Machine Simulation for Workflow Integration Testing

IOAN SALOMIE, TUDOR CIOARA, IONUT ANGHEL,
MIHAELA DINSOREANU, TUDOR IOAN SALOMIE
Department of Computer Science
Technical University of Cluj-Napoca
15 C.Daicoviciu street, Cluj-Napoca, 400020 ROMANIA
Ioan.Salomie@cs.utcluj.ro *http://dsrl.coned.utcluj.ro*

Abstract

This paper addresses the problems of modeling and simulating physical machines, part of complex industrial production lines. The direct execution of industrial workflows on production line machines before integration testing can be very expensive and may lead to improper machine operation and even to non recoverable faults. In order to simulate the execution of industrial workflow models for integration testing purposes, we propose a physical machine simulator based on nondeterministic, probability-based state machines. For each physical machine a behavioral model is constructed using operational scenarios followed by its translation into a state machine representation. The proposed simulator was used for a sausage preparing production line in the context of the Food Trace project [8].

1. Introduction

The problem of deploying and executing industrial process workflows in manufacturing production lines, without having tested them, can lead to sever faults on the production line. The problem can be overcome by inserting a machine-into-workflow integration testing phase between the design-time and run-time.

To simulate an industrial process which includes physical machines, a model that reproduces the real physical machine has to be used. Relevant simulation results are obtained only if the physical machine model accurately reproduces real machine's machine behavior. Due to their level of complexity, modeling real physical machines is not always a simple task, being difficult to describe them with precise deterministic or mathematic models.

The present paper aims to provide a means for simulating industrial machines, so that workflow integration testing can be done in a virtual environment. To obtain a relevant physical machine model we capture its behavioral model using machine's operational scenarios which are then transformed into an equivalent state machine representation. After the simulation, the industrial process reengineering can be used to remodel and correct the workflow faulty processes detected during the simulation process.

In the context of business process modeling and execution, many simulation approaches have been proposed. In [2] the authors present a process simulation system based on interactive events. The system simulates the interactions between composite services and considers two types of events: internal and external business process events. The proposed system architecture is suited for loosely coupled service computing environments and is based on an extension of the XPDL [7] meta-model. The architecture incorporates interactive event flows between individual workflows, explicitly modeled at design time, while the event interactions with data correlation are implemented at run time. The main advantage of this approach is that model verification is being done by using simulated events. The disadvantage is that non-functional properties like performance or QoS can not be simulated using events.

Another approach is the SQMA (Situation-based Qualitative Modeling and Analysis) model described in [1]. The SQMA offers a model for representing and simulating industrial systems using Rough Set Intervals. This work uses interval-based representation for qualitative models for implementing the behavior of real systems. The SQMA model hierarchically

structures the whole system and decomposes the system's levels into components. Component variables are modeled using intervals and characteristic values represented as a one-value interval. Physical rules that are used for the model verification are formulated using interval arithmetic to complete the description of each component. Based on Rough Set Intervals and physical rules, a transition matrix between components is constructed and used in simulation. The main disadvantages of this approach consist of inaccurate representation of machines business logic and difficulties in model management. Also, complex business scenarios involving cooperating machines are difficult to model using this formalism.

In [3], a framework for the simulation of business process workflow models is discussed. The approach uses BPEL language for modeling the workflow which is then transformed into a dataflow network model. A model checker for verifying the correctness of workflow properties is used. The authors have developed a framework for fault simulation by inspecting the dataflow model by using Petri Nets and graph theory. The main disadvantage of this approach is that the run time errors (e.g. "the bank refuses to transfer money") can not be detected because the workflow model in being verified at design time.

Our approach on modeling and simulating industrial business processes is presented in the context of the Food Trace research project [8]. The Food Trace project aims to study, design and implement an integrated IT system for food industry processing organizations, in response to the EU requirements regarding food traceability and quality assurance.

