

# A Case Study in Parallel Verification of Component-Based Systems

N. Beneš, I. Černá, J. Sochor, P. Vařeková and B. Zimmerova

Faculty of Informatics, Masaryk University  
Brno, Czech Republic

PDMC'08

March 29, 2008

## ① Introduction

Motivation

Contribution

## ② Foundations

CoCoME modelling contest

Modelling language

Logic for properties specification

## ③ Verification

Model of the system

Verification of the model

Overview of the results

## ④ Conclusion

Summary of the talk

## ① Introduction

- Motivation
- Contribution

## ② Foundations

- CoCoME modelling contest
- Modelling language
- Logic for properties specification

## ③ Verification

- Model of the system
- Verification of the model
- Overview of the results

## ④ Conclusion

- Summary of the talk

# Motivation

## **Component-based development** as a current trend in SE

- Systems assembled from third-party components  
→ issue of **correct interaction** among components

## **Parallel verification** becomes a need

- Concurrency of a large number of components  
→ applicability of sequential verification techniques becomes challenging

# Contribution

- 1 Identification of the **component-specific properties** defining correctness issues in component-based systems, and their formalization
  - expressed in the logic CI-LTL – an extended version of the action-based Linear Temporal Logic
  - check the **validity of the model** and the **correctness of the system**
- 2 Experimental **application of parallel model checking** to the model of a real system
  - CoCoME modelling example
  - parallel model-checking tool DiVinE

## ① Introduction

Motivation

Contribution

## ② Foundations

CoCoME modelling contest

Modelling language

Logic for properties specification

## ③ Verification

Model of the system

Verification of the model

Overview of the results

## ④ Conclusion

Summary of the talk

# CoCoME modelling contest

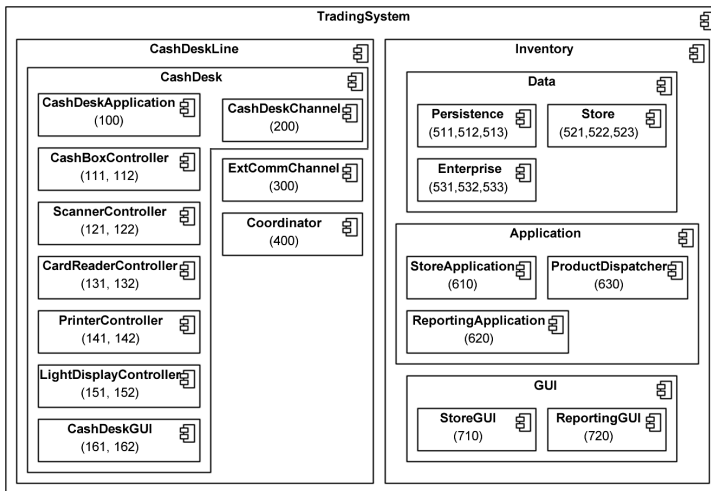
## The contest

- **The aim** to evaluate and compare the **existing component modelling approaches** on a common modelling example
- 18/13 modelling teams, URL <http://www.cocome.org>  
Organized by A. Rausch, R. Reussner, R. Mirandola, F. Plasil

## The Common Component Modelling Example

- A trading system handling sales in a chain of supermarkets
  - interaction with the cashier, inventory, ordering goods, generating reports
- Specification in terms of Java source code (125 classes)
  - to prevent ambiguities in the interpretation of the specification by the teams

# Common Component Modelling Example



## Component-Interaction automata language (or CI automata for short)

- Automata-based language
  - finite state model, infinite executions/traces
- Three types of actions
  - input, output and internal
- Captures important interaction information
  - participants of communication, hierarchy of components
- Flexible composition
  - can be parametrized by architectural assembly, communication strategy
- General to meet various component models
  - by fixing the composition operator and semantics of actions

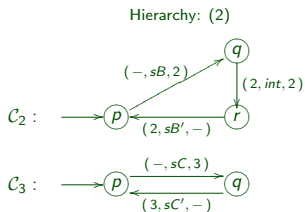
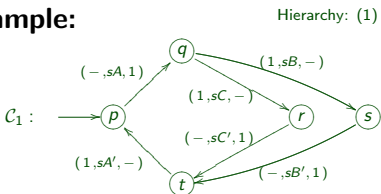
# Modelling language

2/3

## A component-Interaction automaton

- Finite set of states
- Labeled transitions with structured labels  
(component names, actions)
- Hierarchy of component names

