Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

MODELING WAITING SYSTEMS FROM DOMAIN EXPERT SPECIFICATIONS

Maâmar El-Amine Hamri, Claudia Frydman, Lucile Torres

LSIS UMR CNRS 6168
Université de Paul Cézanne
Aix-Marseille III
Av. Escadrille Normandie Niemen
13397 MARSEILLE CEDEX 20 FRANCE

ABSTRACT

This paper proposes an application that consists in allocating services by managing waiting queues. The chosen example is the Jaspar bank described by Banks *et al.* The specification is based on the task type, named dynamic assignment, of the methodology CommonKADS. Then, the task type is transformed into an DEVS atomic model based on rules defined to facilitate this passage for the experts which are not familiarized with formal specifications. The operational model is verified and validated by simulation.

1 INTRODUCTION

The activity of knowledge modeling leads to build a model, usually named expertise model. This model is essentially based on the task/method paradigm, where task identifies a problem to be solved, and method is the way to solve the problem. Various knowledge engineering approaches provide rules and/or tools to assist knowledge engineers in developing knowledge based systems. They use to give methodological recommendations allowing the expertise model to be built by decomposing tasks into subtasks until obtaining atomic tasks (top down design). However, more recent approaches advocate to build too the expertise model by reusing expertise components, particularly in form of generic tasks like those provided in the CommonKADS methodology libraries (Breuker and Van de Velde, 1994; Schreiber *et al.*, 1999). This allows expertise models to be partially reused in new applications. The main guide for reuse in CommonKADS methodology consists in identifying a suitable task template by recognizing the task type in the provided hierarchy.

In this paper, we use a CommonKADS dynamic assignment template that plans resources (services) allocation by managing waiting queues. The behavior of service providing system is fundamentally event-driven and describing such a behavior needs the use of event notion and time constraints (Brazile and Swigger, 1988 ; Hovestadt *et al.,* 2003; Motta *et al*, 2002). Describing the inferences behavior in a task method does not consist in defining a sequential ordering of inferences, but must indicate what inferences have to be run when external events happen (Garrido de Ceita *et al*., 2003). Transforming this semi-formal CommonKADS description into a DEVS (Zeigler *et al.*, 1976) model makes available for the knowledge engineers, an executable generic task model, adequate for dynamic assignment providing objective.

We present our approach of event-driven behavior specification on the example of services providing. We define the dynamic assignment template and we transform it into an DEVS atomic model. The validation by simulation of the behavior model is finally discussed in the last section of this paper.

2 THE EXPERTISE MODEL OF THE JASPAR BANK EXAMPLE

2.1 The CommonKADS template for queuing system

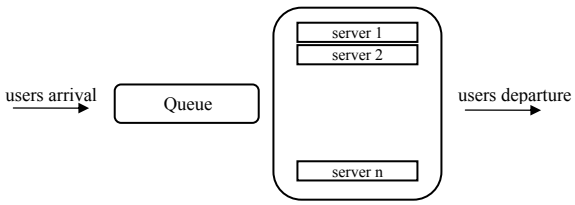

Figure 1. A generic queuing system

We propose a dynamic assignment template which specifies the full assignment/de-assignment process of users to/from servers. The inference structure that we deduce to the dynamic assignment template is shown in Figure 2.

