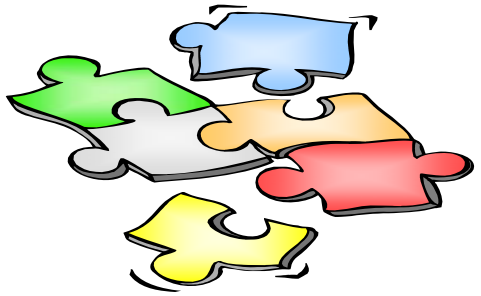
PROGRESS



Analyzing a Pattern Based Model of a Real-time Turntable System

Davor Slutej, John Håkansson,
Jagadish Suryadevara, Cristina Seceleanu, Paul Pettersson

MÄLARDALENS HÖGSKOLA

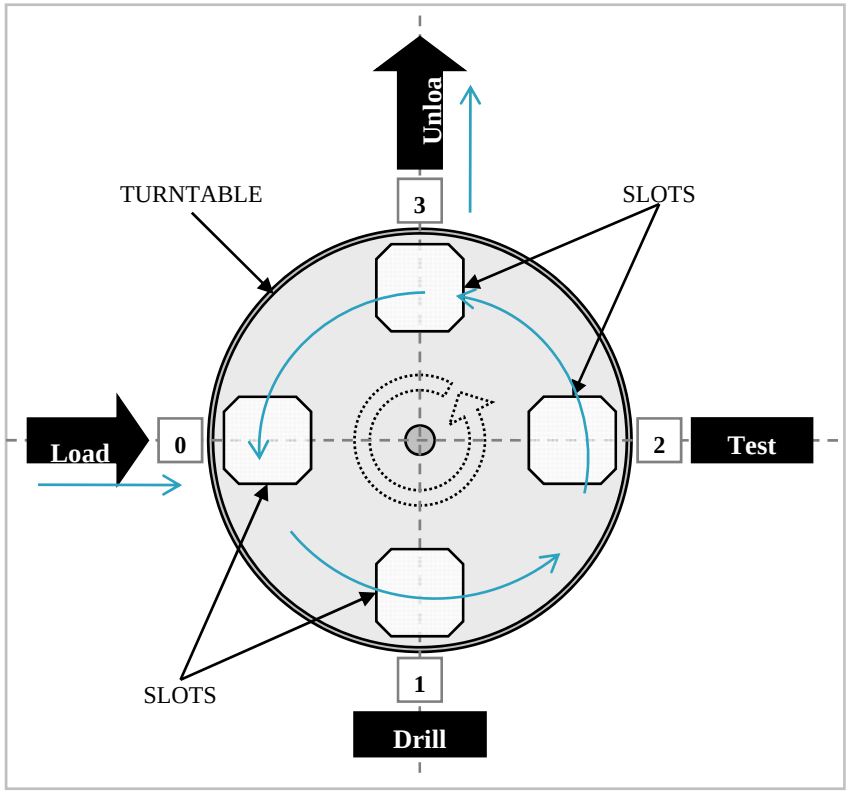
MRTC
MÄLARDALEN REAL-TIME RESEARCH CENTRE

Real-Time Embedded Systems

- Problems
  - IncreasingSystem Complexity
  - Behavior Models
    - Un-Intuitive
    - Non-Reusable
    - Un-maintainable
  - Lack of Predictability
  - Lack of Continuous Development
    - Design⇔ Analysis
- Solutions
  - Component-Based Development
  - Behavior Models
    - Abstract design models
    - Modeling Patterns
  - Formal translation
    - Timed automata
  - Tool Environment
    - SaveIDE, Uppaal-PORT

