USING QUERY-DRIVEN SIMULATIONS FOR QUERYING OUTCOMES OF BUSINESS PROCESSES

P. Balasubramanian

Alexander Tuzhilin

Information Systems Department
Stern School of Business
New York University

Working Paper Series
STERN IS-93-43

Using Query-Driven Simulations for Querying Outcomes of Business Processes

P. Balasubramanian Alexander Tuzhilin [*]

Information Systems Department
Leonard N. Stern School of Business
New York University

Abstract

When decision makers want to know outcomes of business processes in their organizations, they often use simulations to do this. This paper describes how a new ***Query-Driven Simulation (QDS)*** approach can be used by decision makers to obtain information about future outcomes of business processes in a more declarative, flexible, and interactive way than the traditional approach of running simulations and then gathering statistics about simulation outcomes. The paper also describes the types of questions decision makers ask about outcomes of business processes and studies how easy it is to express these questions in terms of an SQL-like query language SimQL designed for Query-Driven Simulations. It also identifies the types of applications that are especially well-suited for QDS. Finally, the paper describes the Query-Driven Simulation Modeling Lifecycle and how QDS provides a feedback loop in the model development process.

KEY WORDS: Decision Support, Query-Driven Simulations, Discrete-Event Simulations, Databases, Query Languages, Modeling Lifecycle.

1 Introduction

Throughout their daily activities, decision makers ask different questions about future outcomes of business processes running within their organizations. The information they obtain by asking these questions helps them make planning, control, and staffing decisions. For example, a foreman in a manufacturing facility may want to know how many parts will be produced within the next shift, so that he or she can notify the distribution department beforehand, and the distribution department can determine how many trucks will be needed to carry these parts. In a mail order company, a salesperson may want to know if it is possible to deliver a customer order within 2 weeks. The salesperson can use this information to promise the customer the delivery date that is earlier than the regular delivery date under the current 3-to-4 weeks delivery policy. In the banking industry, a

[*]Address: 44 West 4th Street, Room 9-78, New York, NY 10012, e-mail: atuzhili@rnd.stern.nyu.edu

manager may want to know how many tellers will the new branch need so that appropriate hiring decisions can be made.

The problem of predicting the future outcomes of business processes has certainly been addressed before, mainly within the framework of Operations Research and Management Science. In particular, mathematical models of business processes are built by model developers who also try to find analytical solutions to these models. If an analytical solution can be found, then the behavior of the model is well understood. However, in complex industrial and organizational settings it is often difficult to find analytical solutions to these models. Hence we have to resort to techniques like simulation to capture the inherent complexity of such systems [30].

To answer questions such as the ones presented above about simulation models, these models are executed many times and summary statistics about individual runs are collected in one of the following two ways. In the first approach, summary statistics are computed inside the simulation program, and the program produces the final results to the user. In the second approach, various simulated events are recorded in the *trace files*, and then statistics are collected from these trace files by either writing programs in one of the programming languages, such as Fortran or C, or by using one of the statistical packages, such as SAS [29]. In both cases, if a user wants to find some additional summary statistics about the model not *a priori* produced by the model, he or she has to know simulation languages, programming languages and/or statistical packages in order to make appropriate changes to the programs and be able to gather statistics.

Since most of the people who ask questions about future outcomes of business processes in their organizations, such as a foreman, a salesman, or a bank manager, do not know much about simulations, programming languages, or statistical packages, they cannot ask ad-hoc questions about future outcomes of their business processes as the questions arise "on-the-fly." Instead, they have to rely on the systems developed by information systems departments that support a fixed set of "canned" questions. Clearly, this situation is unsatisfactory in many organizations, such as manufacturing, transportation, or in the military, where various users want to ask many different questions about simulation outcomes of various models.

To address this problem, the idea of *Query-Driven Simulations* was proposed in [24, 36] and was subsequently developed in [3]. *Query-Driven Simulations (QDS)* is an approach to simulations in which the user first asks queries about outcomes of simulations and then, depending on the query, different simulations are launched in order to answer it. After simulations are finished, the query expressed by the user is evaluated on the trace files generated by simulations. For example, assume a foreman wants to know how many parts will be manufactured within the next 8 hours based on the manufacturing model *Manufact-Model-2* of the plant. According to the QDS approach, it should be determined first what events in the *Manufact-Model-2* should be traced and for how long in order to answer the query (i.e. the event *Finished(Part, Time)* should be traced for 8 hours of simulated time). Then simulations are launched for that amount of time and designated events (i.e. Finished(Part,Time)) are recorded in the trace files that are stored in a temporal database [33]. The query is then evaluated on these trace files.

In [3], we describe an architecture, and an implementation of a specific QDS system, called *Cassandra*$^+$. We also present the query language *SimQL* for asking questions about simulation outcomes. The main feature of the Cassandra$^+$ architecture and the language SimQL is that *any* temporal relational query language, such as TQuel [32] or TSQL [25], can be embedded in SimQL, and that SimQL queries can be asked about simulation models written in *any* simulation language assuming that the language can produce trace files having a certain format. We also argue in [3] that the QDS approach provides a more declarative, flexible, and interactive way of obtaining information about outcomes of business processes than the traditional approach of launching simulations and gathering statistics since it eliminates the problems associated with the latter approach described above.

In this paper, we describe in which applications, how, and by whom QDS systems, and Cassandra$^+$ in particular, can be used. We will start with the analysis of the types of questions decision makers typically ask about outcomes of business processes in Section 2. In Section 3, we explain how Cassandra$^+$ can be used to answer these questions. In Section 4, we compare the QDS approach with the traditional approach to simulations. In Section 5, we describe the types of applications where QDS systems are the most useful. Finally, in Section 6 we describe the Query-Driven Modeling Lifecycle that specifies the process of development, installation, usage, and modification of simulation models based on the QDS approach.

2 Types of Questions Decision Makers Ask About Outcomes of Business Processes

The types of questions decision makers ask about outcomes of business processes vary significantly across different industries, organizations within an industry, functional units within an organization, and levels of management within a functional unit. These questions also vary depending on whether the question is about possible outcomes of business processes that are currently running in real-time (e.g., how many parts will be manufactured in the next shift?) or about some hypothetical business process (e.g., what will the average cost of producing new product be?). We will call the first type of questions *on-line* or *real-time*, and the second one we will call *off-line*. Furthermore, questions can vary depending on whether they are asked about the present state of the system, about the past, or about the future state of the system. Questions about the future, can be of *predictive* or *evaluative* nature (e.g., could we have fulfilled Order-5 (assuming it was not) if we had used FIFO queuing discipline as opposed to the LIFO discipline we used over the last month).