The objective of this paper is to present a simulation environment for physical machines, part of a manufacturing production line, as the main tool for machine-workflow integration testing. This way, the workflow model, involving physical machines can be fully tested in a virtual environment before its execution in the real environment. Our objective was achieved by: (i) capturing and validating the physical machine behavioral model; (ii) representing the physical machine behavioral model using probabilistic state machines and (iii) designing and developing a simulator that uses the state machine model to simulate the physical machines' behavior in the context of industrial workflow model execution.

The industrial workflow model was obtained using the layered construction methodology presented in [4]. The proposed construction methodology facilitates the reuse of organization specific services and allows the modeling of a wide range of business domains while eliminating the incomplete workflow-BPEL mapping.

The rest of the paper is organized as follows. In section 2, we present our approach for representing physical machines as state machines models. In section 3, we present the simulator as a bridge between the industrial workflow model and the physical machine's representation. To illustrate the simulator's functionality, a case study is presented in section 4. Section 5 gives conclusions and promising future work.

2. Physical Machine Modeling

In our proposal, the industrial manufacturing lines are modeled and represented as workflows in which each physical machine is controlled by a web service. The direct execution of industrial workflows on production manufacturing machines, before integration testing, can be expensive and may lead to improper machine operation and even to unrecoverable faults.

For simulating the physical machines we propose a probability-based state machine simulator. In order to obtain the physical machine model used by the proposed simulator, we split the methodology in two steps. First, we will use a formalism to capture the behavioral model of the physical machine. The output of this step is a graph representation of the machine's behavior. The behavioral model is the input for the second step, which creates the state machine model used by the simulator. In the following two subsections we discuss into more detail both steps, and explain the enhancements brought by the state machine model to the behavioral model.

2.1. Behavioral Model

Representing a physical machine as a state machine model is not a trivial task. This is due to having to overcome the gap between the machine specification and the state machine representation formalism, as well as having to avoid logical faults and deadlocks in the state machine representation.

Lately, many efforts have been made to fill the gap between the machine specification and its representation and validation. In [5] the authors propose a method for generating specifications from system goals and representing them in a way that permits reasoning and validation. Another approach is to construct system models that permit validation based on operational scenarios [6]. Using operational scenarios, behavioral models for physical machines can be obtained.

```
MeatCutting  =  (flip -> StartCut),
StartCut       = (startcut -> Temp),
Temp           = (flip -> CheckTemp),
CheckTemp   = (flip ->Normal
                 |flip -> ErrorTemp),
ErrorTemp    = (flip -> MeatCutting),
Normal         = (startcutting -> StartCutting),
StartCutting   = (flip ->CutFinished
                  |flip -> ErrorCutting),
ErrorCutting  = (flip -> MeatCutting),
CutFinished   = (restartmachine -> StartCut
                  |stopmachine -> MeatCutting).
```

Figure 1. MeatCutting machine behavioral model

By using the physical machine's operational scenarios we have identified the physical machine's operational states and the transitions between them. Based on states and transitions we have constructed a machine behavior model that allows validation by using the LTSA tool [9]. Figure 1 presents the behavioral model of a MeatCutting machine in the LTSA formalism.

The validation consists in detecting the possible logical flaws, deadlocks or inconsistencies in the behavioral model. If an error is detected, the behavioral model should be reengineered and verified until no more errors are detected. The MeatCutting behavioral graph model represented in Figure 2 is obtained by compiling the LTSA machine specification of Figure 1.

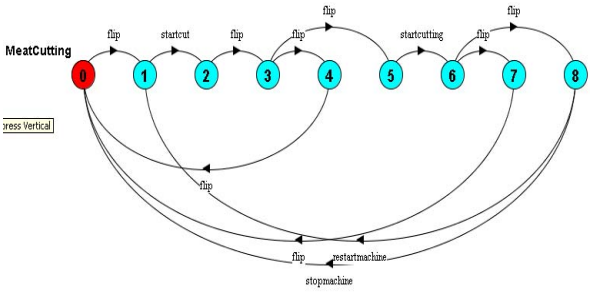

Figure 2. MeatCutting behavioral model graph representation

The advantage of using machine behavioral models is that we can build safe, error-free state machines corresponding to the physical machines.