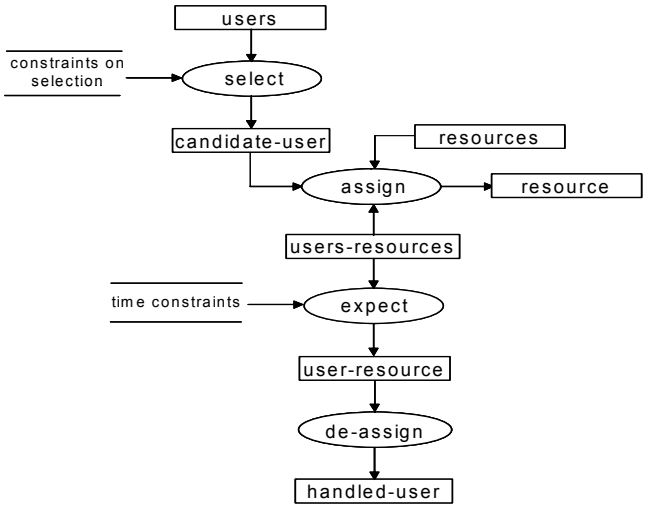

Figure 2. Inference structure for dynamic assignment method

Four inferences are involved for handling the user:
Select: this function selects a candidate user.
Assign: in the assign function, a resource is selected that satisfies constraints connected to, according to the current situation.
Expect: the expect function identifies the next user-resource to be de-assigned among the users currently in handling according to the time constraints.
De-assign: at the end of handling the user, the function releases the seized resource and produces the output handled user.

```
TASK dynamic assignment
    ROLES:
        INPUT:
          users:;
          resources:;
        OUTPUT:
          handled-user:;
    START-END-TIME: [date1, date2]
    SPECIFICATION:;
END TASK dynamic assignment
TASK-METHOD: dynamic-assignment-event-driven-behavior
    REALIZES: dynamic assignment
    DECOMPOSITION
        INFERENCES: select, assign, de-assign, expect
        TRANSFER-FUNCTION: receive;
        ROLES:
            INTERMEDIATE:
                user:;
                resource:;
                user-resource:;
                users-resources:;
REACTIVE-BEHAVIOR:;
    (receive(user), users:=users ADD user);
    (,select(users→candidate-user)
    ([ressource!=0], assign(candidate-user+users+users-resources);
    (,users-resources:= users-resources ADD <candidate-user,resource>);
    (/release, experct(users-resources→user-resource);
    (release, de-assign(user-resource);
    (,users-resources:= users-resources DELETE user-resource);
    ([users!=0], select(users→candidate-user);
    (, assign(candidate-user+users+users-resources);
    (/release, experct(users-resources→user-resource);
END TASK-METHOD dynamic-assignment-event-driven-behavior
```

Figure 3. Dynamic assignment template

The Figure 3 illustrates the task specification and method of the dynamic assignment template. Input roles are users and resources. The role user-handled represents the output role of the dynamic assignment task type. The method is active during a time interval which starts by date1 and finishes by date2. The **receive** transfer function is invoked at each time a new user comes in or submits a request.

Table 1. Features of the dynamic assignment template

Goal:	handle or serve a set of users
Typical:	handling passengers at airports, assignment run
example	ways to planes, etc.
Terminology	
users:	a set of users may be handled
resources:	a set of resources to which a user can be assigned to
release:	internal event indicates that a resource is released
Input:	users arrivals
Output:	departures of users served or handled

An important difference between the dynamic assignment task type and the other task types defined by CommonKADS methodology is that the dynamic assignment task type shows a reactive behavior through state changes: resource idle or busy, user waiting or served. The dynamic assignment task type can be used to analyze and/or to design system. In fact an execution of this task type allows us to obtain results about frequencies of resources utilizations, waiting times in queues, quality of facilities, etc. Those results can be used as input data to redesign systems for minimizing the development cost.

2.2 The example details

Let us suppose the example of Jaspar bank specified in (Banks *et al.*, 2000). The different steps to build the expertise model of the example using CommonKADS are as follows:

1. First, we construct the domain knowledge associated to this problem, and
2. Secondly, we execute the dynamic assignment template after transforming it into an operational description; to compute the average waiting time for the driver customers which is estimated by the expert manager to 4.3 minutes for only the drive-in customers between 11:00 A.M. and 1:00 P.M (rush period).

The teller serves the driver customers that allows transactions. When there is no car (driver) waiting, the teller had other duties, mainly serving walk-in customers. The transactions between the teller and the walk-in customer are mostly commercial in nature taking a considerably longer time than the time required to serve a driver customer. The walk-in customers are only present during a rush period.

From this description, concepts defined for the Jaspar bank example are modeled. The teller concept and the

queue concept that is composite of the car concept. Car and walk-in are sub concepts of the customer concept. The cars arrive with a Poisson distribution function (λ=0.75). The arrivals distribution function is not expressed for walk-in customers, we suppose that they are present at any moment during a rush period, so we do not model the walk-in customer queue.

These concepts lead us to construct a knowledge base and the relation between the car queue and walk-in customers with the teller are expressed with the rule type: serving constraints.

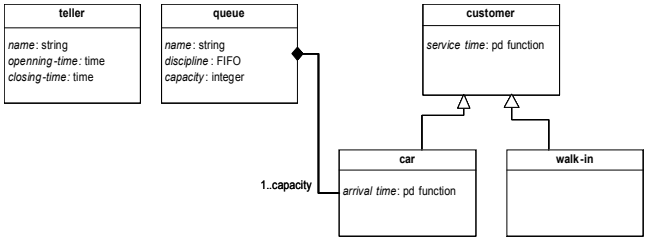

Figure 4. A graphical representation for the Jaspar bank concepts

As we can see in the example, the service time is expressed by a probability function and not with a time interval defined by min-max value. Thanks to the input data modeling phase which provides us to obtain probability distributions of data for the service time (*Normal(1.1, 0.04)*, *Exponential(3)* for drive-in and walk-in customer service time respectively).

