interfaces

Code Generator

“Real” BC Component

Input Interfaces
Acquired Business Logic
Output Interfaces

System Of Systems Common Operating Environment

Input Interfaces
Business Logic (Generated)
Output Interfaces

“Real” BC Component
Outline

• Integration Challenge in Software Intensive Systems
• Pursuing Compositionality
• Model-Based System Integration
  – Approach
  – Model-Based Integration Process
  – **Software/Systems “Wind Tunnel”**
  – Challenges
How can we integrate the models?
How can we integrate the simulated heterogeneous system components?
How can we integrate the simulation engines?
Model-based Integration Architecture

experiment specification & configuration

"Virtual" components

model integration layer

controller models
network models
org. models
fusion models
env. models

run-time

models

simulink federate
omnet++ federate
 cpn federate
 devs federate
 ogre federate

instrumentation layer

simulation integration platform (HLA)

simulation data distribution/communication middleware

distributed simulation platform

https://wiki.isis.vanderbilt.edu/OpenC2WT
Experiments: Impact of Cyber Attacks

- **Network attack:**
  - A sub-network with hundreds of zombie nodes attacks a critical router on the main network.
  - Flood attack on udp, tcp or ping
Outline

- Integration Challenge in Software Intensive Systems
- Pursuing Compositionality
- Model-Based System Integration
  - Approach
  - Model-Based Integration Process
  - Software/Systems “Wind Tunnel”
- Challenges
### Open Research Problems: Modeling at Scale

- **Model Versioning**
  - Granularity
  - Semantic conflict detection with DSL extensibility
- **Coordinating model versioning**
  - with evolving modeling languages, tools, and tool integration frameworks
  - with other development artifacts
- **Collaborative distributed modeling**
  - Conflict visualization and collaborative resolution
  - Merge consistency guarantees
System Integration is a grand challenge of engineering large-scale systems

Composition/compositionality have practical and theoretical limits in complex heterogeneous systems

Model-based methods provide promising solutions for hard problems