Specification of Complex Systems
An Overview

Laure Petrucci

LIPN, CNRS UMR 7030
99, avenue Jean-Baptiste Clément
93430 Villetaneuse
France

ICTAC’2006
Motivation

Problem

- tackling more and more complex systems
- critical behaviour (avionics, medicine . . .)

- necessary to obtain reliable systems
- verify behaviour before actual implementation/use of the system
Motivation

Problem
- tackling more and more complex systems
- critical behaviour (avionics, medicine ...)

- necessary to obtain reliable systems
- verify behaviour before actual implementation/use of the system
Motivation

Traditional approach

- Physical **test beds** (e.g. models)
- **Simulation** traces

Drawbacks

- material infrastructure
  - difficult setup
  - costly
  - requires highly qualified personnel
- too many **details** in the traces (fine grain)
- traces difficult to interpret
Motivation

**Traditional approach**
- Physical test beds (e.g. models)
- Simulation traces

**Drawbacks**
- material infrastructure
  - difficult setup
  - costly
  - requires highly qualified personnel
- too many details in the traces (fine grain)
- traces difficult to interpret
Motivation

Approach advocated

- logical model to abstract from physical system
- choose an adequate level of abstraction

Advantages of using a model

- low cost as not physical
- analysed using software tools ⇒ easy to modify without extra cost
- clearer and rigorous view of the system to be designed
- prototype development after verification, with a high level of confidence
- provides a clear description facilitating maintenance
Motivation

Approach advocated

- logical model to abstract from physical system
- choose an adequate level of abstraction

Advantages of using a model

- low cost as not physical
- analysed using software tools $\Rightarrow$ easy to modify without extra cost
- clearer and rigorous view of the system to be designed
- prototype development after verification, with a high level of confidence
- provides a clear description facilitating maintenance

Laure Petrucci

20 November 2006

4 / 27
Outline

1. Semi-formal models

2. Formal models
   - Automata
   - Petri nets
   - After specification, verification

3. Relevant criteria for specification
   - Relevant Concepts
   - Level of abstraction
   - Structuring
   - Properties
   - Further development of the model
   - Tool support
1 Semi-formal models

2 Formal models
   - Automata
   - Petri nets
   - After specification, verification

3 Relevant criteria for specification
   - Relevant Concepts
   - Level of abstraction
   - Structuring
   - Properties
   - Further development of the model
   - Tool support
Main approach

**UML**: Unified Modelling Language
- design steps
- functionning of the system captured in a collection of diagrams
- graphical presentation providing an easy understanding
Semi-formal models

Pros and Cons

Aims at (pros)

- a **better understanding** of the system to design and the **interaction** between components
- eventually **reconsidering the conceptual choices**
- **facilitating** the communication with clients, by providing an **easy to understand notation**
- **providing a precise documentation** for later **maintenance**

Cons

- difficult to handle the collection of diagrams as a whole
- no exhaustive analysis
Pros and Cons

Aims at (pros)

- a **better understanding** of the system to design and the **interaction** between components
- eventually **reconsidering** the **conceptual choices**
- **facilitating** the communication with clients, by providing an **easy to understand notation**
- providing a **precise documentation** for later **maintenance**

Cons

- **difficult** to handle the collection of diagrams as a whole
- **no exhaustive** analysis
Semi-formal models

Formal models
- Automata
- Petri nets
- After specification, verification

Relevant criteria for specification
- Relevant Concepts
- Level of abstraction
- Structuring
- Properties
- Further development of the model
- Tool support
Formal Models

Aims

- prove formally (i.e. mathematically) that the system has the appropriate behaviour
- whatever the execution, behaves as expected

Analysis techniques

- simulation ⇒ have sufficient confidence in the behaviour of the system. Find initial errors.
- exhaustive analysis of expected properties
Automata

Different sorts of automata

- communicating
- timed
- with variables
- ...

Automaton

- represented as a graph
- States : nodes
- Transitions : arcs, indicate how to move from one state to the next
- the choice of the transition to fire is non-deterministic
Automata

Different sorts of automata

- communicating
- timed
- with variables
- ...

Automaton

- represented as a graph
- States: nodes
- Transitions: arcs, indicate how to move from one state to the next
- the choice of the transition to fire is non-deterministic
Automata

Different sorts of automata
- communicating
- timed
- with variables
- ...

Automaton
- represented as a graph
- **States**: nodes
- **Transitions**: arcs, indicate how to move from one state to the next
- the choice of the transition to fire is non-deterministic
Automata

Different sorts of automata
- communicating
- timed
- with variables
- ...