The rule type serving constraints is described as follows:

```
Rule-Type serving constraints
Description: "rule permits to link or to serve a cus-
tomer according to the current situation";
   Argument-1: customer;
   Argument-2: teller;
   Connection-Symbol: Serves;
End rule type serving constraints;
```

From this rule type, we can derive the different assumptions of the bank example:

```
car waiting Serves queue // driver customers
No car waiting Serves walk-in customer
```

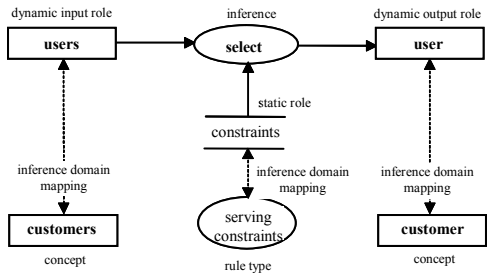

Figure 5. How the **select** inference is related to domain knowledge via the knowledge roles

3 TRANSLATING THE EXPERTISE MODEL INTO DEVS MODEL

The specification of the CommonKADS templates is defined with a semi formal language; it does not make possible simulations (Torres and Frydman, 2001). In fact, the specification of templates is based on analytical methods. Using the extended inference definition to specify tasks allows to obtain models which can be simulated.

To operationalize the expertise models of CommonKADS, we adopt the DEVS formalism such as an operationalization language. Briefly, we recall this formalism:

$$DEVS = <S, X, Y, \lambda, \delta_{int}, \delta_{ext}, \partial>$$

where:

- S is the set of states (not necessarily finished),
- X is the set of input events,
- Y is the set of output events,
- λ: S → Y is the output function,
- δ_{int}: S → S is the internal transition function,
- δ_{ext}: Q x X → S is the external transition function,
- ∂: S → R^+ is the live time function. ∂ (s) is the lifetime during which the model will remain in the state s, if no external event occurs. The total state set of the system specified in DEVS is Q = {(s, e)/ s∈ S, 0≤e≤∂(s)}. The classic function of transition is composed of two functions:
  - the internal function δ_{int}(s), representing autonomous evolutions. δ_{int}(s) is activated when the elapsed time e in the given state will be equal to its length of life ∂(s),
  - the external function δ_{ext}(s, e, x), owed to the external events. If x ∈X arrives, and the system is in state s for an elapsed time e, it transits immediately to δ_{ext}(s, e, x). Simultaneously, the elapsed time e is reset to zero.

To transform the expertise model of CommonKADS into an DEVS model, we propose the following rules:

- X: represents the set of external events, which can be noted using the transfer functions of CommonKADS.
- Y: a set of output roles of the task.
- S: (∏input roles R_i x ∏intermediate roles R_j x σ) / R_i and R_j are roles of the task, σ is a real positive variable. The σ variable depends on the time constraint defined to the inference triggered, and must be updated when external events occur by subtracting the elapsed time e in the current state.
- δ_{ext}(S, e, x): the sequence of inferences specified in the task that are triggered with the external event x and compute the new values of roles.
- λ(S): the last inference of the task that computes the output role.
- δ_{int}(S): the sequence of inferences that are triggered with an internal event and computes the new values of roles. Internal events represent output events of some inferences.
- ∂(S) =σ. If resources belong to input roles of the task to model, σ is n-tuples $\sigma_1,\ldots,\sigma_i,\ldots,\sigma_n$ where i is the resource identifier and ∂(S)=Minimum($\sigma_1,\ldots,\sigma_i,\ldots,\sigma_n$).

Based on these rules, the expertise model specified above (figures 2, 3 and 4) is transformed into an DEVS atomic model, as follows:

$M = <X, Y, S, \lambda, \delta_{int}, \delta_{ext}, \partial>$

$X = \{customer_i / customer_i \in users\ i = 1..n\}$

$Y = \{customer_i / customer_i = handled\text{-}user \in handled\text{-}users\ i = 1..n\}$

$S = (users \times resources \times user\text{-}resource \times users\text{-}resources \times user\text{-}resource \times \sigma) / \sigma = \sigma_1, \ldots, \sigma_i, \ldots, \sigma_n$ are real variables}

External transition function $\delta_{ext}(S, e, receive(user))$:
  users =users ADD user;
  select(users→candidate-user);
  if (resources!=0) then assign(candidate-user + resources + users-resources→resource);
  users-resources =users-resources ADD <user, resource>;
  expect(users-resources→user-resource);