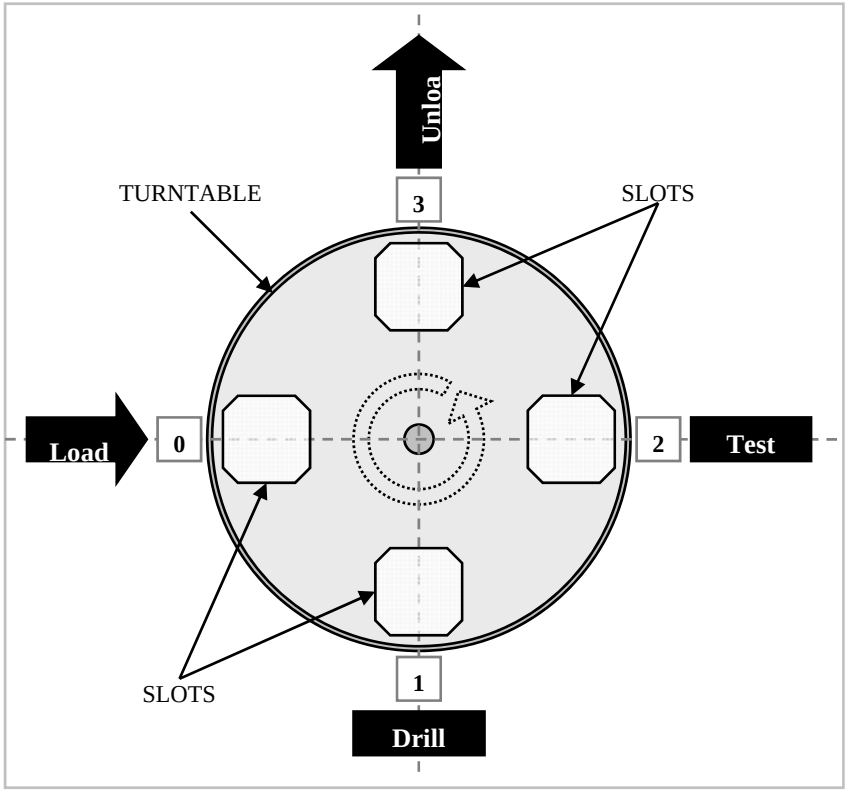
Case Study: Industrial Real-Time Turntable System

- Rotary disc
- Processing Slots
  - Loader, Driller, Tester, Unloader
- Handling Tools
  - Driller, Clamps, Tester
- Workflows
  - Load→Drill→Test→UnLoad
  - Load→(Drill→Test)*→UnLoad
  - Concurrent
- Functional Requirements
  - Safety, Liveness



Case Study: Requirements

- Safety Properties
  - Absence of deadlock
  - When turntable is rotating no other component is executing
  - …
- Liveness Properties
  - System progress, i.e. turntable continuously moves between Idle and executing states
  - Each component progresses
  - …

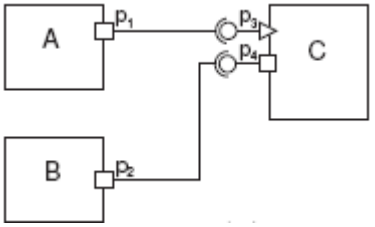


Outline

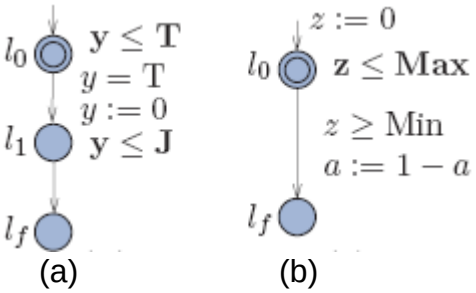
- Introduction
- Case Study
  - Industrial Real-time Turntable System
  - Modeling and formal verification
- SAVE: Component-based Framework
- Turntable Design Model
- Component Modeling Patterns
- Results
- Conclusions

SAVE: Component-Based Framework

- SaveCCM component model
  - Component-based modeling of embedded systems
  - Graphical syntax, Formal semantics
- Component interface
  - Trigger port – activation and transfer of control
  - Data port – input/output data
- SaveIDE tool environment
  - Architectural modeling
  - Behavioral modeling in Timed Automata
  - Formal verification by Uppaal-PORT
    - Uppaal for components and partial order reduction technique



