Perspectives in Modeling: Simulation, Database and Workflow

John A. Miller, Amit P. Sheth and Krys J. Kochut
Large Scale Distributed Information Systems Lab (LSDIS)
Department of Computer Science
The University of Georgia
Athens, GA 30602-7404
email: <*jam,amit,kochut*>*@cs.uga.edu*
URL: *http://LSDIS.cs.uga.edu*

Abstract

Development of today's advanced applications is increasingly being accomplished using multi-faceted modeling. For example, the areas of simulation and workflow modeling generally need data modeling as a foundational capability. In addition, simulation modeling and workflow modeling can be used together, synergistically. Based on the experience of the LSDIS group in developing systems and models, we have found that establishing rich linkages between disparate models works better than having one comprehensive unified model. In addition, we agree with the consensus that two dimensional models are generally considered to be easier to create and understand than one dimensional models. Furthermore, just as richly linked text is referred to as hyper-text, richly linked diagrams may be referred to as hyper-diagrams. Two modeling toolkits, METEOR Designer and the JSIM Modeling Toolkit, illustrate the advantages of using such approaches.

0.1 Introduction

The route to rapid application development is increasingly becoming model oriented. To develop complex applications, facets can be described visually at a high-level using models. Many years ago models were hand translated into code or schema specifications. How tedious. Later, tools and systems were developed that utilized the artifacts of the modeling process to automatically produce part of the application. Today, there is a proliferation of modeling techniques in use. A grand unification into a single notation is probably not useful, since models need to focus on salient features of reality from different perspectives (e.g., data, process, events, usage, authorization, etc.). Still, efforts such as the one producing the Unified Modeling Language (UML) [Sof97] are useful to reduce the learning curve required to use a new modeling technique and to provide common foundations and interrelationships between disparate models. Models are used in nearly all areas of science, engineering and business. This paper examines modeling from three distinct perspectives, Simulation, Database and Workflow, based on our experience developing systems and models in these three areas.

0.2 Types of Models

Given a system S(t) evolving over time, a model is simply an approximation M(t) of S(t). Commonly, models are classified according to how they deal with time (*Static vs. Dynamic*), state (*Discrete vs. Continuous*) and randomness (*Deterministic vs. Stochastic*). Two additional criteria are mentioned below that may be used to help assess what constitutes a good model.

- *Abstractness vs. Similarity.* Abstractness measures the level of repression of detail. This is useful for two distinct reasons: First, real systems have too many details to be fully understood in any timely fashion, if at all. Second, abstraction leads to generality. An abstract model may be used to design/analyze several systems simply by adjusting its parameters. Abstractness is fine, but needs to be counterbalanced by another desirable property of models, namely similarity. Similarity (also referred to as fidelity) means that the model should reflect the characteristics of the real system (e.g., similar shapes and interconnections for objects, proportionate scaling as well as analogous performance, behavior, etc.). One should avoid models with low abstraction and low similarity. Obviously, models with high abstraction and high similarity would be ideal. Unfortunately, this is not possible as illustrated in figure 0.1. There is a tradeoff between abstraction and similarity. Depending on the goals of the modeling effort, good arguments can be made for choosing different points along the modeling frontier.
- *Dimensionality of Models.* Another useful way to distinguish modeling techniques is according to the dimensionality of their representations.
  1. <1D>. Most commonly, one dimensional models are expressed as *text.* Both formal and informal languages can be used.
  2. <1.5D>. A useful improvement on plain text is *hyper-text*, allowing rapid navigation from general concepts to detailed information.
  3. <2D>. Since humans are visually oriented, moving to higher dimensions usually makes the models easier for humans to create and understand. A general term for such two dimensional models is *diagram.* Of course, diagrams can be annotated (on screen or with a pop up window) with text or hypertext.
  4. <2.5D>. As mentioned earlier, it is useful to have multiple types of models as well as multiple instances of the same type of model. It is also useful, if these diagrams are linked to support navigation from one related diagram to the next. This facilitates rapid changes of perspective. One can quickly move from a global view to a detailed view (e.g., by clicking on a compound task icon to display its subworkflow). Alternatively, one could change from a functional view (map design) to a data view (data design) by clicking on an arc connecting two task icons to see the data that flows between them. Analogously to converting text into hyper-text, this converts diagrams into *hyper-diagrams.*