In this section, we consider a manufacturing organization and describe the types of questions decision makers within such an organization may want to ask about the *future* outcomes of various processes in this organization. We choose a manufacturing example for the following reasons. First, there has been much interest developed in the last several years in applying information technologies for solving manufacturing problems [28, 2, 27]. Second, simulations are widely used in manufacturing [21, 8, 20] thus making it a good application for testing our ideas. Finally, as we show later in this section, various people in a manufacturing organization want to ask many different questions about future outcomes of manufacturing processes. Also we decided to consider questions only about the future because these questions are of great interest to decision makers and because answering these questions requires running simulations.

We divide the questions manufacturing people want to ask about outcomes of manufacturing activities into several groups according to the standard classification of [15] as presented in Figure 1. As Figure 1 shows, the decision makers are grouped based on the functional units they belong to and based on the level in the organizational structure to which they belong.

For each group of people in Figure 1, we describe the questions about outcomes of business processes in which they are mostly interested. These questions were obtained from the literature [6, 34, 31, 35], and through extensive discussions with manufacturing professionals [11, 38, 26, 18]. We assume that the questions presented below are real-time predictive questions, unless stated otherwise.

Cell-1: In this class of queries, decision makers are interested in asking questions about production targets, quality assurance, job priority assignment, machine utilizations, raw-material availability, set-up time, work-flow co-ordination, and so on. Below, we provide examples of some of the typical questions different operations people may want to ask.

Foreman - How many jobs will be produced in the next shift?

Foreman - Which machines will have an utilization ratio of more than 90% during the next shift?

Foreman - What will be total set-up time when we change from process 1 to process 2?

Manager - When should component-1 be delivered to station-A so as to complete assembly

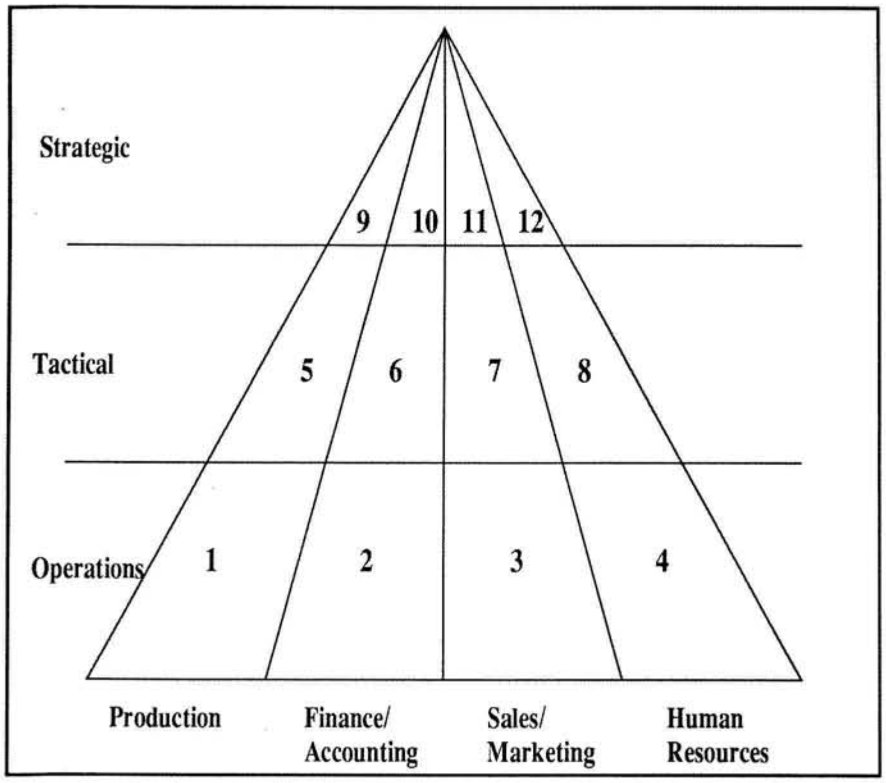

Figure 1: A Classification of Questions Arising Within Organizations

of unit PJ-352 on time?

Foreman - Where will job-5 be 5 hours from now?

Manager - Will job-5 be completed if we give it the highest priority?

Quality Control Engineer - How many jobs will be rejected in the next batch?

Foreman - What is the expected transit time for a part in the plant?

Foreman - What will be the scrap rate per machine in the next shift?

Cell-2: These queries are asked to help decision makers in finance department manage their cash flow and for accountants to see if the manufacturing costs are under control and to ascertain costs of manufacturing new products. Some typical questions finance and accounting managers may want to ask are:

Finance Manager - How many jobs will be produced this week, next week and week after that?

Accountant - What will be the variation from the standard costs of the product per machine during the next shift?

Accountant - What will be the material costs for order-5 during the next shift?
Accountant - What will be the labor cost for the next shift?
Finance Manger - What will be the average cost of producing jobs of type A? (off line)

Cell-3: These kinds of questions are used to help the salesman to answer questions about customer order processing, delivery speed-up, etc. These questions will also help a salesman determine if a particular customer request can be met on time, track the status of a customer order and to see which customer requests can be met. The marketing manager at the operations level is interested in manufacturing and delivery times of products and flexibility in delivery schedules. Some typical questions sales and marketing managers may want to ask are:

Salesman - Is it possible to complete job-5 in 8 hours?
Salesman - What will be the cost of fulfilling the Order-5 if it is completed using the least priority?
Salesman - Could we have fulfilled Order-5 (assuming it was not) if we had used FIFO queuing discipline instead of LIFO discipline that we used over the last month? (evaluative)
Brand manager - Is it possible to fulfill orders for a new product within the next two months?
Salesman - How long will it take to complete Order-5 if we add optional feature-1?
Salesman - If job-5 is assigned a different priority, which are the jobs that will not be completed in the given time but would have been completed under the old priority?
Brand manager - If the actual demand for 1 liter bottles of the soft drink A is 7% larger than the forecasted demand for these bottles, then how fast will the manufacturing department be able to meet this increased demand?

Cell-4: The queries listed below help human resources department make staffing decisions. This will ensure that manpower is available to meet order requirements. Some typical questions human resource managers may want to ask are:

Manager - What will be the average output of jobs if we vary the number of shifts from 1 through 3? (off-line)
Manager - Given that employee-A is not available for the rest of the week, can the work be routed to employees C and D?
Personnel Manager - To complete order-20 by the 30^{th} of this month what is the total overtime that will be granted?
Personnel Manger - How many new employees do we need to hire in the next two weeks for the new project?