A composite component in SaveCCM

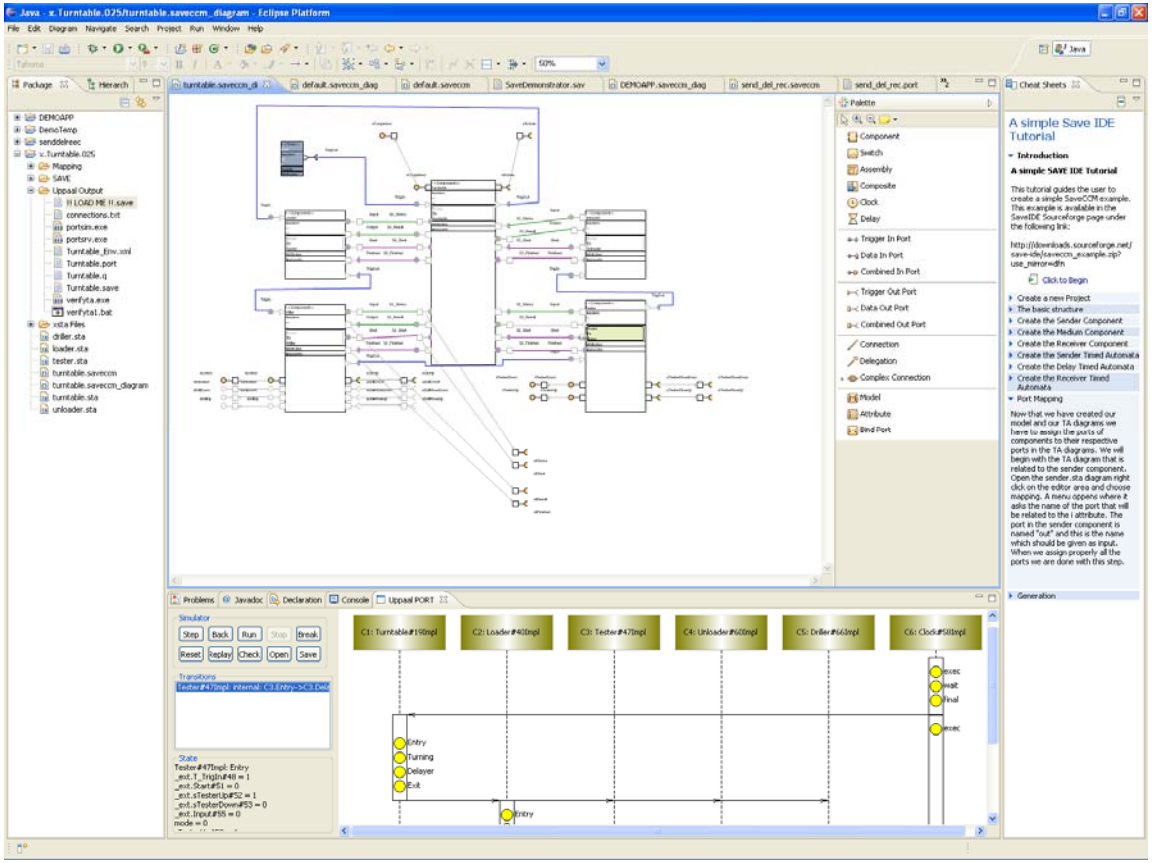


(a) Timed Automata specification of A clock with period **T** and jitter **J**

(b) A computation updating a data variable *a* between ***Min*** and ***Max*** time units