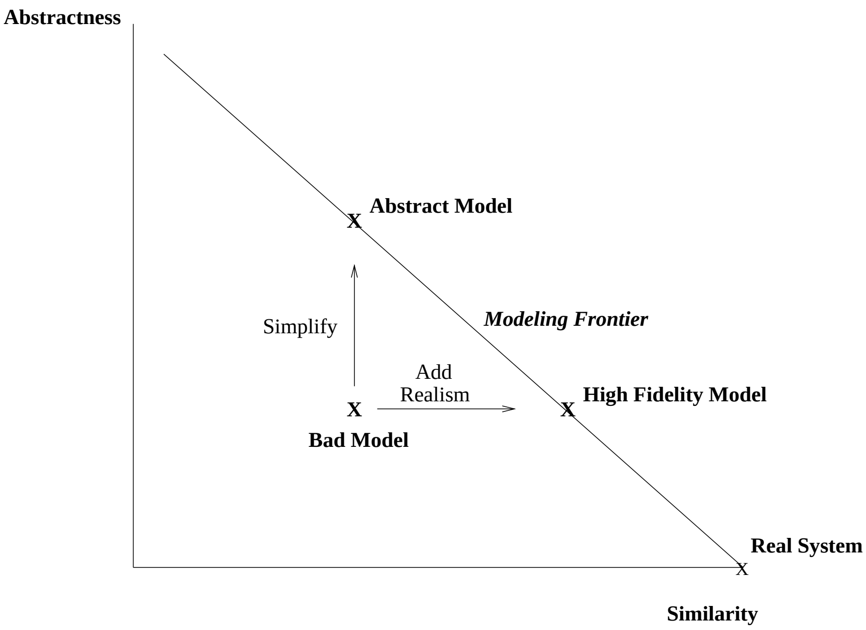

Fig. 0.1. Abstractness vs. Similarity

5. <3D>. To increase the realism of models, creating three dimensional representations is helpful. The advent of cheap fast processors with large memories and accelerated graphics makes this feasible. For example, Taylor II supports the development and animation of three dimensional simulation models. [1] One could term the process of producing a three dimensional representation as creating a *scene*.
6. <3.5D>. If scenes are linked to support navigation from one scene to another, one could term this a *hyper-scene*. The emergence of virtual reality on the Web is an example of this.

Note, half dimensions are used here in an intuitive as opposed to mathematically rigorous sense. This paper concentrates on the current state-of-the-practice, two dimensional modeling, as well as the current state-of-the-art, two and half dimensional modeling. For these type of models, the most popular modeling techniques for simulation and database are briefly described. For workflow, one possible taxonomy is mentioned.

0.3 Simulation Modeling

Of the numerous types of modeling, simulation usually stresses similarity. Although abstract simulations are useful, they are often used as a stepping stone to building a higher fidelity model. Simulation models as a representation of

[1] See *http://www.taylorii.com/scrnsht.htm* for a gallery of screenshots.

an existing or proposed system, strive to capture performance characteristics as closely as possible. Some of them also endeavor to mimic the behavior of real systems. This is particularly important if animation is supported.

There are two broad types of simulation modeling: continuous simulation and discrete-event simulation. The distinction is based on whether the state can change continuously (water level in a reservoir) or at discrete points in time (number of customers in a bank). Discrete-event simulation models are very popular for modeling many types of real-world systems such as banks, hospitals and transportations systems, so we will focus our attention on them.

For discrete-event simulation modeling there are three main world views (simulation modeling paradigms) that can be used: event-scheduling, activity-scanning and process-interaction. Although state-transition diagrams (e.g., finite-state automata or Markov chains) can be used for simulation modeling, they are less common because the state spaces involved are typically very large.

Most of the diagrammatic simulation modeling techniques implicitly or explicitly depict entities (e.g., bank customers) flowing through a system. The classification below mentions five of the most popular general simulation modeling techniques partitioned according to their principal world view. Note, space limitations prevent the display of example diagrams, however, an expanded version of this paper is available on the Web which includes these diagrams. [2]

0.3.1 Event-Scheduling

These models focus on the events that can occur in a system. An event instantaneously transforms the state of the system and schedules future events.

- *Event Graphs (EG).* In an event graph, nodes represent events, while directed edges represent causality (one event causing another) [Sch83]. Edges can be annotated with time delays and conditions.

0.3.2 Activity-Scanning

These models focus on activities and their preconditions (triggers). An activity consists of an event-pair (a start event and an end event). Activities are scheduled when their preconditions becomes true.

- *Activity Cycle Diagrams (ACD).* These diagrams (graphs) depict the life-cycles of interacting entities flowing through a system. The nodes in the graphs represent either activities or wait states (e.g., a queue) [Pid92]. These models can be used to generate simulations following either the activity-scanning or process-interaction world views.

[2] See *http://lsdis.cs.uga.edu/publications/model.ps.*

0.3.3 Process-Interaction

These models focus on processes and their interaction with resources. A process captures the behavior of an entity. Using an object-oriented approach, entities may be treated as active objects (e.g., in JSIM [NMZ96] this is done by deriving them from `Thread` and implementing the `run ()` method).

- *Activity Diagrams (AD).* Activity diagrams are graphs consisting of a well defined set of functional nodes such as start, terminate, delay, engage resource and release resource [Bir79, PH91]. The graph shows the flow of entities through the system.
- *Network Diagrams (ND).* Network (or block) diagrams are used by many popular commercial simulation packages (e.g., GPSS [Sch74], SLAM [Pri79] and SIMAN [PSS90]). These network diagrams are similar to activity diagrams but have more type of nodes corresponding to the underlying primitives supported in their simulation languages. The models are very useful, but have been criticized for being ad hoc and having too many types of nodes. JSIM models fall under this category, but have a smaller number of node types to choose from.
- *Petri Nets (PN).* Petri nets are graphs with two types of nodes, places and transitions [Pet77]. A place is a storage area for tokens (entities), while a transition takes input token(s) to produce output token(s). A transition will fire if there is a token at each of its input places. In Timed Petri Nets, transitions have delays associated with them. In Colored Petri Nets, tokens can have attributes.

0.4 Data Modeling

These techniques model data objects (their attributes and possibly methods) and relationships between data objects. Data modeling has traditionally been a static approach. With the advent of active databases and multimedia databases as well as the need for workflow models to include sophisticated data models, modeling of activity or behavior is becoming more important. The popular types of diagrammatic data modeling techniques are listed below.