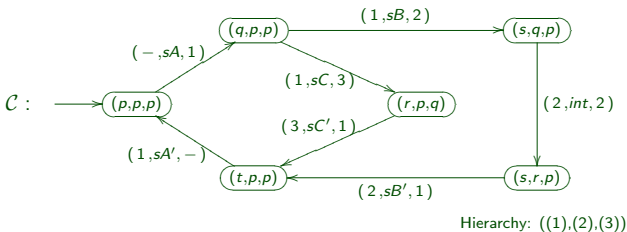
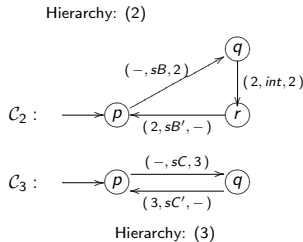
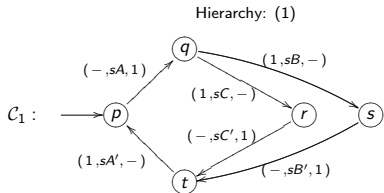
### Example:



# Modelling language

3/3

Composition of CI automata  $\mathcal{C} = \otimes^{\mathcal{F}} \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$



# Logic for properties specification

## CI-LTL

- extended version of the action based LTL
- motivated by CBSE requirements
- expresses properties about both
  - occurring component interaction (i.e. labels in automata)
  - possible component interaction (i.e. label enabledness)

## Operators

- possible interaction (in a state)  $\mathcal{E}(I)$
- actual interaction (on a path)  $\mathcal{P}(I)$
- LTL operators – next  $\mathcal{X}\Phi$ , until  $\Phi \mathcal{U} \Psi$ , and  $\Phi \wedge \Psi$ , not  $\neg\Phi$

**Example**  $\mathcal{G} (\mathcal{E}(1, a, 2) \Rightarrow \mathcal{F} \mathcal{P}(1, a, 2))$

## ① Introduction

Motivation

Contribution

## ② Foundations

CoCoME modelling contest

Modelling language

Logic for properties specification

## ③ Verification

Model of the system

Verification of the model

Overview of the results

## ④ Conclusion

Summary of the talk

# Model of the system

## Structure of the model

- 140 primitive CI automata,
- composed hierarchically – 34 composite automata, 6 levels
- complemented with a usage profile of the Cashier

## State space of the model

- Without usage profile
  - over 322 millions of states (60 GB of memory)
- With usage profile
  - 749 340 reachable states, 3 181 473 reachable transitions

## Correctness of the system

- Local deadlocks of components
  - one of the components cannot move further
- Blocking of components
  - temporary blocking of a component
- Infinite loops in the model
  - finite `for/while` cycles traversed infinitely-many times
- Use-case scenarios
  - presence of a particular scenario (sequence of actions) in the system

## Validity of the model

- checking safety of considered abstractions
- simplification of the communicational scheme
- internal behaviour of primitive components
- checked via a number of test cases translated into CI-LTL

**Property 4 (Local deadlock on one action).** It cannot happen that the *CashDeskApplication* (100) is ready to send a notification to the *CashDeskChannel* (200) saying that it received the *SaleStartedEvent*, but the *CashDeskChannel* is never ready to accept the notification.

$$[\mathcal{FP}(100, oESS'', -)] \vee \mathcal{G}[\mathcal{E}(100, oESS'', -) \Rightarrow \mathcal{FE}(100, oESS'', 200)]$$

property	states	transitions	memory	time	result
prop4	749 343	3 181 479	532 MB	67 s	holds

**Property 7 (Blocking of a component).** It cannot happen that the *CashDeskApplication* (100) is ready to send a notification to the *CashDeskChannel* (200) saying that it received the *SaleStartedEvent*, but the *CashDeskChannel* is not right in the current state ready to accept the notification.

$$[\mathcal{FP}(100, \circ ESS'', -)] \vee \mathcal{G} \neg [\mathcal{E}(100, \circ ESS'', -) \wedge \neg \mathcal{E}(100, \circ ESS'', 200)]$$

property	states	transitions	memory	time	result
prop7	1 498 671	6 362 935	688 MB	534 s	holds not