Cell-5: Production middle managers may want to ask questions about future outcomes of manufacturing processes to make better production planning decisions. These questions help to ensure that production, utilization and various other targets are met. Capacity planning, workflow redesign and coordination decisions are also addressed by these questions. Some typical questions production middle managers may want to ask are:

Manager - What will be the machine utilization ratios for the various plants for the next shift?

Manager - What should the lot size be to satisfy our future demand profile at the minimum cost? (optimization)

Manager - Which of the four scheduling heuristics will help meet the target production of 100 jobs in 8 hours?

Plant Manager - How will throughput be affected if I rearrange my factory?

Plant Manager - Can I reduce my Work-In-Process (WIP) by reducing set-up times?

Manager - Among the four workshop configurations which one delivers the maximum throughput? (off-line)

Manager - What will be the yield per plant for the next quarter for Product-C?

Manager - How many additional machines need to be purchased if we change the production process?

Cell-6: Financial and accounting middle managers ask the questions in this group in order to make budgeting decisions and ensuring that production costs are under control. They can also be used for performing sensitivity analysis of costs. Some typical questions middle managers in finance and accounting may want to ask are:

Finance Manager - What will be the maintenance cost for Plan-A for the next 6 months?

Finance Manager - If we had done only preventive maintenance what would have been the total manufacturing costs?

Accountant - What will be the variance in production costs and production times for product-A between the plants in the east coast and the ones in the west?

Accountant - Is the total cost of producing product-A more sensitive to transportation costs, production costs, or tariffs?

Finance Manager - What will be the average cost of manufacturing product-A per plant for the next quarter? (off-line)

Cell-7: Marketing managers are interested to know how much time it will take to manufacture new products or modify existing products to meet customer demand. They are also interested in the promotion planning questions. Some typical questions these managers may want to ask are:

Marketing Research Manager - Which promotional campaign will generate maximum sales for Brand A in the eastern region?
Product Manager - How much time will it take to bring the prototype of a new product to the market (set up a manufacturing line for it and move it into production)?
Product Manager - How much time will it take to add this new feature to Brand A and move it into production?
Marketing Research Manager - Can the new product be produced in size X using existing technology and at the current cost?
Product Manager – Should we change our marketing strategy for Product-A so that it appeals to the younger generation?
Sales Manager – Which incentive scheme for the sales force will result in a 10% increase in total sales?

Cell-8: Just like quadrant 4 such questions are helpful in making staffing and training decisions. In this case however, equipment purchase decisions are made along with staffing decisions. This quadrant should also handle questions that help negotiate pay scales with unions.

Manager - Is our manpower sufficient to meet the production schedule for the next quarter?
Manager - How many employees have to be trained in the new methodology?
Manager - If we had fixed the pay rates of Class A employees at $X per hour for 2 years and new recruits at $Y per hour, would we have made better return on investment for plant A? (evaluative)
Manager - How many people experienced in programming CNC machines have to be recruited?

Cell-9: Senior management in operations typically asks questions about operations that have a long-term impact on the company, such as business process re-engineering or make-vs-buy questions.

Manager (of a leading PC manufacturer) - Is it better to manufacture a new microprocessor in house or buy it from a leading chip manufacturer?

Manager - Will automation of manufacturing process C increase productivity?

Manager - What will happen if I re-engineer my workflow?

Manager - If we completely automate workstation-A how will it improve the overall performance of our shop?

Manager - Is it worth investing in the new machine or should we buy more of the old machine?

Cell-10: Senior management in Finance and Accounting asks budgeting and process redesign questions having long-term impact on the organization.

Accountant – What does my invoicing process really look like, and where can improvements be made to minimize processing delays in our new plant?

Accountant – What will the break-up of expenses be when our new plant is opened in Latin America in 6 months?

Finance Manager – In which plants can budget cuts be enforced to reduce total production cost by 10%?

Finance Manager – How will NAFTA affect our revenues and profits?

Cell-11: Examples of questions asked by senior sales and marketing managers are long term budget allocation and market share estimation questions.

Marketing VP – How much money should be assigned to advertising?

Marketing VP - How will my distribution system behave with 25% fewer employees?

Marketing VP – What will be our total sales in the Asian markets for the next year?

Marketing VP – How can we arrest the decline in sales in the domestic market?

Cell-12: A Senior Human Resources Manager asks questions that help him or her make long-term staffing, training, and promotion decisions.

Human Resource Director – How drastically will the shortage of qualified personnel affect the output in our Latin American Plant?

Human Resource Director – What will be the total training cost for running the Mexican Plant?

Human Resource Director – What should the hiring pattern be for next year?

As we showed in this section, various people across different functional units and across various levels of management in a manufacturing organization are interested in many questions about outcomes of their business processes.

As we pointed out in the introduction already, the need to answer these questions was addressed before within the framework of Operations Research and Management Science by developing mathematical models of manufacturing processes and solving these models. Whenever analytical solutions could not be found, modelers resorted to simulations. We also pointed out in the introduction that the traditional simulate-and-gather-statistics approach has problems since it requires the end-users either to know simulations, statistical, and programming systems, or use the "turn-key" software developed by simulation specialists allowing the end-users to ask only a fixed set of "canned" queries.

In the next section, we briefly describe the Query-Driven Simulation system Cassandra$^+$ and its query language SimQL that lets the users formulate and answer questions about outcomes of simulation models. We will also show how Cassandra$^+$ can answer some of the questions presented in this section.

3 How Users Can Get Answers About Outcomes of Business Processes Using Cassandra$^+$

In this section, we describe how end-users can get answers to questions similar to the ones described in Section 2. In order to explain this, we first describe the query language SimQL in which some of these queries can be expressed. We also describe how SimQL is supported by the Query-Driven Simulation system Cassandra$^+$.

3.1 Language SimQL

Since queries can be asked about various simulation models supporting different kinds of business processes in an organization, there is a need to store these models in a *modelbase* [5] and allow the query language to ask queries against this modelbase. For example, assume that a manufacturing organization has three manufacturing plants, PL1, PL2, and PL3 that have different manufacturing processes. Correspondingly, we need three different models, Mfc-Model-1, Mfc-Model-2, and Mfc-Model-3, that model manufacturing processes in plants PL1, PL2, and PL3 respectively. We assume that these models are stored in a modelbase which is the central repository of all the information about all the simulation models maintained by the organization. Besides the models themselves, the modelbase contains additional information about the features of these models, such as the name of the simulation language in which a model is written, default simulation parameters for the model, etc., and information needed to interface the query and simulation languages, such as the events that the model can trace and whether real-time queries are allowed on the model.

