

CoCoME in SOFA 2.0



Tomáš Bureš, Martin Děcký, Petr Hnětynka, Jan Kofroň, Pavel Parízek, František Plášil,
Tomáš Poch, Ondřej Šerý, Petr Tůma

Common Component Modeling Example

CoCoME [3] is an international contest aiming at comparison of different component modeling approaches on a common assignment organized by Technische Universität Clausthal, Universität Karlsruhe, Politecnico di Milano, Charles University in Prague. Sharing a common close-to-real-life assignment should reveal strengths and weaknesses of the different modeling approaches and thoroughly test their applicability in practice.

The assignment: TradingSystem

- Business application for managing chain of stores
 - ~ 200 stores, ~ 8 cash desks per store
- Specification via UML + use-cases + reference implementation

SOFA 2.0

SOFA 2.0 [1] is a hierarchical component model developed at Distributed Systems Research Group at Charles University in Prague. It profits from our long-year experience and contains many innovative ideas. The most distinguished features are:

- Advanced component model
 - Supporting hierarchical components, versioning, service oriented architectures, aspects, connectors
- Behavior specification and verification of component
 - Formalism of **Extended Behavior Protocols** (EBP)
- Automated connector generation
- Support for components not only at design-time but also at development- and deployment- and run-time

CoCoME in SOFA 2.0

One of the CoCoME participants was the SOFA team. The following summarizes modeling of the CoCoME assignment using SOFA 2.0.

- Architecture**
 - First, the SOFA architecture of the TradingSystem (Fig. 1) was created, forming basis for further work
- Behavior modeling**
 - All CoCoME components were specified using EBP (available at [2])
 - Parts of the CoCoME assignment are ambiguous and even contradictory → the behavior specification is mainly based on the reference implementation
- Verification (work in progress)**
 - Checking communication of components for errors
 - Bad-activity, no-activity (deadlock), infinite-activity
 - Compliance checking – idea
 - Translation of EBP specification into **Promela**
 - Using the **SPIN** model checker [6] to search the state space
 - Code checking
 - “Does a primitive component obey its behavior specification?”
 - Via combining **Java PathFinder** [4] with Behavior Protocol Checker [5]
- Performance**
 - Look for another poster here...:o)