2.2. State Machine Model

Each physical machine, part of the manufacturing line is represented by a state machine model. Our machine simulator is based on non-deterministic, probability-based state machine models and uses timing constraints and clearly defined error and final states. Since these simulator features are not captured by the behavioral graph model we have proposed its translation into an XML state machine representation.

In this representation, the states and state-transitions of a physical machine behavioral model are mapped onto vertexes and edges, respectively. The events that trigger the physical machines' state transitions are simulated by messages passed to the state machine simulator. Each transition has an associated probability which expresses the chance of changing the current state with a next state. Also, each transition contains timing constants which specify the waiting time before and after the transition. The timing constants can be used to simulate processes which need a given amount of time to complete. An example of a real machine modeled as a state machine is the meat cutting machine presented in Figure 3.

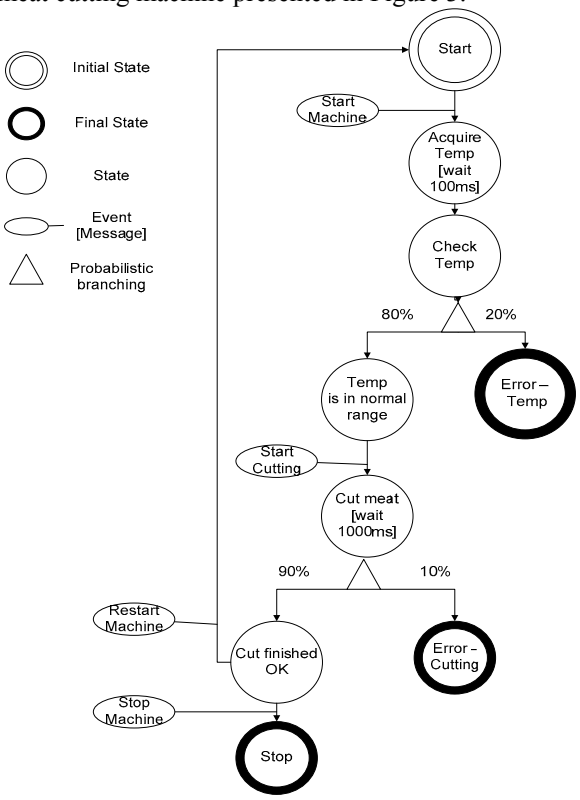

Figure 3. The MeatCutting state machine

Every physical machine, modeled as a state machine, has a corresponding XML description. The main features of the XML representation of the state machine are the initial state, the messages to which the state machine reacts, the valid states, possible actions for each state and their associated probabilities.

The initial state of the state machine is specified by the mandatory "initialState" attribute of the XML root node. The messages that drive the flow of the state machine are grouped under the "messages" element in the XML document and are uniquely identified by their
"name". All states that the machine can be in are enumerated using the "state" element. The state type can be auto, message, error and final. The auto states automatically advance from the current state to the next state, depending on the contained "action" elements. The message states wait for a message to advance. The error states represent logical or physical errors that the simulated machine reached. The final states mark the end of the state machine execution.

Each state defines a list of actions that can lead to a state transition. The actions are associated to XML "action" nodes, each of which may have the "waitBefore", "waitAfter" and "probability" attributes and must have the "nextState" attribute. The "waitBefore" and "waitAfter" attributes express the amount of time in milliseconds to pause before and after executing the actions. The "probability" is a real number in the interval [0.0 ..1.0]. It determines the probability of a specific action to be chosen.

Figure 4 presents the state machine XML definition corresponding to Figure 3 “MeatCutting” state machine.

3. Physical machine simulation

The simulation process is driven by the workflow model of the production line. The execution of the industrial workflow model requires message passing between the workflow and the simulator. This leads to developing the simulator front end as a web service. The BPEL – simulator interaction uses SOAP messages. The main simulator functionality is exposed through a set of web service operations.

