Component-Based Modeling and Simulation (M&S): Status and Perspectives

Axel Lehmann

Institut für Technik Intelligenter Systeme
at the
Universität der Bundeswehr München
Germany

Institut für Technische Informatik
and ITIS e.V.
Universität der Bundeswehr München
Germany

Email: lehman@informatik.unibw-muenchen.de
URL: http://inf4-www.informatik.unibw-muenchen.de/
Introduction: „Airport Traffic“

- Planning analysis goals e.g.:
  - Efficiency of public traffic control?
  - Cost-benefit relation for additional gates?
  - Safety issues in case of accidents?
Major Challenges

- **„Mastering“ of complex systems over lifetime**
  w.r.t. multiple aspects / goals!
  requires:
  - Goal-oriented, efficient effective and dependable modeling!
  but:
  - Increasing system complexity ↔ increasing modeling efforts &
    complexity

- **Strategic modeling approach:**
  - „Divide and conquer“
  - Model design, implementation and application as a MODEL ENGINEERING approach
Major Challenges for M&S:

- Increasing systems / M&S complexity
- Decreasing cycle times for systems / M&S innovations
- Increasing lifetimes of systems: → models, and simulations
- Increasing variety of M&S-aspects / purposes
- Safety, reliability, ... cost-benefit constraints (for systems / M&S)
- User acceptance; ease of use & credibility
- Integration of Virtual / Augmented Reality (VR, AR) etc.
General Approaches to Cope these M&S-Challenges:

- Hierarchical modeling
- Hybrid modeling (e.g. analytic & simulative solution techniques)
- Reuseability of models, or “model components”, requiring
  - adaptability (→ standards?!) → model documentation (→ syntax, semantic)
- Interoperability of models
- Collaborative M&S
- Distributed M&S
- Model credibility analyses
  (→ model verification, validation, accreditation; VV&A)
- Model engineering

⇒ Conclusion:
  M&S as an ENGINEERING PROCESS based on reusable, interoperable model components!!
Component-Based M&S: Status and Perspectives

Outline

Introduction
1. M&S-Design, -Implementation and -Application:
2. Component-Based M&S: Vision versus Reality
3. Formal & Technical Approaches
4. Component Architectures and Technologies
Outlook
M&S - Source of Knowledge and Expertise

⇒ A Multiple-Phase M&S Development Process

- Problem Definition
- System Analysis
- Model Formalization
- Implementation
- Experimentation
M&S – Development Team

- Problem Definition
- System Analysis
- Model Formalization
- Implementation
- Experimentation

- User Knowledge
- Domain Knowledge
- Modeling Expertise
- Experimental Design and Analysis
- (HW-)SW-Expertise

- Customer
- Domain Expert
- User
- Modeller
- Project Manager (Contractor)
- Programmer
Phases & Products in the M&S-Development Process

Input:
- Examination Aim
- System Observations
- Conceptual req. & constr.
- Modeling Method
- Formal req. & constr.
- Solution Techniques
- Technical req. & constr.
- Model Input Data
- Experimental req. & constr.

Phase:
- Problem Definition
- System Analysis
- Model Formalization
- Implementation
- Experimentation

Intermediate Products:
- Structured Problem Description
- Communicative Conceptual Model
- Formal Model
- Executable Model
- Model Results
Example: Effectiveness and efficiency of a “Booking System”

(Customers who access their bank accounts to transfer or to deposit money)

- e.g. client-server architecture
Structured Problem Description

- Goal parameters e.g.:  
  - processing time per transaction  
  - client/server utilization  
  - queueing time, queue length

- Constraints, e.g.:  
  - fidelity of M&S results  
  - budget and deadline of the project  
  - queueing
Example: „Booking System“

- **Conceptual Model:**

```
+--------------------------+
| I/O device 1             |
| I/O device n             |
+--------------------------+
| CPU                      |
| T 1                      |
| .                        |
| .                        |
| T n                      |
+--------------------------+
```