- *Entity Relationship Models (ERM).* Entities are interconnected by relationships to form an easy to read graphical structure [Che76]. Relationships have cardinality and participation constraints that are very useful. This model has been used extensively in practice and has been extended in numerous ways.
- *Functional Data Models (FDM).* In this modeling technique, binary relationships are treated as functions [KP76].

- *Semantic Data Models (SDM).* The semantic data model added the concepts of generalization and aggregation into a convenient graphical form [HM81].
- *Object Modeling Technique (OMT).* This modeling technique evolved from ERM, converting entities into objects, allowing methods in addition to attributes to be specified [R+91]. It also includes generalization, aggregation and association.
- *Unified Modeling Language (UML).* UML represents a unification of three popular modeling techniques (OMT, Booch and OOSE) [Sof97].

Note, OMT and UML are themselves multi-faceted making them also usable in many other areas besides data modeling.

0.5 Workflow Modeling

Workflow models focus on the tasks (or activities) and the flow of control/data between them. Unlike the more mature areas of simulation and database, workflow has yet to establish any widely accepted taxonomy of workflow models. However, several fundamental models of computations, especially distributed computation have provided a basis or framework for developing workflow process models. Principle among these are Temporal Logics such as CTL, Petri Nets, State and Activity Charts and Speech-Act Theory (see [S+97] for further discussion). A comprehensive workflow process model, besides modeling the process (also called workflow map or coordination of activities) may also have secondary models to support modeling of data objects involved in workflow processes, error handling, etc.

0.6 Interactions between Models

As mentioned earlier, complex systems should be modeled using multiple modeling techniques. Using hyper-diagrams to establish rich and useful connections between the component models provides as great an advantage over ordinary diagrams as hyper-text does over text.

In the rest of this section, useful interactions between simulation, database and workflow models are considered. Then, two concrete examples, the METEOR Designer and the JSIM modeling toolkit, of hyper-diagrammatic modeling toolkits are examined.

- **Simulation Modeling**. Simulations can be used to refine Workflow Models.
- **Data Modeling**. Data objects can be used in both Simulation and Workflow Models.

– **Workflow Modeling**. Statistics from workflow executions can be used to calibrate and later validate Simulation Models.

The group in the LSDIS Lab is currently in the process integrating or establishing interoperability between a simulation system/environment, JSIM, and a workflow management system, METEOR2. The JSIM simulation environment includes a simulation and database component, while METEOR2 includes the development of advanced workflow models and systems. A genealogy, figure 0.2, of these systems will be useful in subsequent discussions. For a brief description of each of these systems, see the appendix.

The systems at the top of figure 0.2 relied upon textual models for specifying application elements. Declarative specification languages are certainly a big improvement over programming. The Active KDL language elements as well as METEOR's specification languages, WFSL and TSL, are both sophisticated and straightforward to use. Still, humans are very visually oriented. Consequently, all of the newer systems (at the bottom of figure 0.2) rely on diagrammatic models.

0.6.1 METEOR Designer

The METEOR Designer [K+97] is a good example of a hyper-diagrammatic modeling toolkit supporting several modeling perspectives with a rich set of links between them for convenient navigation and browsing. The METEOR Designer [Zhe97, K+97] models five distinct facets of workflows:

1. *Map Designer*. This designer is used to specify the control/data flow from one task/activity to another.
2. *Data Designer*. This designer specifies the structure of and relationships between the data that flows between the tasks.
3. *Task Designer*. This designer specifies the interfaces to actual programs (possibly even legacy systems) which carry out the details of a task's execution.
4. *Interface Designer*. This designer produces a simple default appearance for each user task and then launches the user's favorite HTML editor for page customization.
5. *Exception Designer*. This designer produces a simple default mapping of errors and failures to actions to deal with the problem. The designer facilitates changing/extending this default mapping.

0.6.2 JSIM Modeling Toolkit

The JSIM modeling toolkit consists of two designers: JMODEL and SWORDD.

JMODEL [Zha97] is a graphical designer used to model process-oriented simulations where entities flow in a network of facilities. It provides an easy-to-use, widely accessible graphical model design environment as the front-end