The InitializeStateMachine operation creates a new web service session for the simulation of the state machine specified by its name. No other operations can be invoked on the web service prior to successfully executing this operation. The PostMessage operation sends a message to the previously initialized state machine. As a result the current machine state is returned. The ResetMachine operation is used to reset the machine to its initial state.

```
<stateMachine initialState="Start">
  <messages>
    <message name="StartMachine" />
    <message name="StartCutting" />
    <message name="RestartMachine" />
    <message name="StopMachine" />
  </messages>
  <state name="Start" type="message" id="0">
    <message name="StartMachine">
      <action nextState="AcquireTemp"/>
    </message>
  </state>
  <state name="AcquireTemp" type="auto" id="1">
    <action nextState="CheckTemp" />
  </state>
  <state name="CheckTemp" type="auto" id="2">
    <action probability="0.8" nextState="TempOK"
                                waitBefore="100" />
    <action probability="0.2"
            nextState="ErrCheckingTemp" />
  </state>
  <state name="TempOK" type="message" id="3">
    <message name="StartCutting">
      <action nextState="Cut" />
    </message>
  </state>
  <state name="Cut" type="auto" id="4">
    <action probability="0.9" nextState="CutOK"
                              waitBefore="1000" />
    <action probability="0.1"nextState="ErrCutting"/>
  </state>
  <state name="CutOK" type="message" id="5">
    <message name="StopMachine">
      <action nextState="Stop" />
    </message>
    <message name="RestartMachine">
      <action nextState="Start" />
    </message>
  </state>
  <state name="ErrCheckingTemp" type="error"
id="-1" />
  <state name="ErrCutting" type="error" id="-2" />
  <state name="Stop" type="final" id="0" />
</stateMachine>
```

Figure 4. XML MeatCutting state machine

The simulator web service front end is required to handle concurrent access to resources (the state machines) and communication sessions. This

requirement is achieved through a per-session web service instantiation model. This means that for each call to the InitializeStateMachine operation, the simulator web service will create a new simulation process instance. By providing a per-session execution model for the simulator, data integrity is also assured. The instantiation model and session behavior of the simulator system using web services were developed using the Microsoft's Windows Communication Foundation [10].

4. Case study – a simulation scenario

The sausage preparing scenario presented in Figure 5, including traceability features, corresponds to a recipe for sausage manufacturing. Based on this scenario, the corresponding sausage preparing workflow model (Figure 6) was constructed using the layered construction methodology presented in [4].

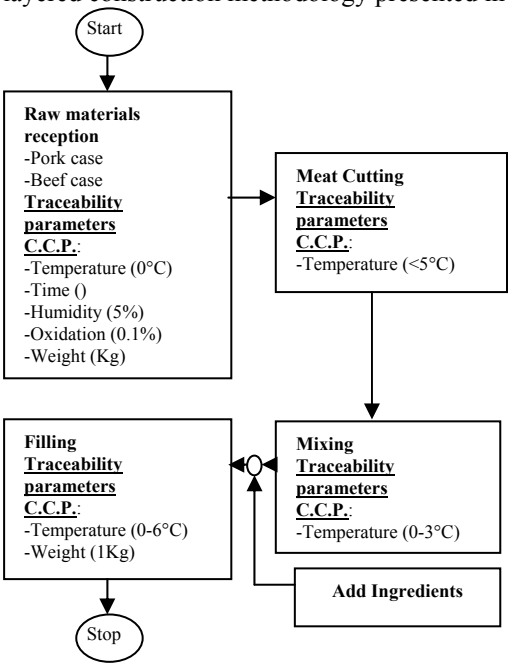

Figure 5. Sausage preparing scenario

According to this methodology a four layer workflow model was obtained. The bottom layer, ARR (Atomic Request/Reply), specifies the atomic services that use a request/reply message exchange pattern. The services on this layer interact with the physical level (real or simulated sensors or simple machines), such as those responsible for acquiring the production line parameters such as temperature, humidity, etc.

The SP (Simple Processes) layer is generated on top of the ARR layer. This layer contains simple processes that are obtained by composing or orchestrating the atomic processes from the ARR layer using domain specific rules. The CP (Complex Process) layer defines complex processes that involve a set of machines working together for achieving a complex task.

The topmost level, the WF (Workflow) layer, represents the workflow which models the entire production line for sausage manufacturing.