Output function $\lambda(S)$:
  $\lambda(S)$=de-assign(user-resource→handled-user);

Internal transition function $\delta_{int}(S)$:
  users-resources = users-resources DELETE user-resource;
  if (users!=0) then select(users→candidate-user);
  assign(user + resources + users-resources→resource);
  users-resources = users-resources ADD <user, resource>;
  expect(users-resources→user-resource);

The time advance function $\partial(S)$:
  $\partial(S) = Minimum(\sigma)$

4 SIMULATION OF THE JASPAR BANK EXAMPLE

To simulate the DEVS model corresponding to a formal description for the dynamic assignment template, the expert must define the initial number of users (customers) waiting or present in the system. By default it is supposed zero. The initial state of the different resources at the beginning of the simulation can be deduced from the domain knowledge.

Whenever a user arrives, the **assign** inference verifies if there is an available resource, if it is the case the customer is affected to the available resource, otherwise it is put in queue. An internal event will be expected at a date computed by the DEVS simulator (t+∂(S)) to simulate the client departure occupied the resource. This event leads to serve a new customer waiting if the resource is yet available.

To verify the outputs model of the Jaspar bank example, let us suppose that we have three arrivals at dates 1, 1.5 and 3 t.u and their service times are 1, 4 and 4 t.u. When simulation runs, we obtain the following behavior.

To validate the model, we use a validation test (Banks *et al.*, 2000) which consists of comparing the real system output, the average waiting time (delay) in car queue estimated by the expert to 4.3 minutes, to the model output *Y*. Formally, a statistical test of the null hypothesis is conducted:

H_0: Average delay in car queue E(Y) = 4.3 minutes
H_1: Average delay in car queue E(Y) ≠ 4.3 minutes

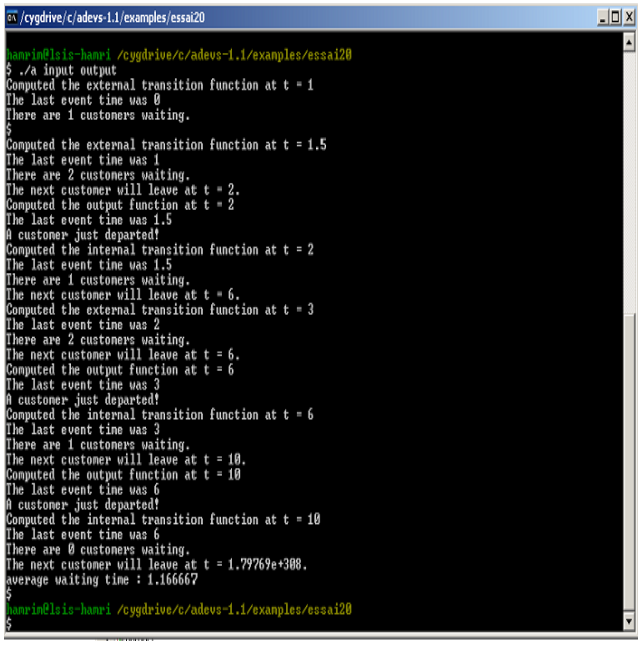

Figure 6. Trace of simulation of the example

We compute the sample mean *Y* (average waiting time) and the sample standard deviation *S* over the *n* replications:

Table 2. Results of six replications of the simulation model

Replication n	Arrivals number per hour	Average waiting time (minutes) Y
1	44.5	6.01
2	44	4.31
3	44.5	3.56
4	42	2.94
5	38	3.94
6	46	4.70

where Y_i i=1..6 are as shown in table 2. For a level of significance $\alpha = 0.05$ and a sample size $n = 6$, for validating the bank model, we get the critical value of *t* from the Chi-Square table. Here $t_{\alpha/2,n-1} = t_{0.025,5} = 2.571$ for a two-sided test.

$$Y = 1/n \sum_{i=1}^{n} Yi = 4.24 \text{ minutes} \qquad S = \left[\frac{\sum_{i=1}^{n} (Yi - Y)^2}{n-1} \right]^{1/2} = 1.05$$

We compute now the test statistic t_0:

$$t_0 = \frac{Y - \mu 0}{S / \sqrt{n}} = -0.13$$

where μ_0 is the specified value of the null hypothesis H_0 (μ_0 = 4.3 minutes).

The model is valid to predict the outputs of the system if H_0 is accepted, that means the value computed from the test static t_0 is smaller than the value got from the Chi-square table $t_{\alpha/2,n-1}$.We have $|t_0| < t_{0.025,5}$ (0.13 < 2.571), we accept the H_0 null hypothesis and we conclude that the model is correct to predict the average customers bank delay. Thus, we conclude that the DEVS atomic model is

valid to model a complex (waiting) system and to predict the system outputs.