We present an informal overview of SimQL based on the three-plant example. The formal description of the language can be found in [3].

Example 1 Assume a foreman from plant PL-3 wants to know how many parts will be manufactured within the next 8 hours at that plant (question 1 for Cell 1 in Section 2).

This query can be formulated in SimQL as:

Time:	8 hours
Answer-Semantics:	Numeric
Core-query:	
SELECT	COUNT(PART)
FROM	FINISHED
Model-Name:	Mfc-Model-3
Error-of-Estimation:	10
Confidence-Coefficient:	95

where FINISHED(PART,TIME) is a temporal relation specifying which parts are finished at what time.

This query consists of two parts: the *core-query* and the *shell* consisting of a set of parameters providing directives to Cassandra$^+$ on how the core query has to be integrated with the simulation module.

The *core-query* expresses the actual question the end-user asks and is represented in this example with an SQL statement

SELECT COUNT(PART)
FROM FINISHED

The *shell parameters* in this example are **Time**, **Answer-Semantics**, **Model-Name**, **Error-of-Estimation**, and **Confidence-Coefficient**. The **Model-Name** parameter specifies the name of the simulation model against which the query is asked (Mfc-Model-3 in our example). The **Time** specifies for how long simulations have to be run (8 hours). The **Answer-Semantics** parameter specifies if the query returns a number as the answer (e.g. the number of parts produced) or a relation (as in Example 2). Depending on this parameter, Cassandra$^+$ returns different answers, as will be explained below. Finally, **Error-of-Estimation** and **Confidence-Coefficient** parameters specify the estimation error of the answer. A typical answer in case the **Answer-Semantics** parameter is *Numeric* would be:

> The average number of parts produced within the next 8 hours is 32 ± 3, and we can make this statement with confidence 95%.

In this example, only five shell parameters are used. However, SimQL supports other parameters as well. This means that the other parameters were given *default* values by Cassandra$^+$. The description of other SimQL parameters can be found in [3].

□

Cassandra$^+$ processes a SimQL query as follows. When the user issues a SimQL query, Cassandra$^+$ determines the simulation model to which the query refers, determines how many simulation runs N are needed to obtain the answer within the estimation bounds specified by the user, runs this simulation model for N simulation runs, storing simulation traces in trace files, converts the resulting simulation trace files into the temporal database format, issues the temporal (core) query against each simulation trace, and statistically analyzes the answers to these queries. The details of this query processing strategy are described in [3]. As follows from this explanation, the architecture of Cassandra$^+$ consists of two components: the simulation and the querying components. The task of the Cassandra$^+$ architecture is to "interface" the querying component with the simulation component.

In Example 1, we used SQL with timestamps as a query language. However as we explain in [3], *any* other temporal relational query language, such as TQuel [32], TSQL [25], or temporal logic calculus [23, 19], can be used instead of SQL as a *core*-query language. Furthermore, simulation models in the modelbase can be written in *any* simulation language, as long as they generate the trace files in a certain relational form described in [3]. This means that the query-driven simulation system Cassandra$^+$ allows end-users to ask queries in the language of their choice against simulation models written in any simulation language.

Example 2 Consider another question (question 2 from Cell-1 in Section 2) that a foreman from plant PL-2 may want to ask:

> Which machines will have a utilization ratio of more than 90% during the next 10 hours?

Assume that the temporal relation MACHINE describes the status of the machines. It has the form MACHINE(ID,STATUS,FROM,TO) specifying that a certain machine has a certain status from time FROM until time TO. Then this query can be expressed in SimQL as[1]

Time:	10 hours
Answer-Semantics:	Relational
Core-query:	
SELECT	ID
FROM	MACHINE
WHERE	STATUS = 'busy' AND $\$NOW <$ FROM AND TO $< \$NOW +$ 10hours
GROUP-BY	ID
HAVING	SUM(TO − FROM) > 90% $*$ 10hours
Model-Name:	Mfc-Model-2
Error-of-Estimation:	10
Confidence-Coefficient:	95

This query returns a *relation* as the answer, i.e. a list of the machines. Therefore, the value of the **Answer-Semantics** parameter is *Relational.* A typical answer for this query is

> The most likely answer for this query is { *Machine-23, Machine-12, Machine-17* }, and it is returned with probability 36% ± 3% and confidence level of 95%.

Note that the answer for the SimQL query with the "relational" value of the **Answer-Semantics** parameter is different from the query with the "numeric" value of this parameter. In the former case, the query returns the most likely relation as the answer and the estimation of its *probability.* In the latter case, the query returns the average value of the answer and the estimate of its *value.*

□

In summary, we provided a short overview of the general purpose temporal query language SimQL that allows users to ask queries about simulation outcomes of various business processes. The language is designed so that core queries can be expressed in *any* temporal relational query language against models written in *any* simulation language. Among other important features of SimQL, not discussed in this paper, is the support for queries about the *past*, support for the *off-line* queries (discussed in Section 2), support for the tuple semantics of answers (alternate type of answer the user can ask for) and the support for *experimental design* (when the user can ask a query about a range of models with different values of parameters). Description of these features and their implementation in Cassandra$^+$ can be found in [3].

[1]We simplified this query a little assuming that at the present time, $\$NOW$, and at the end of the time interval, $\$NOW + 10$, all machines are idle. We can remove this restriction, but the query will be much more cumbersome in this case.