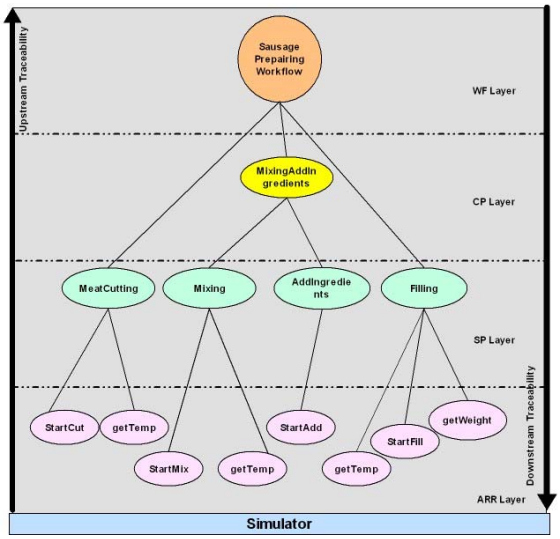

Figure 6. Sausage preparing workflow model

Based on the sausage preparing scenario represented in Figure 5, we have identified the following atomic request/reply processes: getTemperature, getTime, getHumidity, getOxidation, getWeight and machineStart/Stop. They are represented as web services based on the request/reply paradigm, which interacts directly with the simulated or real machines.

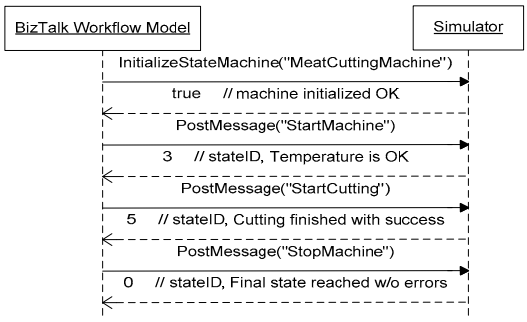

Figure 7. "Meat Cutting Machine" Simulation Sequence Diagram

The simple processes of the SP layer such as "meat-cutting", "mixing" or "filling" are constructed by orchestrating the atomic request/reply web services. Using Microsoft BizTalk Server Orchestrator [11], the simple processes are orchestrated into complex processes. For the complex processes level (CP layer), we have identified the process of "Mixing and Add-Ingredients".

The sequence diagram presented in Figure 7 describes the interaction between the BPEL workflow and the "MeatCutting" machine simulator for the best case test scenario.