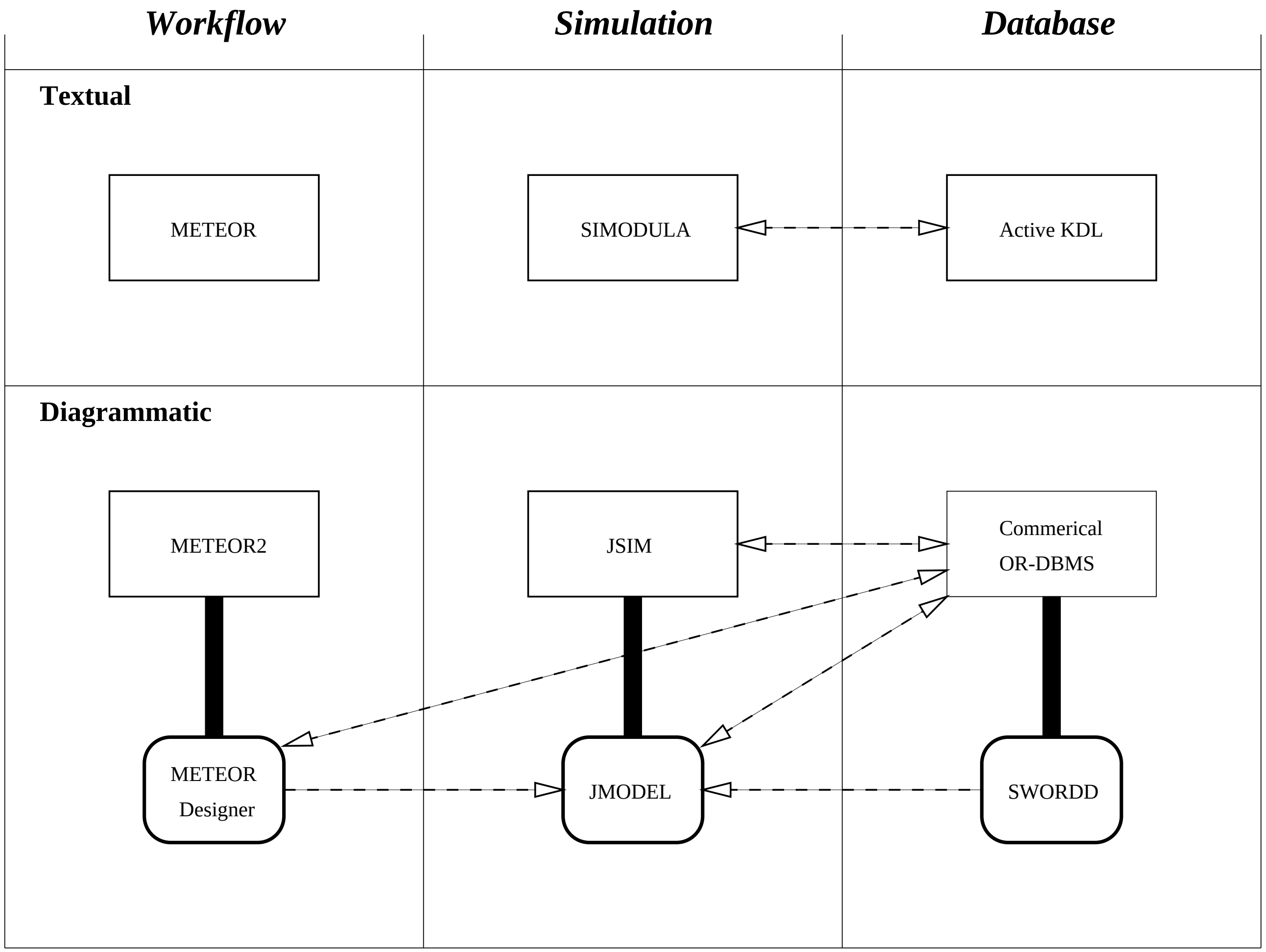

Fig. 0.2. Genealogy of Systems

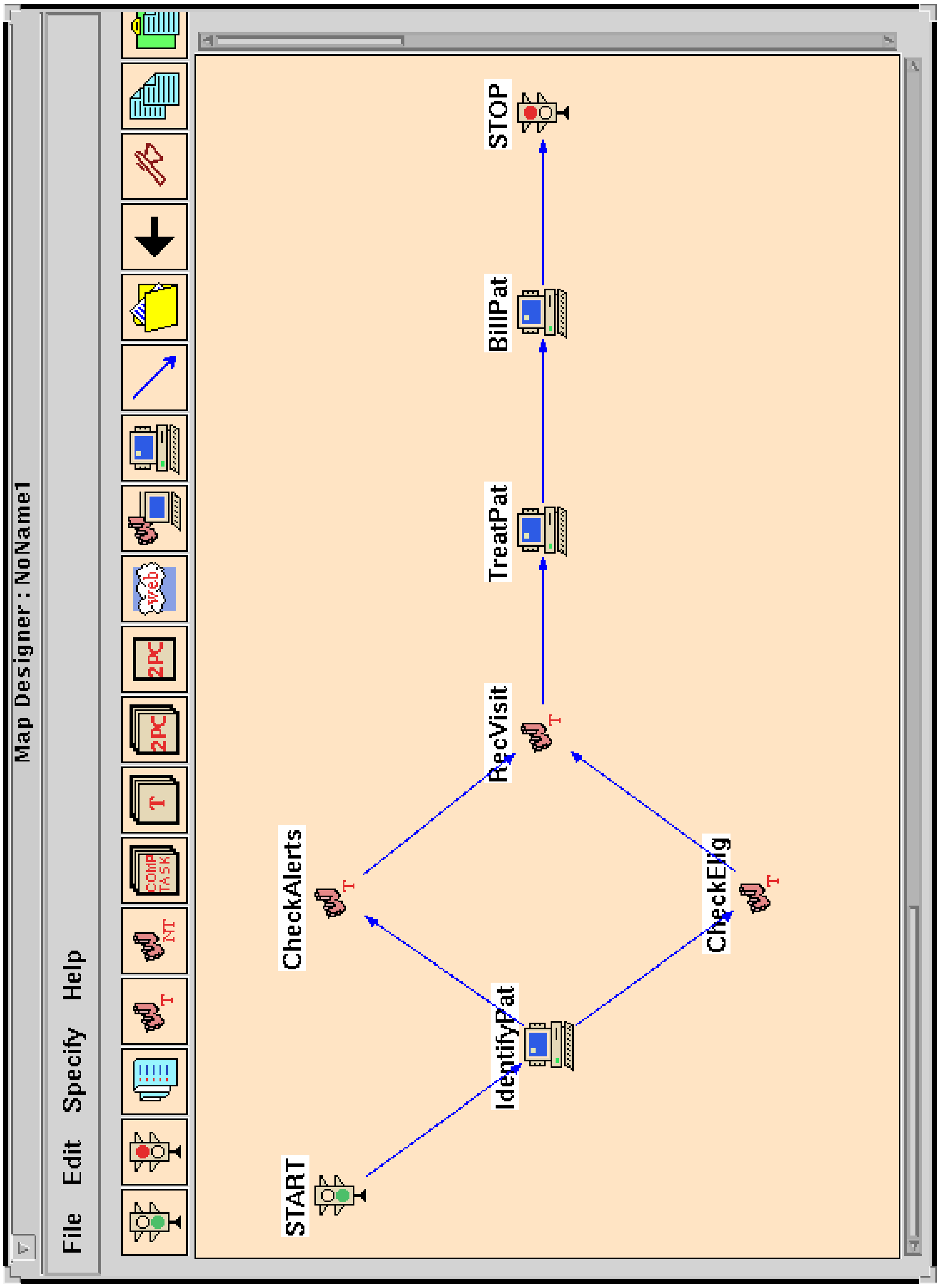

Fig. 0.3. METEOR Designer: Clinic Workflow Model

for the JSIM software package. For wide accessibility, JMODEL is implemented using the Java programming language (as a Java applet) utilizing the Abstract Windowing Toolkit (AWT). For ease of use, it supplies simulationists with a direct, intuitive means to design a simulation model, check and verify the design, and modify and reconstruct the design. The graphical designer provides database connectivity to allow users to be able to store their designed simulation models in a database and browse existing simulation models in the database. A code generator is also implemented, which resides as a back-end to retrieve simulation model from the database upon a user's request, and to generate Java source code based on the designed model to make the model come alive.

JMODEL designs are more general than METEOR designs in the following sense. JMODEL can model applications like the clinic (see the figure showing the clinic simulation model) which are workflow oriented or model systems that do not necessarily have anything to do with workflow (see the figure showing the bus simulation model). Conversely, less information is available from a JMODEL design for implementing a workflow application or prototype. Each have their own distinct purpose and are useful in their own right. Still, there is synergy in using them together and we explore this later in the paper.

An enhancement that is currently being added to JMODEL is support for (De)Composition nodes in the simulation model graph. A Decomposition node decomposes a data object into component parts and send them along parallel transports.

In figure 0.5, the patient data object is decomposed into medical and financial subobjects which are sent in different directions. At a later stage, the patient data is put back together in by a Composition node. This approach has the advantage that it avoids the lost update problem and merge problem inherent with parallel branching.

These types of nodes are based on the data modeling concept of Aggregation. Composition in UML corresponds to strong aggregation. A weaker connection between data objects (or entities) is an Association. Corresponding to this are Association and Disassociation nodes. These are illustrated in the bus model shown in figure 0.6.

SWORDD [LeC97] is a graphical designer used to create data models for modern Object-Relational DBMSs such as Oracle 8 or Informix Universal Server. It provides two distinct advantages over forthcoming commercial products such as the Oracle Database Designer for Oracle 8. First, it is Java-based so it provides wider availability, and second, hooks placed in the Java source code facilitate its interoperability with JMODEL, thus supporting the hyper-diagrammatic capabilities of the overall JSIM modeling toolkit.

SWORDD is used to design several types data models for JSIM: scenario (input parameter) definitions, simulation result (output parameter) defini-