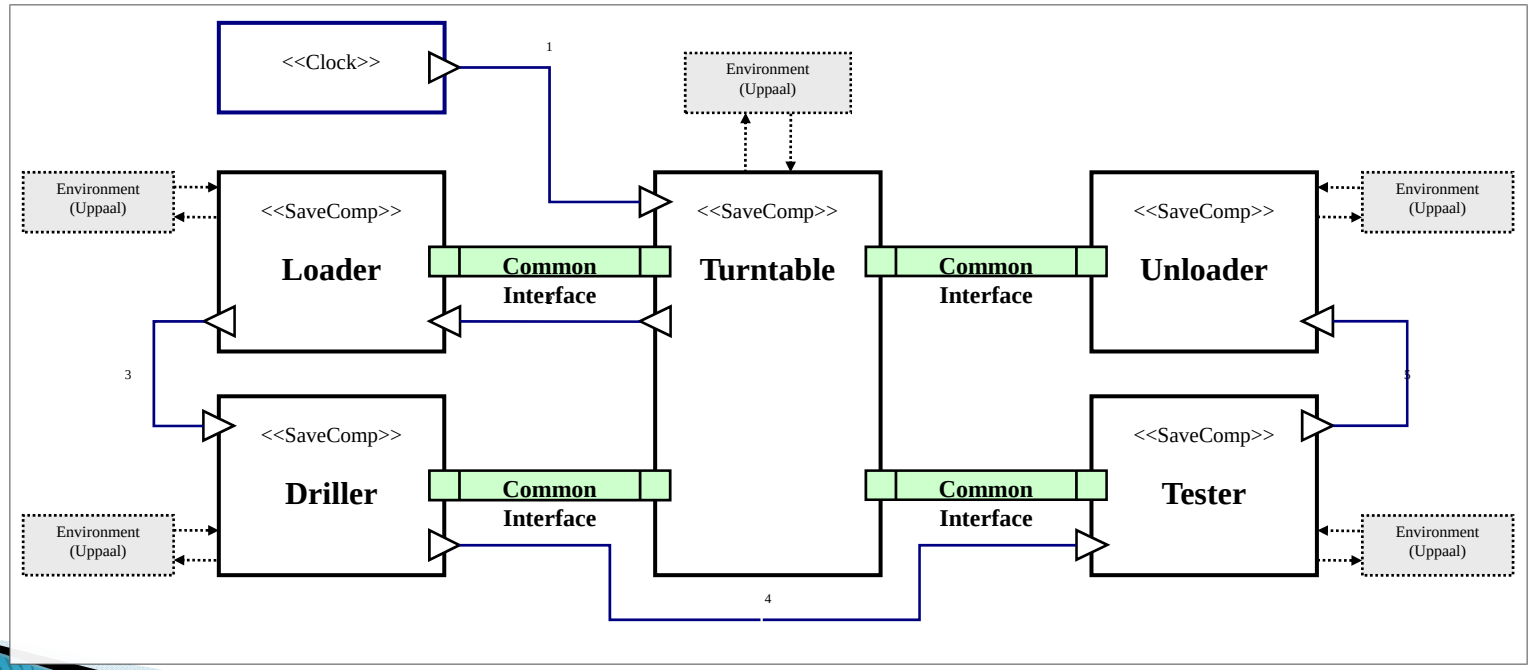
SaveIDE environment: Modeling and Verification

- IDE
- GUI
- Eclipse
- Editors
- Simulator
- Verifer
- UPPAAL Port
  - Sourceforge: save-ide/



Turntable Design overview

- One component per controllable ”device”
- Components interface with the environment and with each other



Implementation: Tester

```
void evaluate()
{
        if (mode==S_READY) {
                if($active) {   $slotresult=$slotstatus;

                 if ($slotstatus==SL_DRILLED)  {
                    $finished=false;
                    mode=S_MOVING_DOWN; ticks=0; }
                 else {
                        $finished=true;
                         mode=S_READY;  }
                }
       else if (mode==S_MOVING_DOWN) {

                ticks++;
                if (ticks<=2) {
                      if ($sensorDown) {
                            $slotresult=SL_TESTGOOD;
                             mode=S_MOVING_UP;}
                               }
                     else
                      { $slotresult=SL_TESTBAD;
                              mode=S_MOVING_UP; }

     else if (mode==S_MOVING_UP) {
                          if ($sensorTop) {  $finished=true;
                                mode=S_READY; }
}
```

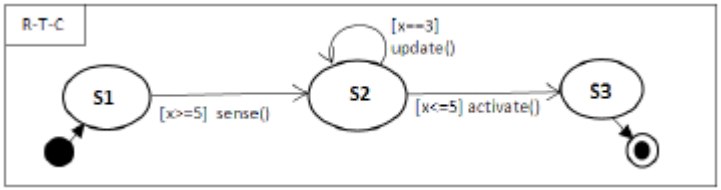
- Typical component
- Characteritics
  - Switch:
    - If…else if… else if…
    - Runs until false if-statement
  - State encoded in variable ”**mode**”
  - Implicit timing