5 CONCLUSION

From the CommonKADS specification of the Jaspar bank example, we obtained a computerized model based on rules of transformation proposed into DEVS models, to allow simulations. The operational model of the dynamic assignment template is made with DEVS formalism and the statistical test is used to validate the model. Before this test, data of the system must be approximated to define distribution functions to obtain more realistic simulations. In reality, these functions are not easy to define and errors may occur when data are collected or parameters of distribution functions are estimated, so the results of the system are less realistic and less accurate (Robinson, 1999). We note also that the knowledge on time is expressed by experts with a min and a max values and not with a precise value.

Our current works consist on using Min-Max DEVS (Giambiasi and Gosh, 2001) formalism to obtain more realistic results of the real system when the classical approaches can not be applied.

REFERENCES

Banks, Jerry , John S. Carson, Barry L. Nelson, and David M. Nicol. 2000. *Discrete Event System Simulation*, Prentice Hall, 2000.

Brazile, Robert P., and Kathleen M. Swigger. 1988. GATES: an airline gate assignment and tracking expert system. *Journal of IEEE Expert*, 3 (2): 33-39, 1988.

Breuker, J., and W. Van de Velde. 1994. *CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, The Netherlands, 1994.

Garrido de Ceita, Aguinaldo, Lucile Torres, and Claudia Frydman. 2003. Specifying and Simulation of Reactive Knowledge Based Systems. In *Proceedings of the Summer Computer Simulation Conference (SCSC2003)*, Montreal, Quebec, Canada, July 20-24 2003.

Giambiasi, Norbert, and Sumit Gosh. 2001. Min-Max-DEVS: A new formalism for the specification of discrete event models with min-max delays. In *Proceedings of the 13th European Simulation Symposium (ESS2001)*, pages 616-621, Marseille, France, October 18-21 2001.

Hovestadt, M. , O. Kao, A. Keller and A. Streit. 2003. Scheduling in HPC Resource Management Systems: Queuing vs. Planning. In *Proceedings of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2003)*, pages 1-20, Springer, 2003.

Motta, E., D. Rajpthak, Z. Zdrahal and R. Roy. 2002. The Epistemology of Scheduling Problems. In *Proceedings of the 15th European Conference Artificial Intelligence (ECAI2002)*, pages 215-219, Lyon, France, July 21-26 2002.

Robinson, Stewart. 1999. Three Sources of Simulation Inaccuracy (And How to Overcome Them). In *Proceedings of the 1999 Winter Simulation Conference*, pages 1701-1708, P. A. Farrington, 1999.

Schreiber, G., H. Akkermans, A. Anjeweirden, R. De Hoog, N. Shadbolt, W. Van de Velde, B. Wielinga. 1999. *Knoweldge Engineering and Management: The CommonKADS Methodology*, MIT Press, London, England, 1999.

Torres , Lucile, and Claudia Frydman. 2001. Utilisation des réseaux de Petri pour la vérification et la validation de modèles d'expertise KADS. *Journal Intelligence artificielle*, Hermès Science Publication, 15 (2): 247-276, 2001.

Zeigler, Bernard. P., H. Praehofer and Tag Gon Kim. 1976. *Theory of Modeling and Simulation,* Academic Press, 2000-1976.

AUTHOR BOGRAPHIES

CLAUDIA FRYDMAN is a professor at the university of Paul Cézanne Aix Marseille III (France). She carries out her research interests at the "Laboratoire des Sciences de l'Information et des Systèmes" (LSIS) laboratory. Her main activities are in the field of the knowledge modeling and simulation. She has been a referee for several scientific journals and a member of the program committee in various international conferences. She has conducted and participated in several international and European projects. She is a member of the McLeod Modeling and Simulation Network (M&Snet). Her web and email addresses are:
www.lsis.org\frydman
claudia.frydman@lsis.org.

LUCILE TORRES is an assistant professor at the university of Paul Cézanne Aix Marseille III. She has been for a long time a consultant for many societies. Her main activities are in the field of knowledge engineering and simulation. She is a member of the McLeod Modeling and Simulation Network (M&Snet). Her web and email addresses are:
www.lsis.org\torres
lucile.torres@lsis.org.

MAAMAR HAMRI is a PhD student at the university of Paul Cézanne of Aix-Marseille III. He has joined the LSIS laboratory since 2001. He has participated in various international conferences on modeling and simulation. His web and email addresses are:
www.lsis.org\hamri
amine.hamri@lsis.org.