Automaton
- represented as a graph
- States : nodes
- Transitions : arcs, indicate how to move from one state to the next
- the choice of the transition to fire is non-deterministic
Example
Petri Nets

- view of both **states** and **transitions**
  - **Places**: nodes represented as circles or ellipses
    - model part of the system state
    - contain tokens which indicate the number of occurrences of this substate
  - **Transitions**: nodes represented as rectangles
    - model events that can take place
    - input arcs: pre-conditions for firing the transition
    - output arcs: post-conditions to firing the transition (i.e. result)
Petri Nets

- view of both states and transitions
- **Places**: nodes represented as circles or ellipses
  - model part of the system state
  - contain tokens which indicate the number of occurrences of this substate
- **Transitions**: nodes represented as rectangles
  - model events that can take place
  - input arcs: pre-conditions for firing the transition
  - output arcs: post-conditions to firing the transition (i.e. result)
Petri Nets

- view of both **states** and **transitions**
- **Places**: nodes represented as **circles** or **ellipses**
  - model part of the system state
  - contain **tokens** which indicate the **number of occurrences** of this substate
- **Transitions**: nodes represented as **rectangles**
  - model events that can take place
  - input arcs: pre-conditions for firing the transition
  - output arcs: post-conditions to firing the transition (i.e. result)
Petri Nets

- view of both states and transitions
- **Places**: nodes represented as circles or ellipses
  - model part of the system state
  - contain tokens which indicate the number of occurrences of this substate
- **Transitions**: nodes represented as rectangles
  - model events that can take place
  - input arcs: pre-conditions for firing the transition
  - output arcs: post-conditions to firing the transition (i.e. result)
Example

Laure Petrucci
High-level Petri Nets

- **flow**: Petri net structure
- **data**: data specification language
  - data values are attached to **tokens**.
- **arcs inscriptions**: terms which test/modify token values
**High-level Petri Nets**

- flow: Petri net structure
- data: data specification language
- data values are attached to tokens.
- arcs inscriptions: terms which test/modify token values
High-level Petri Nets

- flow: Petri net structure
- data: data specification language
- data values are attached to tokens.
- arcs inscriptions: terms which test/modify token values
Example

Formal models

Petri nets

Example

Laure Petrucci 20 November 2006 16/27
After Specification, Verification

Simulation can be
- **fully automatic**: choice between transitions made by the tool
- manual or guided: choice made by the user

Stop criteria
- number of firing steps for each simulation
- property: simulation stops when the property is not satisfied \(\Rightarrow\) trace leading to malfunction
After Specification, Verification

Simulation can be
- **fully automatic**: choice between transitions made by the tool
- **manual** or **guided**: choice made by the user

Stop criteria
- number of firing steps for each simulation
- property: simulation stops when the property is not satisfied ⇒ trace leading to malfunction
After Specification, Verification

Simulation can be

- fully automatic: choice between transitions made by the tool
- manual or guided: choice made by the user

Stop criteria

- number of firing steps for each simulation
- property: simulation stops when the property is not satisfied $\Rightarrow$ trace leading to malfunction
### Simulation can be
- **fully automatic**: choice between transitions made by the tool
- **manual or guided**: choice made by the user

### Stop criteria
- **number of firing steps** for each simulation
- **property**: simulation stops when the property is not satisfied ⇒ **trace** leading to malfunction
## Properties

- **analysis**
  - **structural**: independent of the initial state
  - **behavioural**: representation (might be exhaustive) of the system behaviour

- properties written using an appropriate formalism
- **standard properties** (of Petri nets): boundedness, deadlocks, liveness, home states . . .

- **temporal logics**:
  - LTL (Linear Time Logic): properties on execution paths
  - CTL (Computation Tree Logic): properties of reachable states
Verification

Properties

- **analysis**
  - **structural**: independent of the initial state
  - **behavioural**: representation (might be exhaustive) of the system behaviour

- properties written using an **appropriate formalism**
  - **standard properties** (of Petri nets): boundedness, deadlocks, liveness, home states . . .
  - **temporal logics**:
    - LTL (Linear Time Logic): properties on execution paths
    - CTL (Computation Tree Logic): properties of reachable states
Verification

### Properties

- **analysis**
  - **structural**: independent of the initial state
  - **behavioural**: representation (might be exhaustive) of the system behaviour

- properties written using an **appropriate formalism**

- **standard properties** (of Petri nets): boundedness, deadlocks, liveness, home states . . .