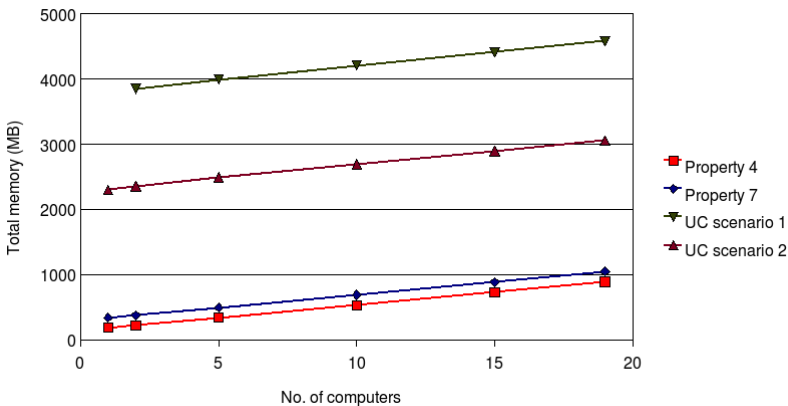
## Use Case scenarios

1. Cash Payment
2. Unsuccessful Card Payment
3. Successful Card Payment

property	states	transitions	memory	time	result
uc1	19 362 460	81 959 821	4 204 MB	5 141 s	found
uc2	11 670 924	49 165 124	2 694 MB	3 203 s	found
uc3	11 680 736	49 202 320	2 698 MB	3 098 s	found

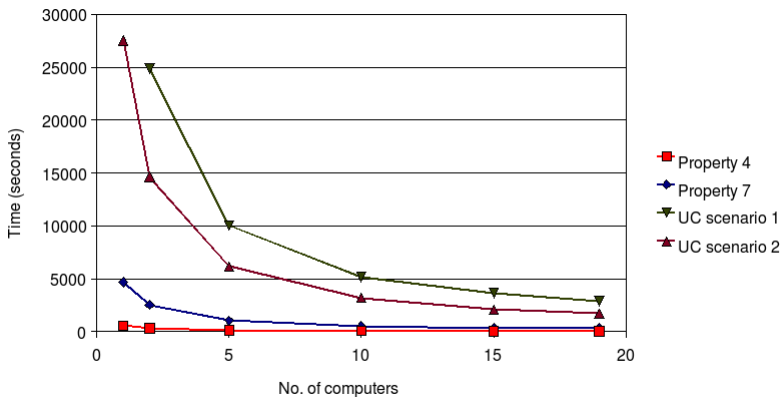
# Effect of the parallelization

## Memory consumption depending on the number of computers



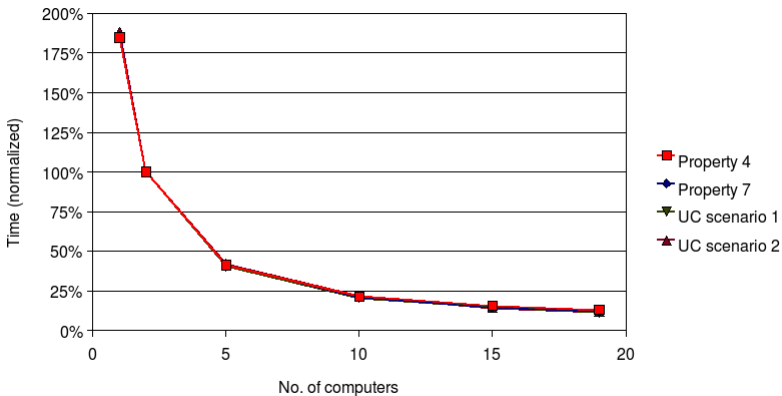
# Effect of the parallelization

**Time consumption** depending on the number of computers



# Effect of the parallelization

**Time consumption** depending on the number of computers



# Lessons learned

## Experience and discussion

- Characteristics of the model
  - state space explosion, irregular changes
- Deadlocks in the model
  - unrealistic behaviour
- Local deadlocks and component-blocking properties
  - enabledness operator
- Parallelization
  - effect of parallelization

## ① Introduction

Motivation

Contribution

## ② Foundations

CoCoME modelling contest

Modelling language

Logic for properties specification

## ③ Verification

Model of the system

Verification of the model

Overview of the results

## ④ Conclusion

Summary of the talk

# Conclusion

## Summary of the talk

- Practical **application of the parallel verification** to a real component-based system
- Identification of a number of typical **component specific properties**
- Formalization of the properties using the logic CI-LTL
- Demonstration of the parallel CI-LTL verification

## Future work

- Reduction techniques
  - the partial-order and the symmetry reduction

# Thank you

**Thank you** for your attention