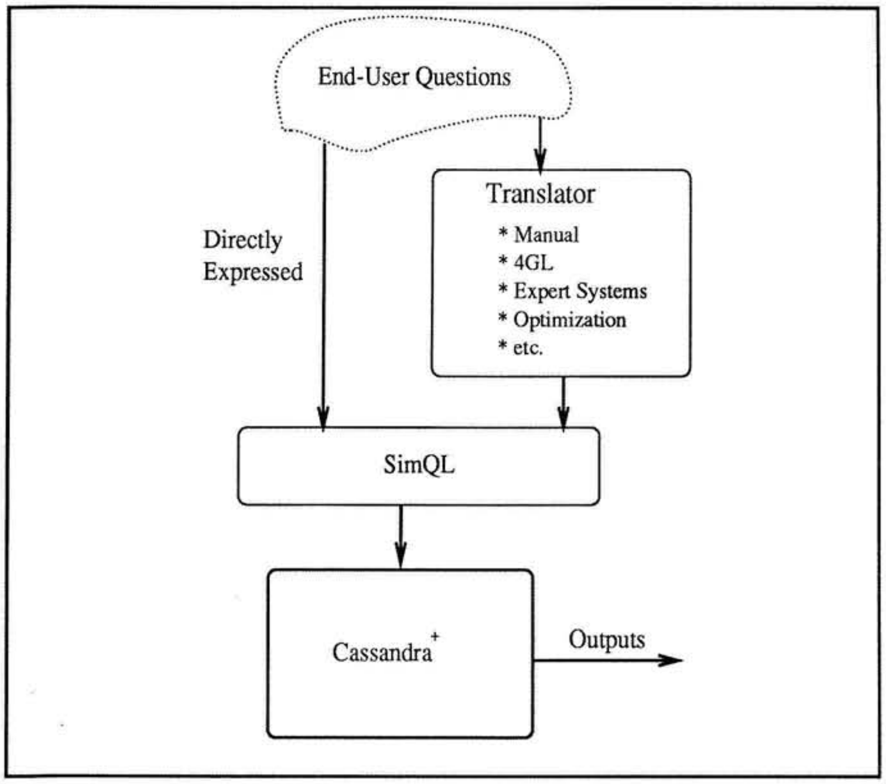

Figure 2: Translation of User Questions into SimQL

3.2 How End-User Questions Are Mapped into SimQL Queries

In Section 2, we presented various questions different people in a manufacturing organization want to ask about outcomes of their business processes, and in Section 3.1 we described the types of questions SimQL supports. Although some of these questions could be asked directly in SimQL as Examples 1 and 2 show, other questions cannot be directly expressed in SimQL. For example, the question "what should the lot size be to satisfy our current demand profile at the minimum cost?" cannot be expressed directly in SimQL because it is a high-level question requiring solving an optimization problem.

Since not all questions of interest to the end-user can be expressed in SimQL, this means that there should be a mapping (translation) from the high-level end-user questions into SimQL queries. For example, if a manager wants to know if it is better to assign a certain customer order to plant A or to plant B, where "better" is measured in terms of delivery time, then he or she can ask two SimQL queries, i.e. "how much time will it take to finish the customer order in plant A" and a similar query for plant B, and then compare the results.

Figure 2 summarizes the distinction between the user queries directly expressible in SimQL and those that need translation. The translation can be achieved by building various tools *on top* of SimQL queries[2] that translate high-level end-user questions into SimQL queries in one of the following ways:

Manually. In this case the user does the translation by him- or herself by formulating a set of SimQL queries and then analyzing the results. We illustrated this process with the question "to which plant is it better to assign a certain job?" As we showed already, the manager translates this question in his or her mind into two SimQL queries and compares the answers in order to obtain the answer to the original question.

Using 4GL's. The 4th Generation Language built "on top" of Cassandra$^+$ provides for the processing of some high-level queries. For example, the "two plant" query can be handled by a 4GL language as well.

Using Conventional PL's. The user can embed SimQL queries into conventional programming languages (e.g. C, COBOL, Assembler, etc.), as is typically done with conventional database query languages. For example, in large data-intensive applications, SQL queries are embedded into conventional programming languages, such as C, COBOL, or ASSEMBLER, each language requiring a separate interface [9]. Therefore, if we want to embed SimQL queries into a programming language, we have to provide an interface between *this* programming language and SimQL, as is done in the classic database case.

Using Expert Systems. An expert system can be built "on top" of SimQL. This expert system will reason about outcomes of various SimQL queries thus answering high-level questions asked by the end-user.

Using Optimization Modules. While dealing with optimization queries, a single query can spawn off several queries. For example in determining the optimum lot size, the optimizer will start with a particular lot size value and keep changing the size until an optimum is obtained. Each such size value will result in a different query and a different set of simulation runs.

As part of our future work, we plan to develop some of these translation tools. In particular, we are interested in the optimization module and in the application of genetic algorithms [16] as

[2]For comparison, database management systems usually have various tools built "on top" of the database that support high-level questions the end-users ask that are either difficult or impossible to express in the query language of the database.

a technique to solve this problem. In this paper, we assume that the translation is done either manually or by embedding SimQL into some conventional programming language.

As the sample of questions presented in Section 2 shows, translation of end-user questions into SimQL queries becomes progressively more difficult as we move up the management hierarchy as shown in Figure 1. For example, at the operational level of management most of the questions in Cells 1 through 4 presented in Section 2 can be expressed directly in SimQL. Some of the examples of such questions are "where will job-5 be 5 hours from now?", "how many jobs will be produced in the next shift?", "what will the material costs for order-5 be?", or "to complete order-20 in a week, how much overtime should be granted?" Furthermore, those queries that cannot be expressed in SimQL can be easily translated into SimQL queries.

At the tactical level of management, some of the questions in Section 2 are directly expressible in SimQL, whereas others are not. For example, such questions as "what will be the yield per plant for the next quarter for manufacturing Product-C?" or "what will be the average cost of producing product-A for the next quarter?" can be directly expressed in SimQL. On the other hand, such questions as "What should the lot size be to satisfy our current demand profile at the minimum cost?" or "Can I reduce my Work-In-Process (WIP) by reducing set-up times?" cannot be directly expressed in SimQL and require translation into SimQL queries. Nevertheless, the translation of these questions into SimQL queries is feasible. For example, the first query requires the solution of an optimization problem using one of the OR methods. The translation of the second query requires changing set-up times and asking by how much work-in-progress will be reduced for various set-up times.

At the strategic level, most of the queries cannot be expressed directly in SimQL. For example, questions such as "is it better to make a new microprocessor in-house for our new line of PCs, or is it better to buy it from the leading manufacturer of microprocessors?" or "what will the benefits of re-engineering of my business processes be?" that are of interest to the senior management, cannot be directly expressed in SimQL. Furthermore, it is difficult to translate these questions into SimQL queries.

In summary, most of the questions at the operational level of management can be directly expressed in SimQL or it is easy to translate them into SimQL queries. Although some of the questions at the tactical level can be directly expressed in SimQL, others require translation, which is typically not hard to do. Most of the queries at the strategic level of management require translation into SimQL queries, and it is often difficult to come up with this translation. This discussion is summarized in Table 1. This means that Query-Driven Simulations, and Cassandra^{+}

Levels of Management	Translation from End-User to SimQL Queries
Strategic	moderate to hard
Tactical	easy to moderate
Operational	direct to easy

Table 1: Translation Difficulty For Various User Queries

in particular, should be used primarily by operational and tactical management in organizations.