- **temporal logics**:
  - LTL (Linear Time Logic): properties on execution paths
  - CTL (Computation Tree Logic): properties of reachable states
1 Semi-formal models

2 Formal models
   • Automata
   • Petri nets
   • After specification, verification

3 Relevant criteria for specification
   • Relevant Concepts
   • Level of abstraction
   • Structuring
   • Properties
   • Further development of the model
   • Tool support
Identify the key concepts used in the system to choose the most appropriate formalism

Data types

- the system may use arbitrarily complex data
- should the data be detailed?
- the higher the description level, the more difficult the analysis
- in case the data description is relevant: rather choose a data oriented formalism, e.g. abstract data types or high-level nets
- if the data description is not relevant: prefer a low-level formalism, e.g. automata, Petri nets
- in an intermediate case: (data can be mapped onto integers ... ) counter automata, symmetric Petri nets
Identify the **key concepts** used in the system to choose the **most appropriate formalism**

**Data types**

- the system may use arbitrarily **complex data**
  - should the data be detailed?
  - the higher the description level, the more difficult the analysis
- in case the **data description is relevant**: rather choose a **data oriented formalism**, e.g. abstract data types or high-level nets
- if the **data description is not relevant**: prefer a **low-level formalism**, e.g. automata, Petri nets
- in an **intermediate case**: (data can be mapped onto integers ... ) counter automata, symmetric Petri nets
Identify the **key concepts** used in the system to choose the **most appropriate formalism**

**Data types**

- the system may use arbitrarily **complex data**
- should the data be detailed?
- the higher the description level, the more difficult the analysis
- in case the **data description is relevant**: rather choose a **data oriented formalism**, e.g. abstract data types or high-level nets
- if the **data description is not relevant**: prefer a **low-level formalism**, e.g. automata, Petri nets
- in an **intermediate case**: (data can be mapped onto integers ... ) counter automata, symmetric Petri nets
Relevant Concepts

Identify the **key concepts** used in the system to choose the **most appropriate formalism**

**Data types**

- the system may use arbitrarily **complex data**
- should the data be detailed?
- the higher the description level, the more difficult the analysis
- in case the **data description is relevant**: rather choose a **data oriented formalism**, e.g. abstract data types or high-level nets
- if the **data description is not relevant**: prefer a **low-level formalism**, e.g. automata, Petri nets
- in an **intermediate case**: (data can be mapped onto integers ...) counter automata, symmetric Petri nets
Relevant Concepts

Time

- **is time necessary? which sort of time?**
  - waiting time
  - execution time
- **formalisms capturing time**: automata, time(d) Petri nets …
- **different mechanisms**:
  - global clock: reference for all transitions
  - local clock for each system component
  - time intervals: firing can take place in a time interval since the transition is enabled
- **discrete time vs. continuous time**
## Relevant Concepts

### Time

- **is time necessary? which sort of time?**
- **waiting time**
- **execution time**

- **formalisms capturing time**: automata, time(d) Petri nets ...

- **different mechanisms**:
  - **global clock**: reference for all transitions
  - **local clock** for each system component
  - **time intervals**: firing can take place in a time interval since the transition is enabled

- **discrete time vs. continuous time**
Relevant Concepts

Time

- is time necessary? which sort of time?
- waiting time
- execution time

- formalisms capturing time: automata, time(d) Petri nets...

- different mechanisms:
  - global clock: reference for all transitions
  - local clock for each system component
  - time intervals: firing can take place in a time interval since the transition is enabled

- discrete time vs. continuous time
Relevant Concepts

Time

- **is time necessary?** which sort of time?
- waiting time
- execution time
- **formalisms capturing time**: automata, time(d) Petri nets
- **different mechanisms**:
  - **global clock**: reference for all transitions
  - **local clock**: for each system component
  - **time intervals**: firing can take place in a time interval since the transition is enabled

**discrete time vs. continuous time**
Relevant Concepts

**Time**

- is time necessary? which sort of time?
- waiting time
- execution time
- formalisms capturing time: automata, time(d) Petri nets . . .
- different mechanisms:
  - global clock: reference for all transitions
  - local clock for each system component
  - time intervals: firing can take place in a time interval since the transition is enabled
- discrete time vs. continuous time
Relevant Concepts

Communication between subsystems
- synchronising by *rendez-vous*: send and receive occur together
- using a *communication channel*
  - channel *policy*: FIFO, reordering messages . . .
  - with/without *loss*
Level of Abstraction

Principles

- first, model with a **high level of abstraction**
- **validate** the model
- then, **incremental construction using refinement techniques**:
  - increase the complexity of data
  - augment the model structure
  - detail a complex action by splitting it into more elementary ones
Level of Abstraction

Principles

- first, model with a **high level of abstraction**
- **validate** the model
- then, **incremental construction** using **refinement techniques**:
  - increase the **complexity of data**
  - augment the model **structure**
  - **detail** a complex action by splitting it into more elementary ones
Structuring

Specification of subsystems and their interactions

- focus on a particular subsystem
- try out different subsystems
- reuse a subsystem in another model
- integrate several instances of a subsystem within a parameterised model
- use on-the-shelf components
Properties

Write properties

- with a formalism compatible with the system specification
- insuring that the key elements in the property are also part of the model, e.g. were not discarded when considering the earlier points
Future enhancements of the model or properties should be foreseen ⇒ justify the choices in the long run.
Tool Support

Tools provide

- model editing
- analysis techniques

for different types of models or properties
Tool Support

Tools provide

- model editing
- analysis techniques
- for different types of models or properties