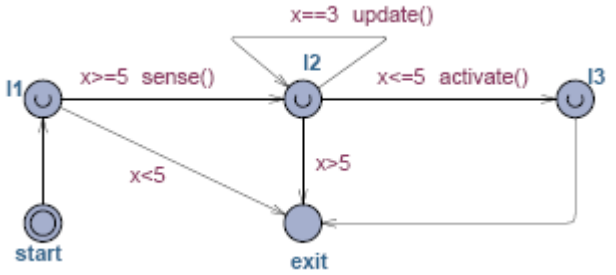
Modeling Patterns

- **Run-to-Completion (RTC) Pattern**
  - Execution in indivisible steps, without interruption from any concurrent activity

    - Example component behavior under RTC-Pattern

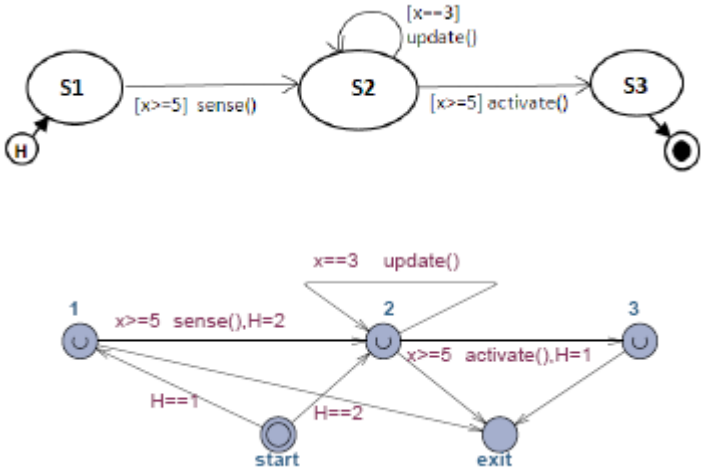
    - Equivalent Timed Automata behavior





Modeling Patterns contd..

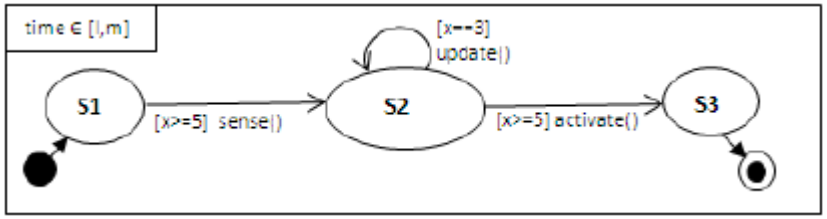
- History Pattern
  - Mechanism for remembering last executed state, before exiting the current execution
    - Example component behavior under History Pattern
    - Equivalent Timed Automata behavior



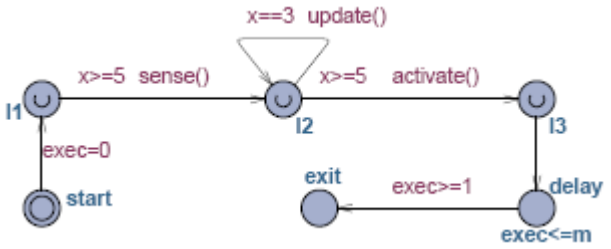
Modeling Patterns contd..