4 Comparison of QDS with the Traditional Approach

In Section 3, we described how Query-Driven Simulations (QDS) can be used to obtain answers about outcomes of business processes. It follows from this description that the QDS approach has the following advantages over the traditional *simulate-and-gather-statistics (SAGS)* approach, in which a decision maker runs simulations and gathers statistics about simulation outcomes.

First, QDS approach provides a more *declarative* way of asking questions about outcomes of simulations than the SAGS approach. The user formulates questions in a declarative general-purpose query language that tells the QDS system *what* the user wants to know. If the user uses this language he/she does not have to specify *how* the system has to obtain the answer. In particular, the user does not have to know any simulation and statistical packages, or write any programs.

Second, QDS approach gives the user extra *flexibility*. The user can ask *any* query expressible in the query language of the QDS system (e.g. SimQL for Cassandra$^+$). This flexibility makes the end-users less dependent on the MIS department since they do not have to rely on a simulation specialist when they want to ask additional questions not supported by the information systems installed by the MIS department.

Third, QDS approach is more *interactive* than the traditional SAGS approach. The user of the QDS system can ask queries "on-the-fly" as they arise without any help from the simulation specialist.

Finally, QDS approach *automatically* provides *statistical answers* to the questions asked by the user without any extra work on his/her part. Unlike the SAGS approach, in which the user has to determine how many simulation runs are needed in order to obtain the answer within the user-specified constraints, the QDS approach does all this work for the user.

However, the QDS approach has certain limitations in comparison to the traditional SAGS approach. First of all, some of the questions expressible in the SAGS approach cannot be expressed as QDS queries. To explain why this is the case, consider Cassandra$^+$, SimQL as its query language, and SQL as the core query language for SimQL. It is well-known that SQL has a limited expressive power, and therefore, SQL queries in complex applications have to be *embedded* in some general-purpose programming languages [10], such as C or Cobol, in order to express complex questions. In contrast to this, the SAGS approach uses general purpose simulation languages, such as MODSIM [4], SIMSCRIPT [7], GPSS [17], and it is well-known that they can perform certain computations that SQL queries cannot do [1][3]. To solve the problem of the limited expressive power of the QDS approach, we propose the same strategy used in SQL applications by embedding SimQL queries in a general-purpose programming language.

Another limitation of the QDS approach is that it requires generation of large trace files. In the worst case, the QDS system must trace *every* event occurring in the simulation model. In contrast to this, the SAGS approach can simply compute summary statistics inside the simulation model and thus generate no trace files *at all*. Furthermore, if the trace files are generated by the SAGS approach, the model developer has a control over the events he or she wants to trace. This means that only a small fraction of all the events may end up being recorded in the trace files in the SAGS approach. Since recording events in the trace files can slow performance of the QDS in comparison to the SAGS approach, it is important to develop query optimization techniques for QDS systems. These techniques would allow recording only those events that are necessary for answering the query. We discuss some of these techniques in [37].

5 Applications Where Cassandra$^+$ Should be Used

The questions we described in Section 2 are important to decision makers in manufacturing organizations, and therefore, as was pointed out in Section 2, organizations have been trying to find ways to provide answers to them.

For example, consider the Logistics Modeling System (LMS) [13] that IBM uses as a dispatcher for its semiconductor facility near Burlington, VT. LMS is intended to coordinate actions and decisions of several logically isolated participants in a serially dependent system of activities. To do that, LMS maintains a real time database of all manufacturing transactions, a knowledge base and model information. In this system, it is important to know not only about the state of

[3]In other words, SQL is not Turning-complete, whereas traditional simulation languages are.

manufacturing processes at present, but also try to predict what will happen in the future [13]. For example, if a protective coating on a lot of wafers expires in 30 minutes then this lot should be processed before that happens; otherwise, the wafers would have to be re-coated and the whole process has to be started from scratch. Therefore, it is important to know when protective coatings on different lots of wafers will expire and make appropriate routing decisions.

Since LMS is a special purpose system whose main goal is to collect the *current* transactional information about the wafer fabrication activities on the shop floor and use this information for the decision support, it has only few questions about the future outcomes of wafer fabrication processes [12]. Therefore, *each* such question can be answered *individually* using the domain knowledge [12]. More specifically, a separate program is written to handle each such question [12]. Thus, the one-program-per-question approach is feasible in case of LMS because it handles only *a few* questions.

However in general, as discussed in Section 2, decision makers in manufacturing organizations have many questions about outcomes of manufacturing processes, and this makes the one-program-per-question approach less attractive. The situation becomes even more complex when an organization has multiple simulation models since a program has to be written for each question on each model in the worst case. For example, if decision makers want to ask 500 different questions against 20 different manufacturing models then it means that one has to write 10,000 programs that handle these questions in the worst case.

This discussion suggests the types of applications where Query-Driven Simulation (QDS) systems, and Cassandra$^+$ in particular, are most useful. These applications can be measured in terms of the following two dimensions:

- how many models are there in the modelbase
- how many different questions users of a QDS system want to ask about these models.

The best types of applications are those where users want to ask *many* queries about *various* types of models since in this case the alternative program-per-question approach is the most expensive. The least interesting application is when there are few models and the users want to ask only few questions about simulation results since in this case the program-per-question approach is feasible. This discussion leads to the following characterization of the QDS applications presented in Table 2.

In Section 2, we considered questions that decision makers in *manufacturing* organizations ask about outcomes of manufacturing processes, and showed that there are many questions of interest to them. Also, large manufacturing organizations typically have many different models describing various aspects of manufacturing [27]. Therefore, manufacturing applications fall into the "many-

		Models	
		Few	**Many**
Queries	**Few**	Fair	Good
	Many	Good	Excellent

Table 2: Applicability of Query-Driven Simulations to Different Types of Applications.

questions-many-models" cell in Table 2 and, thus, provide a good application for query-driven simulations and for Cassandra$^+$ in particular.

Besides manufacturing applications, we believe that military and transportation applications belong to the many-questions-many-models cell in Table 2 and, therefore, are also well-suited for query-driven simulations.

6 Query-Driven Modeling Lifecycle

In the previous sections we described what types of questions Cassandra$^+$ can answer and in what applications it can be used. In this section, we describe what it takes to run and maintain the system.

