INSTITUTE OF COMPUTER AND NETWORK ENGINEERING

Safe Evolution in Real-Time Systems

Moritz Neukirchner, Steffen Stein, Rolf Ernst
{neukirchner|stein|ernst}@ida.ing.tu-bs.de

TECHNISCHE UNIVERSITÄT CAROLO-WILHELMINA ZU BRAUNSCHWEIG

Challenges

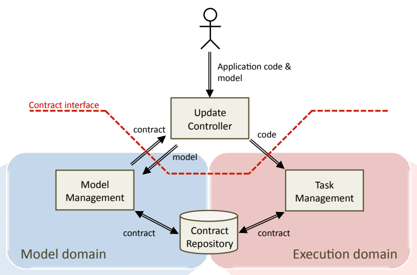
- In-Field updates
  - Increasing share of invention (only) in software
  - Exploding configuration space
- Component/System Variability
  - Aging effects
  - Addition/Removal of System components
- Design Complexity
  - More functionality
  - Diverse system variants
  - Third-party integration
  - Increasingly networked systems

- Complex changing timing behaviour
- Timing properties specific to individual systems

Contracting Architecture

Contracting Procedure:

- An Update Controller, which may reside at an arbitrary position in the System, distributes the application model to all CPUs that are affected by the request
- Model Management evaluates feasibility of the request based on its model and existing contracts
- In case of acceptance the contract is closed and stored in the Contract Repository
- The Update Manager distributes the program code to the corresponding CPUs
- Task Management schedules the code for execution based on their contracts in the repository

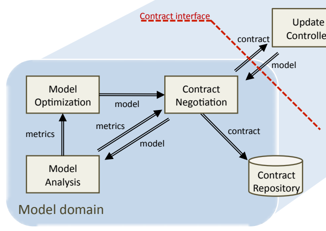


Separation of domains:

- System architecture is separated into two domains: *Model domain* & *Execution domain*
- Model domain performs worst-case performance analysis based on formal methods to ensure system feasibility at all times
- Execution domain enforces parameter settings based on contract information and detects deviation from specified behavior

Advantages:

- Decoupling of analysis and execution → analysis has minimal influence on application execution
- System optimization and analysis can be performed during processor idle times



Model domain:

Contract Negotiation:

- Protocol Handler for communication with Update Controller
- Inserts model into Analysis component and evaluates results

Model Analysis:

- Performs model based distributed performance analysis

Model Optimization:

- Modifies system model based on analysis metrics to optimize configurations
- Multi-objective otimization

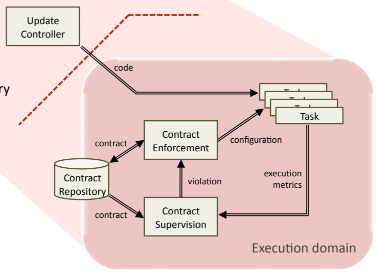
Execution domain:

Contract Enforcement:

- Sets execution parameters according to contracts in the Contract Repository
- In case of contract violations alters system configuration to isolate misbehavior and protect remaining applications
- if applicable alters contract information for new analysis

Contract Supervision:

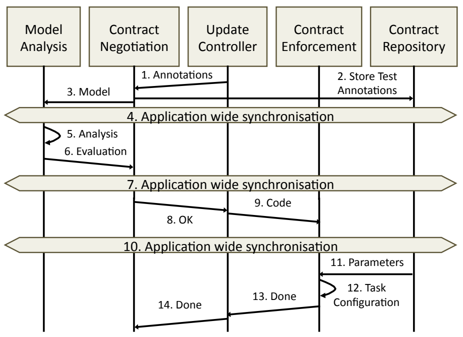
- detects anomalous application behaviour
- monitors hardware performance



Contracting Protocol

The Contracting Protocol in the Model domain

1. The Update Controller distributes the annotations to the Contract Negotiation components of all RTE instances affected by the update.
2. Each Contract Negotiation component stores only a partial model in its local Contract Repository.
3. The Contract Negotiation components store the received Model in the Model Analysis components.
4. An application wide synchronization ensures model consistency across the distributed system.
5. The Model Analysis components analyze the system configuration distributedly and cooperatively.
6. Analysis results are reported to the Contract Negotiation components for evaluation.
7. Consistency of the analysis results is ensured in another synchronization step.
8. The analysis result is reported to the Update Controller. It case of feasibility it can start the distribution of the application.



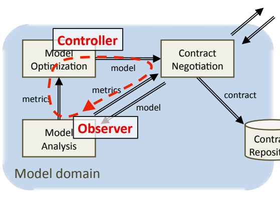
The Contracting Protocol in the Execution domain

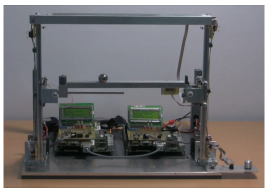
9. The Update Controller distributes the program code of the application to the appropriate resources.
10. In an application wide synchronization the system assures that the code has arrived at all affected resources.
11. The Contract Enforcement components read the execution parameters of the application from the contracts, that are stored in the Contract Repository instances.
12. All Contract Enforcement components start execution of the new or updated application with the appropriate parameters from the contracts.
13. The Contract Enforcement components report successful enforcement of the configuration change to the Update Controller.
14. The Update Controller notifies the Contract Negotiation components of the completion of the update process to allow the next configuration change.

From Self-Protection to Self-Configuration

The Model Domain has been extended by a distributed model-based optimization component to close an Observer Controller loop.

As a result updates do not have to be rejected in case of infeasibility. Instead the framework modifies the system model to evaluate parameter changes. Thus the system can search for feasible configurations, providing a self-configuration service.





The self-configuration service has been implemented for the demonstrator and has proven to find priority assignments that allow acceptance of system changes that would have been rejected otherwise.

The implementation of the model-based optimization is low in overhead and thus blends perfectly into the overall framework concept.

supported by DFG SPP 1183 Organic Computing