- **Execution-Time Pattern**
  - Specifying best and worst execution times of a component

  - Example component behavior under History Pattern

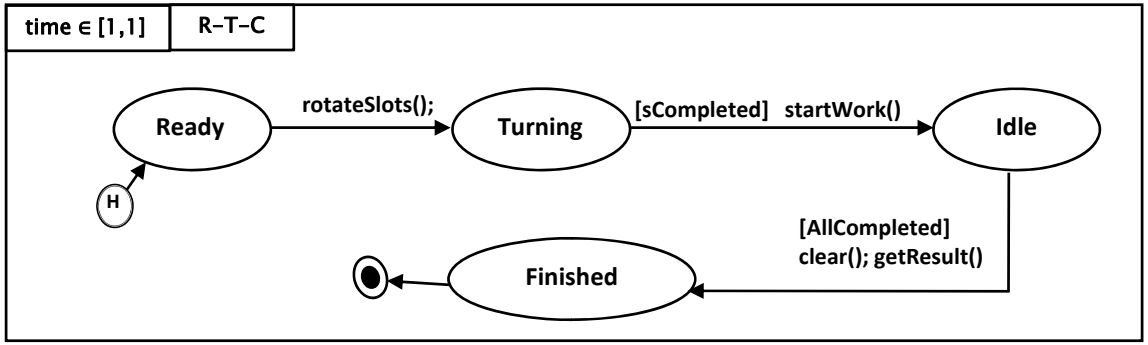


  - Equivalent Timed Automata behavior



Modeling Turntable Components using patterns

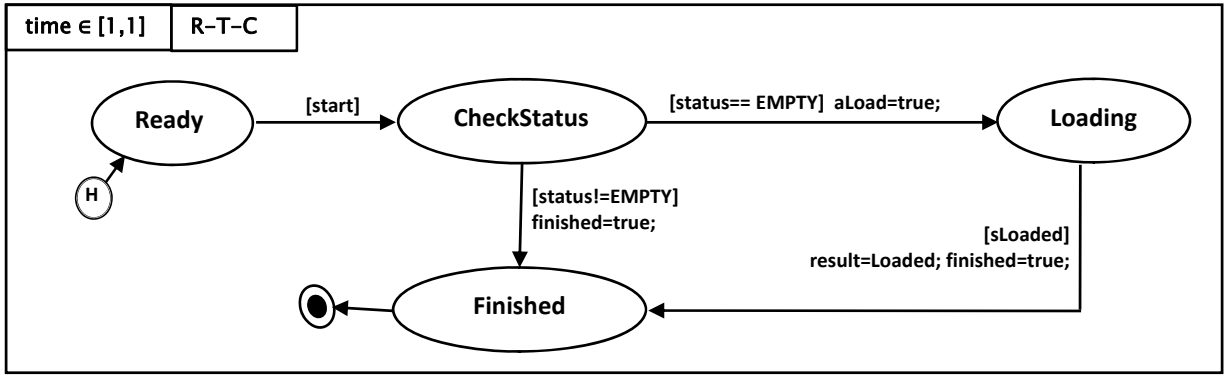
- Modeling the behavior of Turntable component
  1. Rotate disc & set ports of other components
  2. Wait for other components to complete
  3. Repeat above..