Clients           Server
Example: „Booking System“

- Formal Model:
Example: „Booking System“

- **Executable model:** Performance measures
  - Little’s law:
    \[ \bar{k} = \lambda + t \]
    
    with: response time \( t \), queueing time \( w \), service rate \( \mu \)
    
    \[ t = w + \frac{1}{\mu} \]
  - State probability \( p(k) \):
    \[ \bar{k} = \sum k \cdot p(k) \]
  - Utilization \( \rho \) (\( m \) service stations):
    \[ \rho = \frac{\text{arrival rate}}{\text{service rate}} = \frac{\lambda}{m \mu} < 1 \]
Model “Component”-Based Approach:

(e.g. regarding the example „Booking System“)

- **Problem Description:**
  → pragmatism (goal specification & constraints)

- **Conceptual Model:**
  → structural & functional description of “components”
  → different levels of abstraction

- **Formal Model:**
  → formal specification of “components”
  (↔ selected modeling paradigm(s))
  → hierarchical modeling approach
  (↔ decomposition into submodels/ “components”)

- **Executable Model(s):**
  e.g. → analytic solution
  → simulation { reusable SW- “components“}
2. Component-Based Modeling: Vision versus Reality

What is a „Component“?

- Douglass (2000): “... it’s whatever you want it to be”
- Szyperski (1997): “Components are for composition”
- Griffel (1998): “... they are having an autonomous functionality” (Translation)
- Meyer (2000): “... may be used by other software elements (clients) ... without the intervention of the components developers.”
- Meyer (2000): “Software is a meta product: .....consisting of:

Each M&S-component has to be specified by a tripel:

\[ M&S-C := (pragmatic, syntax, semantic) \]

\[ \text{Specification} \]
\[ \text{plan, parameterized} \]
\[ \text{instance(s)}...\] (Translation)
Properties of a (Software) Component:

- "Includes a specification of all dependencies; (HW-, SW-, ect.)"
- includes a precise specification of offered functionalities;
- is usable on the sole basis of that specification;
- is composable with other components;
- can be integrated into a system quickly and smoothly. “

(Meyer, 2000)

component-class /-type  
-------------
component-objects /-instances  

(description of structure & implementation)  
(implementation)
„Components“ versus Objects and Modules

(according to Szyperski, Mayer)

- According to references in literature: **Very vague differences!!**

- **Objects:**
  - have a status which can be persistent
  - have import / export interfaces (data, methods)
  - can be part of a component

- **Modules:**
  - can be seen as “minimal components”
  - can include abstract data types & object classes
  - have no resources
  - connections to other modules are known
    (fixed structural arrangements)
  - can contain classes (e.g. Oberon, Modula-3)
Component-Based M&S : Current Approaches

Already applied:

- Hierarchical modeling via
  - decomposition
  - aggregation
  - hybrid solution / implementation techniques

- Generic modeling object templates (depending on the modeling paradigm), e.g.
  - class / object libraries

- Function / Program libraries, e.g.
  - statistical analyses
  - random number generators

- Coupling of monolithic models, e.g.
  - federation of models (DIS, HLA, . . .)
  - agent-based simulation

Conclusion: Missing comprehensive formal & methodological approaches for component-based M&S
Component-Based M&S: Perspectives

Input:
- Examination Aim
- System Observations
- Conceptual req.&constr.
- Modeling Method
- Formal req.&constr.
- Solution Techniques
- Technical req.&constr.
- Model Input Data
- Experimental req.&constr.