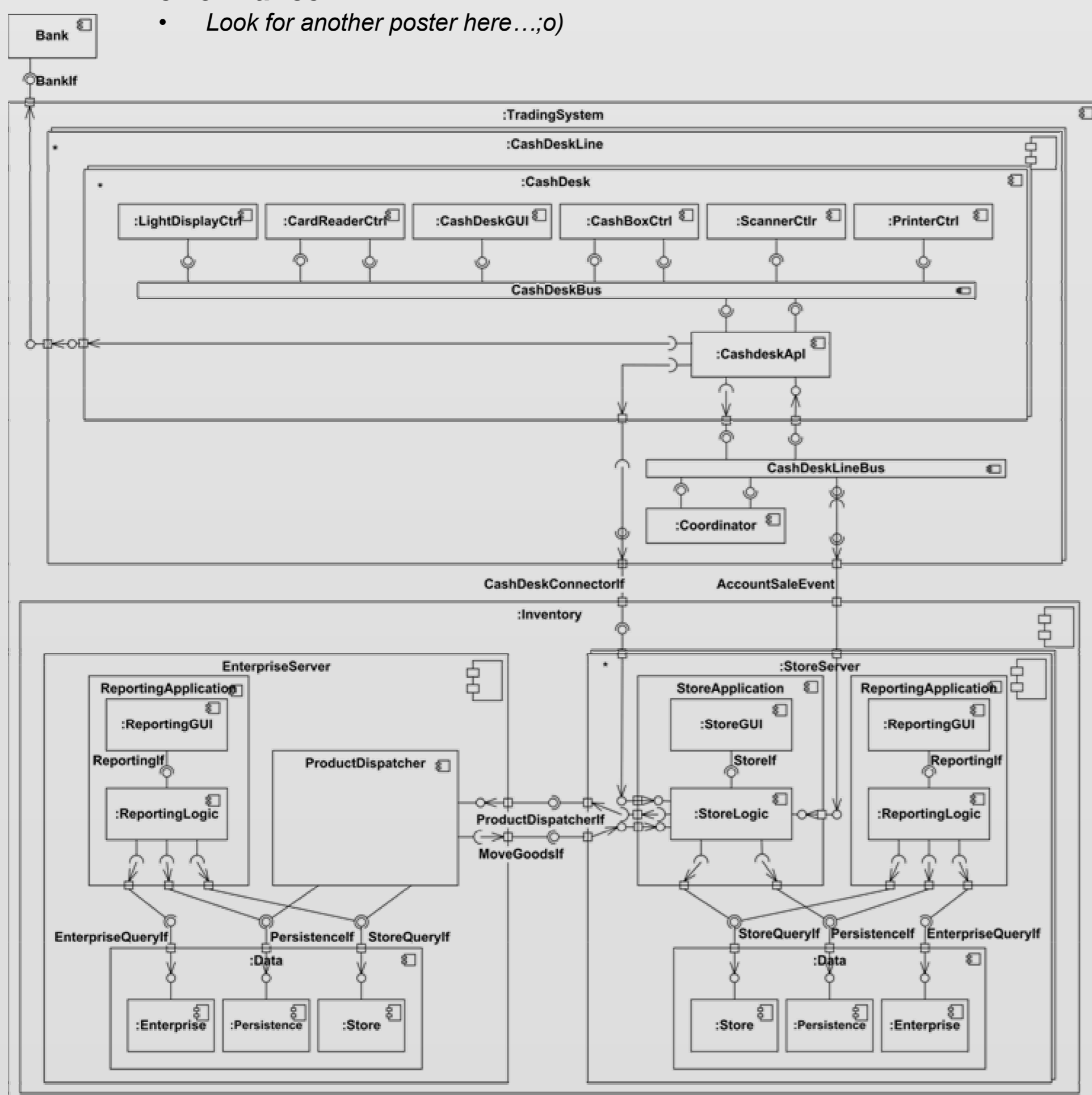


Fig. 1 SOFA architecture of the CoCoME assignment

Benefits

As SOFA 2.0 explicitly supports components throughout the entire software life-cycle, the architecture erosion is mitigated. Moreover automatic connector generation provides seamless component distribution. Behavioral modeling via EBP and subsequent verification allows for reasoning about correctness of the design even before actually having an implementation. Then with the implementation available one, can use code checking to find out, whether the implementation obeys its EBP specification.

What are **Extended Behavior Protocols** (EBP)?

The EBP formalism extends the formalism of **Behavior Protocols** (BP) [5], which aims behavior specification of software components. Similar to BP, the syntax of EBP is regular-expression-like extended by parallel compositions. In addition, the formalism was enhanced with Component local state variables & method parameters, **while** and **switch** statements, and multi-synchronization.

- Basic events:
 - Accepting a method call with formal resp. concrete parameters:


```
?iface.method(type1 arg1, type2 arg2, ...)↑
?iface.method(val1, val2, ...)↑
```
 - Accepting return from a method call: `?iface.method↓`
 - Calling a method: `!iface.method(val1, val2, ...)↑`
 - Returning from method call: `!iface.method↓`
 - Multi-synchronization: `@multisync_event`
 - Assignment: `local_var <- symbolic_value`
- Abbreviations:
 - `?iface.method(params) ≡ ?iface.method(params)↑; !iface.method↓`
 - `!iface.method(params) ≡ !iface.method(params)↑; ?iface.method↓`
 - `?iface.method(params) {expr} ≡ ?iface.method(params)↑; expr; !iface.method↓`
- Operators:
 - alternative '+', sequence ';', repetition '*', and-parallel '|', or-parallel '||'

EBP Example

The **LightDisplay** component is one the CoCoME components—subcomponent of the **CashDesk** component. **LightDisplay** is “enabled” (on) if the associated cash desk is in an express mode and “disabled” (off) otherwise.

```
component LightDisplay {
  types {
    states = {LIGHT_ENABLED, LIGHT_DISABLED}
  }
  vars {
    states state = LIGHT_ENABLED
  }
  behavior {
    ?LightDisplayCtrlEventHandlerIf.onEvent(ExpressModeEnabled) {
      state <- LIGHT_ENABLED
    } * |
    ?LightDisplayCtrlEventHandlerIf.onEvent(ExpressModeDisable) {
      state <- LIGHT_DISABLED
    } *
  }
}
```

References

- Bures, T., Hnetynka, P., Plasil, F.: SOFA 2.0: Balancing Advanced Features in a Hierarchical Component Model, Proc. of SERA 2006, Seattle, USA, IEEE CS, Aug 2006.
- CoCoME in SOFA website, <http://dsrg.mff.cuni.cz/cocome/sofa>.
- CoCoME website, <http://agrausch.informatik.uni-kl.de/CoCoME>.
- Java PathFinder website: <http://javapathfinder.sourceforge.net>.
- Parizek, P., Plasil, F., Kofron, J.: Model Checking of Software Components: Combining Java PathFinder and Behavior Protocol Model Checker, in Proceedings of 30th IEEE/NASA Software Engineering Workshop (SEW-30), IEEE CS, Jan 2007.
- Plasil, F., Visnovsky, S.: Behavior Protocols for Software Components, IEEE Transactions on Software Engineering, Vol. 28, No. 11, Nov 2002.
- Spin website, <http://www.spinroot.com>.