Modeling *Turntable* component

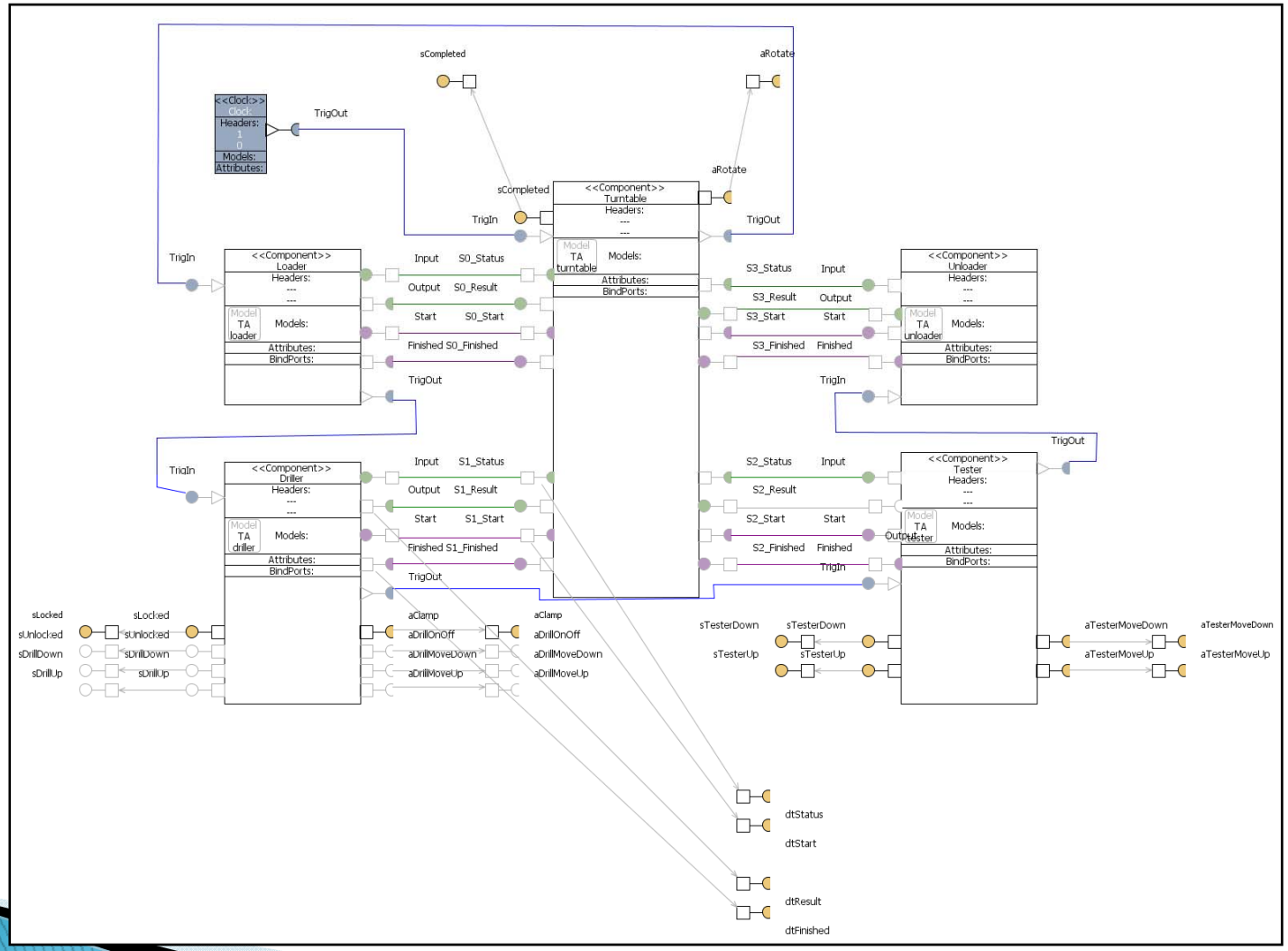
Modeling Turntable Components using patterns

- Modeling the behavior of Loader component
  1. When triggered, check the status of the slot
  2. If a product is still present do nothing
  3. Otherwise load a new product
  4. Repeat above..



Modeling *Loader* component

System Architecture



Model Checking Results

- **A[] not deadlock**
  - A safety property, verifying absence of deadlock situations
- **A[] Turntable.Turning ⇒ (Loader.Ready∧ Tester.Ready∧Unloader.Ready∧Driller.Ready)**
  - A safety property, stating that when the Turntable component is executing, no other components are executing
- **Loader.Ready --> Loader.Finished**
  - A progress (liveness) property, that verifies that Loader always progresses
- **A<> Turntable.Turning Turntable.Turning--> Turntable.Idle, Turntable.Idle --> Turntable.Turning**
  - The above properties establishes that the Turntable always progresses

Model checking Results contd..

- 32 properties were specified
- Verified on a PC w/ Intel T2600 CPU @ 2.16GHz (2GB RAM)
- Most optimization options switched off
- Under 45 seconds to verify all properties
- Less than 18MB used by the verifyer

Conclusion

- Modeling and analysis of real-time embedded systems
- Abstract design models and Behavior Modeling patterns
  - Run-to-completion
  - History
  - Execution-time
- Formal verification of functional requirements
  - A set of safety, liveness properties
- Future work
  - Extension of pattern-framework
  - Component based behavioral modeling language

Thank you.

Questions?