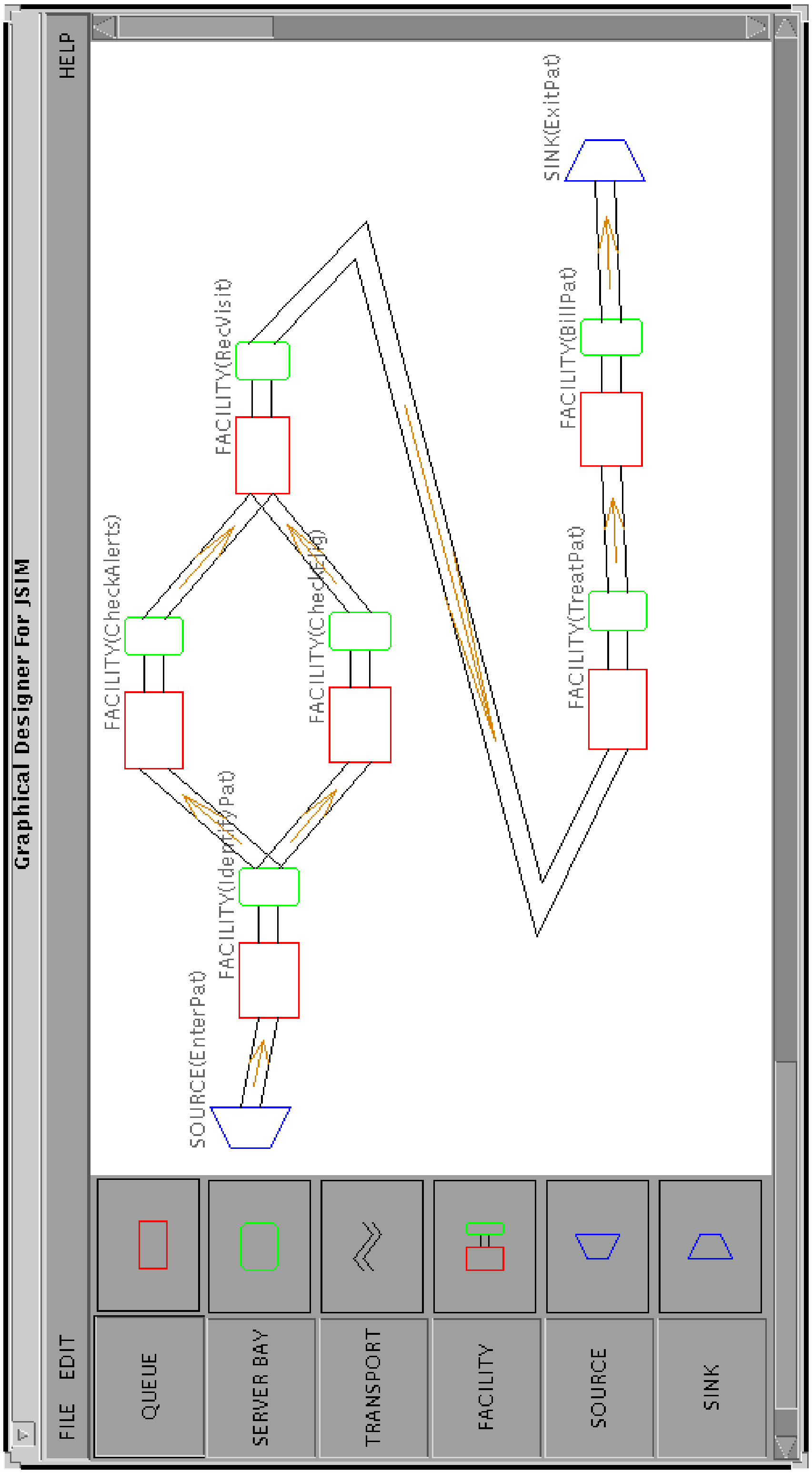

Fig. 0.4. JMODEL: Clinic Simulation Model

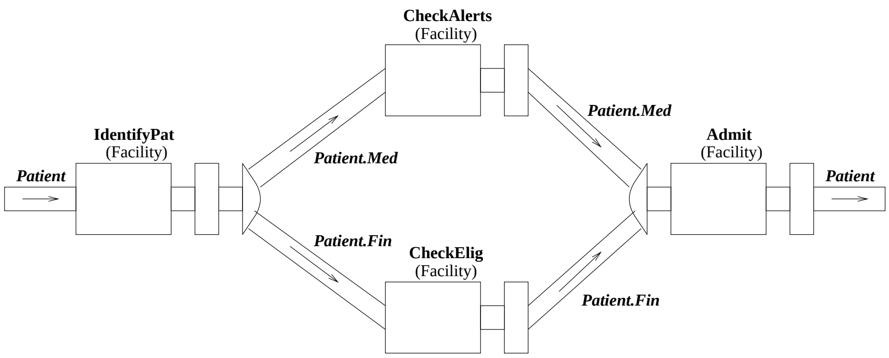

Fig. 0.5. (De)Composition Nodes

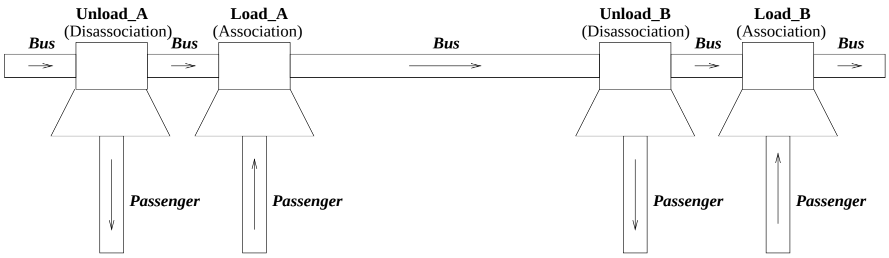

Fig. 0.6. (Dis)Association Nodes

tions, model definitions (design graph) and data objects (data flowing along transports).

The data design in figure 0.7 illustrates an interesting interaction between a JMODEL simulation model and SWORDD data model. When the bus arrives at the loading zone of the bus stop, an association between passengers and the bus will be established. Because of the cardinality constraint on the association given in the data model, unless there are at least 10 passengers waiting, the bus will wait. Similarly, if more than 30 passengers are waiting only 30 will be taken.

0.6.3 Synergy between METEOR and JSIM

METEOR2 and JSIM provide synergy in two directions: (1) After a workflow has been designed, a simulation model can be created for it. Studying the performance profiles of the workflow design using simulations will provide feedback to the design process for improving the workflow design before developing or deploying the workflow. (2) Once a workflow is deployed, performance data can be collected to refine or validate the simulation model. A validated simulation model will be very useful for future adaptations of the workflow [MSK+95].