We identify the following major tasks in building, running, and maintaining a Query-Driven Simulation system:

1. *Model Development.* This task requires writing simulation models using one of the simulation languages, such as Simscript, Modsim, etc. These simulation models are developed by the *model developer.*

2. *Model Administration.* A group of simulation models are stored in the modelbase[4]. The modelbase also stores additional information about these models, such as the language in which the simulation model is written, default simulation parameters, the list of events traced by the model, and various other information needed for asking queries on that model. The modelbase is maintained by a *model administrator*, who is responsible for adding new, removing

[4]More precisely, the models themselves are stored in the secondary storage. However, the names of these models and their access paths are stored in the modelbase.

old, and updating existing models. In addition, this person grants access privileges to various users of the models in the modelbase.

3. *Model Querying.* This task is the major purpose of Query-Driven Simulation systems, and we described it at length in the paper. The *end-users* usually perform this task. To ask a query on a simulation model, the end-user has to make sure that the model exists in the modelbase. The end-user should also know the names of the temporal relations associated with the model that he or she intends to query. Finally, the end-user should know the names of the attributes in these temporal relations. All this information is contained in the modelbase. To retrieve this information from the modelbase, we need *another* query language that asks questions about *characteristics* of the models (as opposed to SimQL that asks questions about simulation runs). For example, this second query language should support such queries as "which simulation language is used to write model Manufact-Model-4," or "find the names of relations that can be queried for model Manufact-Model-5." Languages of this type were proposed before [22, 14] and can be used for that purpose on the modelbase. The information retrieved from the modelbase is needed by the end-user in order to ask questions about simulation outcomes.

All the tasks described above are closely related to each other in the way shown in Figure 3. As Figure 3 shows, the model developer initially designs one or several working simulation models of an enterprise and delivers them to the model administrator who installs them into the modelbase. After the model administrator installs the model in the modelbase, end-users can start using it by issuing queries about simulation outcomes expressed in SimQL.

At this point, either the user is satisfied with the model and keeps using it, or he/she might experience some problems. There are two types of problems the user can face. First, the information, as specified in the modelbase, does not satisfy user's needs. For example, it may turn out that the user wants to ask a query about a relation that does not appear in the modelbase but can be easily computed from the events that the simulation model traces. Note that this problem can be solved by the model administrator who makes appropriate *administrative changes* to the modelbase. Note that the simulation model itself is not changed for this type of a problem. Second, the user may want to ask a query that the simulation model cannot handle. For example, the user may want to know how many parts will be painted in red color within the next 10 hours, and the simulation model does not keep track of the colors of different parts and the painting information. In this case, the model developer must make changes to the simulation model *itself.*

In both cases, Query-Driven Simulations provide a *feedback loop* in the process of model devel-

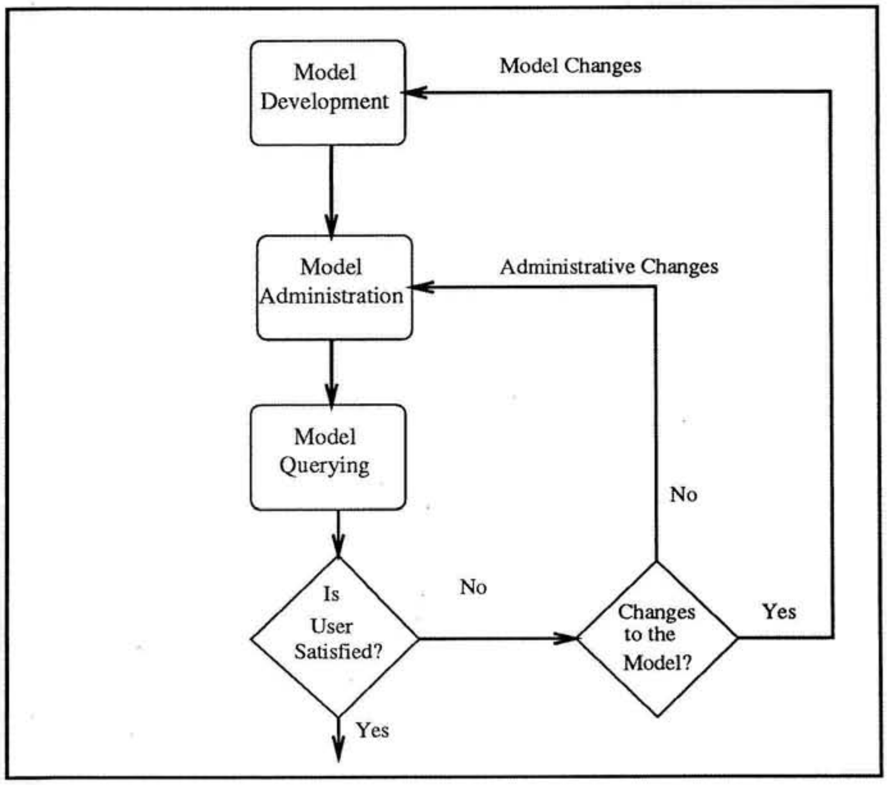

Figure 3: Query-Driven Modeling Lifecycle

opment and model administration: the models are modified based on the feedback coming from the user after the user asks various questions about these models. The process of development, installation, usage, and feedback represents a *Query-Driven Modeling Lifecycle* of a model. The model development process can go through several iterations before it converges to a stable simulation model satisfying end-user's needs.

We have described in this section the simulation modeling issues that arise while doing Query-Driven Simulations. A QDS system, as any other software system, also requires *system administration.* The system administration functions include setting resource quotas, assigning privileges, configuring the system, and so on. One function, special for the Query-Driven Simulations is to let the end-user use a requested temporal database for Query-Driven Simulations. For example, assume that user A wants to use Quel for asking queries about simulations and Ingres for storing simulation traces, user B wants to use SQL and Oracle for that purpose, and user C wants to use TQuel (assuming TQuel were implemented on some temporal database). In [37] we describe the steps necessary to take in order to switch a user from one database and query language to another one. In addition, all the three users should be assigned privileges specifying who can use which databases, and their accounts should be configured appropriately. All these tasks should be performed by the *system administrator.*

7 Conclusions

In this paper, we described how Query-Driven Simulations (QDS), and Cassandra$^+$ in particular, can be used by different people in various organizations for asking questions about future outcomes of business processes. The information provided by a QDS system can help managers make better planning, control, and staffing decisions.

In particular, we considered a manufacturing organization and described the types of questions decision makers across different functional units and levels of management ask about outcomes of manufacturing processes. We considered how these questions can be expressed in the query language SimQL and concluded that the higher the person is in the management hierarchy, the more difficult it is to translate his/her questions in SimQL.