Phases:
- Problem Definition
- System Analysis
- Model Formalization
- Implementation
- Experimentation

Products:
- Structured Problem Description
- Communicative Conceptual Model
- Formal Model
- Executable Model
- Model Results

M&S- „components“:
Model („Component“) Specification Levels

- Model Federation Level:
  "Black Boxes"

- Model Level:
  Autonomous, interoperable models

- Submodel/Object Level:
  Submodels/Object structures

- Function Level:
  Coded basic functions/algorithms
Model Development & V&V

- Sponsor Needs
- Structured Problem Description
- Conceptual Model
- Model Development
- Model Requirements Validation
  - P(E_P)
- Static Model Structure Verification & Validation
  - P(E_A)
- Formal Model
- Static Code & HW Verification & Validation
  - P(E_F)
- Executable Model
- Dynamic Simulation Model Behavior Verification & Validation
  - P(E_I)
- Model Results
  - P(E_E)

Ok, we have got the following problem...
3. Formal and Technical Approaches

(→ Formalisms for Component-Based M&S Specifications)

- **DEVS (Discrete Event System Specification, B. Zeigler)**

  Atomic DEVS : = (X, S, Y, in, ex, au, v)

  - \(X: \) set of input ports & input events set
  - \(S: \) sequential states set
  - \(Y: \) set of output ports & output events set
  - \(\text{in}: = S \rightarrow S: \) internal state transition function
  - \(\text{ex}: = Q \times X: \) external transition function
  - \(\text{au}: = S \rightarrow Y: \) output function
  - \(\nu: = S \rightarrow R^+: \) time advanced function

- **Coupled Models ↔ hierarchical modeling approach**
UML-Based M&S

- UML: "standard" notation for analysis, design and implementation of 00-systems:
- UML-meta model $\leftrightarrow$ specifies abstract semantics
- UML-notation $\leftrightarrow$ describes a set of diagrams ($\rightarrow$ syntax):
  - Static diagrams: Use Cases, Class / Object Diagrams
  - Behavioural diagrams: Activity $\sim$, State $\sim$
  - Interaction diagrams: Sequence $\sim$, Collaboration $\sim$
  - Implementation diagrams: Component $\sim$, Deployment $\sim$

$\rightarrow$ Extensions, e.g. for quantitative software performance M&S
VHDL: Hierarchical & Component-Based Hardware / Microelectronic Circuit Design

- Gajski-/Y-Diagram
Components and ist instances in VHDL

ARCHITECTURE behavior
out1 = f1(in1,..., inn)
...
outm = fm(in1,..., inn)

ARCHITECTURE structure
ENTITY subkomp1
ENTITY subkomp2
ENTITY subkomp3
ARCH. structure1
ARCH. structure2
4. Components Architectures and Technologies

- **Architectural Approaches**
  - OMA (Object Management Architecture; basis for CCM)
  - DIS (Distributed Interactive Simulation)
  - HLA (High Level Architecture)
  - Agent-based Simulation

- **(Software) Component Technologies**
  - Component Object Model (Microsoft: COM, COM+, DCOM, .Net, ActiveX)
  - Java Beans, Model Beans, EJB (Sun Microsystems)
  - CORBA Component Model (CCM; OMG)
  - Web Services (SOAP; WSDL; UDDI)
Outlook

Summary

“Mastering” of systems / M&S complexity over lifetime requires:

- **Hierarchical M&S** e.g. by
  - decomposition
  - aggregation

- **Reusability & interoperability** of M&S-components via:
  - a semiotic approach:
    M&S-component: = (pragmatism, semantic, syntax)

- M&S development & maintenance: is an engineering discipline !!
Component architectures are developed
Component technologies are available

but: M&S-component definition requires specification of:
   (pragmatism (intended purpose),
    semantic (e.g. the plan),
    syntax (an implementation))

Long Term Perspectives:
   Repositories for: pragmatic concepts,
                    formal model components (algorithms),
                    executable model components (code)
   Increasing productivity & credibility (→ VV&A of M&S-
                                 components)
THANK YOU
FOR YOUR INTEREST & ATTENTION!

Please find further details under:
http://inf4-www.informatik.unibw-muenchen.de/
High Level Architecture (HLA)

- Federation „A“
  - Federate $A_1$
    - HLA Interface
  - Federate $A_n$
    - HLA Interface
- Federation Execution (FedEx) „A“

**HLA Runtime Infrastructure (RTIExec)**: FedEx-Management, Naming Service etc.

Operating System Level

(RTI of the Defense Modeling & Simulation Office)