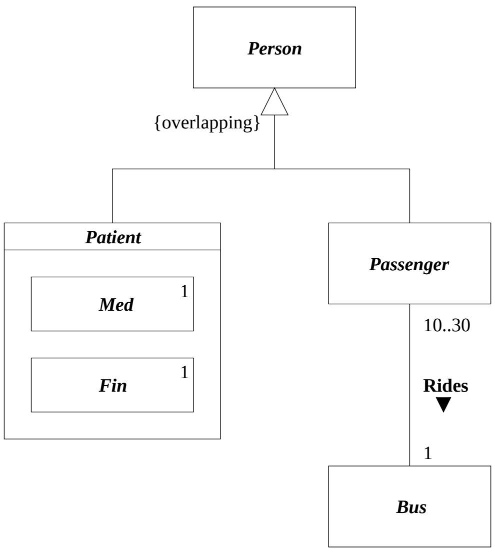

Fig. 0.7. Data Design in UML

There are similarities between the METEOR map designer and JMODEL. Both conceptualize entities flowing through a network of nodes. Workflow is concerned with scheduling and transformations that take place in tasks, while simulation is mainly concerned with system performance. Similarly, SWORDD and the METEOR data designer have commonality. SWORDD is used to model schema for today's Object-Relational DBMSs, while the METEOR data designer is used to model data objects sent between tasks.

0.7 Summary

Simulation, database and workflow modeling are distinct areas which overlap and provide synergy when combined together. While simulation and workflow modeling are more dynamic, database modeling has traditionally been more static. Simulation models focus on performance and behavior, while workflow models focus more on use and high-level implementation aspects. Time is essential in simulation although it is a simulated time, while time may be absent in workflow models or used in soft real-time scheduling constraints. Often simulation and workflow models are represented as directed graphs and the nodes in both types of graphs may fall into correspondence, however, the types of nodes are usually different (e.g., Queue, Server and Facility for simulation versus HumanComputer, Transactional and NonTransactional Tasks for workflow). Finally, this paper has given two examples, METEOR Designer and JSIM Modeling Toolkit, of how hyper-diagrams can be used to link these multiple modeling aspects.

Appendix

- Simulation: *SIMODULA* [MW89]. SIMODULA is a simple simulation system based on the process-interaction simulation world view. A simulation environment was developed to enhance the processes of developing and running models. Because of the complex nature of simulation data and model specifications, relational databases were not ideal, so a toy database system, called OBJECTR, was developed. This was a relational database system with simple object-oriented extensions. It was later replaced with a more sophisticated database system, Active KDL.
- Database: *Active KDL* [PT88, PTE89, MPKW90, MKP+91, MPK+91, PBMK92]. Active KDL is an Object-Oriented Database Management System providing methods, constraints and triggers as well as active objects. All schema elements are specified using a high-level functional style. Active KDL has both a query language (QL) and a database programming language (DBPL). The DBPL is sophisticated enough to even express simple simulation models entirely within the language. Active KDL evolved from the Knowledge Data Model (KDM) which is a hyper-semantic data model [PT88].
- Workflow: *METEOR* [SR93, GHS95, JNRS93, KS95, RS95]. The METEOR Workflow Management System (WfMS) includes a centralized workflow enactment service developed to control task execution from specifications given in the Workflow Specification Language (WFSL) and Task Specification Language (TSL).
- Simulation: *JSIM* [NMZ96, MNZZ97]. JSIM is a redevelopment of SIMODULA providing higher-level facilities as well as animation. It also has the advantage that JSIM models can run over the Web. At this time, quality object-relational database systems are commercially available and suitable for the job of storing simulation data and models (e.g., Oracle 8 and Informix Universal Server). *JMODEL* makes it substantially easier to produce simulation models.
- Database: *Commercial ORDBMS*. Modern Object-Relational DBMSs provide a comprehensive set of useful capabilities, fully adequate for both JSIM and METEOR. Graphical designers such as Oracle Database Designer for Oracle 8 or *SWORDD* simply schema production.
- Workflow: *METEOR2 (WebWork and OrbWork)* [SKM+96, MSKW96, MPS+97, K+97]. The METEOR2 project in the LSDIS Lab is focused on developing state-of-the-art workflow models (*METEOR Designer*) and enactment systems (WebWork and OrbWork). WebWork is a purely Web-based enactment service featuring ease of development and deployment as well as transactional and recovery capabilities. OrbWork is a CORBA-based cousin of WebWork which supports advanced transactional and dynamic capabilities. Both WebWork and OrbWork utilize the outputs of the METEOR Designer for generating workflow applications.

References

[Bir79] G.M. Birtwistle. *Discrete Event Modeling in SIMULA.* MacMillian, London, England, 1979.

[Che76] P. Chen. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), 1976.

[GHS95] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–154, April 1995.

[HM81] M. Hammer and D. McLeod. Database Description with SDM: A Semantic Data Model. *ACM Transactions on Database Systems*, 6(3), September 1981.

[JNRS93] W. Jin, L. Ness, M. Rusinkiewicz, and A. Sheth. Concurrency Control and Recovery of Multidatabase Work Flows in Telecommunication Applications. In *Proc. of ACM SIGMOD Conference*, May 1993.

[K+97] K.J. Kochut et al. OrbWork: A Distributed, Dynamic Workflow Enactment Service for METEOR_2. Technical report, LSDIS Lab, University of Georgia, 1997. URL: http://lsdis.cs.uga.edu.

[KP76] L. Kerschberg and J. Pacheco. A Functional Data Base Model. Technical report, Portificia Univ. Catolica do Rio De Janeiro, February 1976.

[KS95] N. Krishnakumar and A. Sheth. Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations. *Distributed and Parallel Databases*, 3(2):155–186, April 1995.