We also described the types of applications that are best suited for Query-Driven Simulations. We concluded that the best applications are those where end-users want to ask *many* different types of questions about simulation outcomes of *many* simulation models. We showed that manufacturing applications belong to this category. We also believe that some other applications, such as transportation and military, also belong to this type of applications.

Finally, we pointed out that the end-users can ask queries about simulation outcomes of the models written in *any* simulation language, as long as its outputs follow a certain standard format. Furthermore, queries can be asked in *any* temporal relational query language of the user's choice. We also described various tasks required to operate a QDS system, such as simulation model development, maintenance of a modelbase, asking queries, and administering the system.

References

[1] A. Aho and J. Ullman. Optimal partial match retrieval when fields are independently specified. *ACM Transactions On Database Systems*, 4(2):168–179, 1979.

[2] R. Askin and C. Standridge. *Modeling and Analysis of Manufacturing Systems.* John Wiley & Sons, Inc., 1st edition, 1993.

[3] P. Balasubramanian and A. Tuzhilin. Cassandra$^+$: A System for Doing Query Driven Simulation. Working Paper IS-93-40, Stern School of Business, NYU, 1993.

[4] R. Belanger, B. Donovan, K. Morse, and D. Rockower. *MODSIM II Reference Manual.* CACI, 1990.

[5] Blanning, R. and Whinston, A. and Ai-Chang, M. and Dhar, V. and Holsapple, C. and Jarke, M. and Kimbrough, S. and Lerch, J. and Prietula, M. Model management systems. In Edward A. Stohr and Benn R. Konsynski, editors, *Information Systems and Decision Processes.* IEEE Computer Society Press, 1992.

[6] CACI. A quick look at SIMFACTORY II.5/SIMPROCESS: Manufacturing and Business Modelling. 1993.

[7] Consolidated Analysis Centers, Inc. *UNIX SIMSCRIPT II.5 User's Manual*, 1987.

[8] R. Conway, W. Maxwell, J. McClain, and S. Worona. *User's Guide To XCELL+ Factory Modeling System, Release 4.0.* The Scientific Press, 1990.

[9] C.J. Date. *An Introduction to Database Systems.* Addison-Wesley, 4th edition, 1986.

[10] R. Elmasri and S. Navate. *Fundamental of Database Systems.* The Benjamin/Cummings Publishing Company, 2nd edition, 1990.

[11] Jim Fidele. Manager of Manufacturing & Logistics, DEC. Personal Communications, Spring 1993.

[12] K. Fordyce. Manager, IBM. Personal Communications, Spring 1991.

[13] K. Fordyce, R. Dunki-Jacobs, B. Gerard, R. Sell, and G. Sullivan. Logistics Management System (LMS): An Advanced Decision Support System for Dispatch or Short Interval Scheduling. *Production and Operations Management*, 1(1):70–86, Winter 1992.

[14] A.M. Geoffrion. An Introduction to Structured Modeling. *Management Science*, 33(5):547–588, May 1987.

[15] R Head. Management Information Systems: A Critical Appraisal. *Datamation*, 13(5):22–28, May 1967.

[16] J. Holland. Genetic algorithms. *Scientific American*, pages 66–72, July 1992.

[17] IBM. *General Purpose Simulation System/360 User's Manual*, 1970.

[18] S. Kotha. Asssitant Professor, Leonard N. Stern School of Business, New York University . Personal Communications, Spring 1993.

[19] F. Kroger. *Temporal Logic of Programs.* Springer-Verlag, 1987. EATCS Monographs on Theoretical Computer Science.

[20] A.M Law and D.W Kelton. Simulation of manufacturing systems. In James L. Riggs, editor, *Simulation Modeling and Analysis.* McGraw-Hill Book Company, 2nd edition, 1991. Chapter 13.

[21] A.M Law and M.G McComas. How simulation pays off. *Manufacturing Engineering*, pages 37–39, February 1988.

[22] M.L. Lenard. A Prototype Implementation of a Model Management System for Discrete-Event Simulation Models. In *Proceedings of the 1993 Winter Simulation Conference*, pages 33–39, 1993.

[23] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag, 1992.

[24] J.A. Miller and O.R. Weyrich. Query Driven Simulation Using SIMODULA. In *Proceedings of the* 22^{nd} *Annual Simulation Symposium*, 1989.

[25] S. B. Navathe and R. Ahmed. TSQL – a language interface for history databases. In C. Rolland, F. Bodart, and M. Leonard, editors, *Temporal Aspects in Information Systems*, pages 109–122. North-Holland, 1988.

[26] Praveen Nayyar. Asssitant Professor, Leonard N. Stern School of Business, New York University . Personal Communications, Spring 1993.

[27] J. Pruett and V. Vasudev. MOSES: Manufacturing Organization Simulation and Evaluation System. *Simulation*, pages 37–45, January 1990.

[28] P. G. Ranky. *Computer Integrated Manufacturing.* Prentice-Hall, 1986. Chapters 6–8.

[29] SAS Institute, Raleigh, NC. *SAS User's Guide*, 1989.

[30] H.A. Simon. Artificial Intelligence, Simulation, and Modeling. *Interfaces*, 17(5):11–31, September-October 1987.

[31] M.R. Smith and J.M. Kay. A case study on the application of simulation to a car assembly line. *Proceeding of the 3rd International Conference in Manufacturing*, pages 207–234, November 1987.

[32] R. Snodgrass. The temporal query language TQuel. *ACM Transactions On Database Systems*, 12(2):247–298, 1987.

[33] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass. *Temporal Databases.* Benjamin/Cummings, 1993.

[34] L. Joseph Thomas, John O. McClain, and David B. Edwards. *Cases in Operations Management: Using the XCELL Factory Modeling System.* The Scientific Press, 1989.

[35] H.G. Thome. Planning and Monitoring of FMS by Simulation. *Proceeding of the 7th International Conference on Flexible Manufacturing Systems*, pages 137–149, September 1988.

[36] A. Tuzhilin. SimTL: A Simulation Language Based on Temporal Logic. *TRANSACTIONs of The Society for Computer Simulation*, 9(2):086–099, 1992.

[37] A. Tuzhilin and P. Balasubramanian. Query Driven Simulation: Issues and Solutions. Working paper, Stern School of Business, NYU, 1993.

[38] S. Walsh. Operations Manager in Manufacturing & Logistics, DEC. Personal Communications, September 1993.