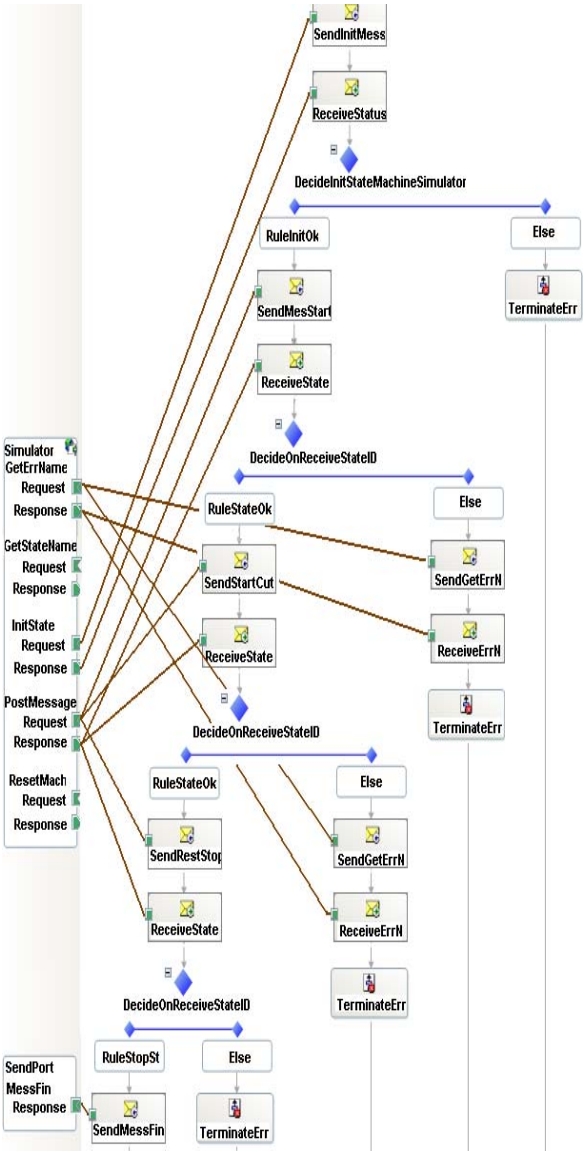

Figure 8. Microsoft Biz Talk workflow

In Figure 8, Microsoft BizTalk Orchestrator is used to describe the interaction between the sausage preparing workflow and the simulated "MeatCutting" machine.

5. Conclusion and Future Development

The present paper presents a solution to the problem of physical machine modeling and simulation that can be used for machine to workflow integration testing purposes. The paper presents the methodology for obtaining a formal description of the simulated physical machines starting from capturing and validating physical machine's behavioral model and then enhancing this model with specific simulating concepts such as probability and timing. The obtained model of the physical machine is a probabilistic state machine that is used in context with the proposed simulator.

Using our simulator, the machine - workflow integration testing can be done before runtime.

Our machine simulator environment is based on non-deterministic, probability-based state machine and uses timing constraints and clearly defined error and final states, thus allowing accurate representations of the machine's business logic and facilitates the management of the obtained physical machine model.

For future work, we intend to improve the simulator by changing the state machine runtime with the Microsoft Workflow Foundation runtime [10]. This will allow the use of a standard state machine XML based representation. Also an inter business approach on simulation, which considers the collaboration among different business partners, is a future enhancement of the simulator.

Acknowledgements

This work was supported by the Food Trace project within the framework of the "Research of Excellence" program initiated by the Romanian Ministry of Education and Research.

References:

[1] M. Rebolledo, Rough intervals — enhancing intervals for qualitative modeling of technical systems, Artificial Intelligence 170, 667–685, 2006.

[2] Y. Zheng, Y. Fan, W. Tan, Interactive-Event-Based Workflow Simulation in Service Oriented Computing, Fifth International Conference on Grid and Cooperative Computing (GCC'06), 2006.

[3] M. Kovacs, L. Gonczy, Simulation and Formal Analysis of Workflow Models, Electronic Notes in Theoretical Computer Science, http://www.elsevier.nl/ locate/entcs.
[4] I. Salomie, T. Cioara, I. Anghel, M. Dinsoreanu, T. I. Salomie, Methodology for Industrial Business Process Modeling Enhanced with Process Algebra Verification, Submitted for ICCP 2007
[5] E. Letier, J. Kramer, J. Magee, S. Uchitel, Deriving Event-Based Transition Systems from Goal-Oriented Requirements Models, http://pubs.doc.ic.ac.uk/deriving-event-based-trans-sys/deriving-event-based-trans-sys.pdf
[6] S. Uchitel, R. Chatley, J. Kramer, J. Magee, System Architecture: the Context for Scenario-based Model Synthesis, FSE 2004
[7] XPDL specifications, http://www.wfmc.org/standards/ xpdl.htm.
[8] FOOD-TRACE project, http://www.coned.utcluj.ro/ FoodTrace/.
[9] LTSA http://www.doc.ic.ac.uk/ltsa/
[10] Microsoft .NET Framework 3.0 http://msdn2.microsoft.com/en-us/netframework/
[11] Microsoft Biztalk Server 2006, http://www.microsoft.com/biztalk/default.mspx
[12] F. Moscato, N. Mazzocca, Vittorini, Workflow Pattern Analysis in Web Services Orchestration: The BPEL4WS Example, 1st International Conference on High Performance Computing and Communications 2005, LNCS 3726, 395-400
[13] H. Li, Z. Lu, Decentralized Workflow Modeling and Execution in Service-Oriented Computing Environment, IEEE International Workshop on Service-Oriented System Engineering, 2005.