[LeC97] L. LeConte. SWORDD: A Simple Widely-Available Object-Relational Database Design-Tool. Master's thesis, University of Georgia, Athens, GA, June 1997.

[MKP+91] J.A. Miller, K.J. Kochut, W.D. Potter, E. Ucar, and A. Keskin. Query-Driven Simulation Using Active KDL: A Functional Object-Oriented Database System. *Int. Journal in Computer Simulation*, 1(1):1–30, 1991.

[MNZZ97] J.A. Miller, R. Nair, Z. Zhang, and H. Zhao. JSIM: A Java-Based Simulation and Animation Environment. In *Proceedings of the 30th Annual Simulation Symposium*, pages 786–793, Atlanta, Georgia, April 1997.

[MPK+91] J.A. Miller, W.D. Potter, K.J. Kochut, A. Keskin, and E. Ucar. The Active KDL Object-Oriented Database System and Its Application to Simulation Support. *Journal of Object-Oriented Programming, Special Issue on Databases*, 4(4):30–45, July-August 1991.

[MPKW90] J.A. Miller, W.D. Potter, K.J. Kochut, and O.R. Weyrich. "Model Instantiation for Query Driven Simulation in Active KDL". In *Proceedings of the 23rd Annual Simulation Symposium*, pages 15–32, Nashville, Tennessee, April 1990.

[MPS+97] J.A. Miller, D. Palaniswami, A.P. Sheth, K.J. Kochut, and H. Singh. WebWork: METEOR_2's Web-Based Workflow Management System. *Journal of Intelligent Information Systems*, 1997. (to appear).

[MSK+95] J.A. Miller, A.P. Sheth, K.J. Kochut, X. Wang, and A. Murugan. Simulation Modeling Within Workflow Technology. In *Proc. of the 1995 Winter Simulation Conference*, Arlington, VA, December 1995.

[MSKW96] J. A. Miller, A. P. Sheth, K. J. Kochut, and X. Wang. CORBA-based Run-Time Architectures for Workflow Management Systems. *Journal of Database Management*, 7(1):16–27, Winter 1996.

[MW89] J.A. Miller and O.R. Weyrich. Query Driven Simulation Using SIMODULA. In *Proceedings of the 22nd Annual Simulation Symposium*, pages 167–181, Tampa, Florida, March 1989.

[NMZ96] R. Nair, J.A. Miller, and Z. Zhang. A Java-Based Query Driven Simulation Environment. In *Proceedings of the 1996 Winter Simulation Conference*, pages 786–793, Coronado, California, December 1996.

[PBMK92] W.D. Potter, T.A. Byrd, J.A. Miller, and K.J. Kochut. Extending Decision Support Systems: The Integration of Data, Knowledge, and Model Management. *Annals of Operations Research, Special Issue on Model Management*, 38:501–527, 1992.

[Pet77] J.L. Peterson. Petri nets. *ACM Computing Surveys*, 9:223–252, 1977.

[PH91] R.J. Pooley and P.M. Hughes. Towards a Standard for Hierarchical Process Oriented Discrete Event Simulation Diagrams. *Transactions of the Society for Computer Simulation*, 8:1–20, 1991.

[Pid92] M. Pidd. *Computer Simulation in Management Science*. Wiley, Chichester, England, 3rd edition, 1992.

[Pri79] A.A.B. Pritsker. *Introduction to Simulation with SLAM*. Wiley, New York, NY, 1979.

[PSS90] C.D. Pegden, R.E. Shannon, and R.P. Sadowski. *Introduction to Simulation Using SIMAN*. McGraw-Hill, NY, 1990.

[PT88] W.D. Potter and R.P. Trueblood. Traditional, Semantic and Hyper-Semantic Approaches to Data Modeling. *IEEE Computer*, 21(6), June 1988.

[PTE89] W.D. Potter, R.P. Trueblood, and C.M. Eastman. Hyper-Semantic Data Modeling. *Data and Knowledge Engineering*, 4(1), July 1989.

[R+91] J. Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, NJ, 1991.

[RS95] M. Rusinkiewicz and A. Sheth. Specification and Execution of Transactional Workflows. In W. Kim, editor, *Modern Database Systems: The Object Model, Interoperability and Beyond*. ACM Press, New York, NY, 1995.

[S+97] Sheth et al. A Taxonomy of Workflow Management Systems. Technical report, LSDIS, University of Georgia, 1997.

[Sch74] T.J. Schriber. *Simulation Using GPSS*. Wiley, NY, 1974.

[Sch83] L. Schruben. Simulation Modeling with Event Graphs. *Communications of the ACM*, 26:957–963, 1983.

[SKM+96] A.P. Sheth, K.J. Kochut, J.A. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch, and I. Shevchenko. Supporting State-Wide Immunization Tracking using Multi-Paradigm Workflow Technology. In *Proc. of the 22nd. Intnl. Conference on Very Large Data Bases*, Bombay, India, September 1996.

[Sof97] Rational Software. UML Notation Guide, version 1.1. Technical report, Rational Software, September 1997.

[SR93] A. Sheth and M. Rusinkiewicz. On Transactional Workflows. *IEEE Computer*, June 1993.

[Zha97] H. Zhao. A Graphical Designer for JSIM. Master's thesis, University of Georgia, Athens, GA, September 1997.

[Zhe97] K. Zheng. Development of the METEOR_2 Designer. Master's thesis, University of Georgia, Athens, GA, June 1997.