Stock and Flow, and Unified Modeling Language Relationships

Prepared for:

The 21st International System Dynamics Conference 2003

New York

February 22, 2003

Background

A newcomer to System Dynamics (SD) announced via the SD Listserve that he was struggling to find some way to apply SD to a commercial project relative to Humanex, Inc., Korea, (Kwak 2002). Mr. Kwak volunteered that the project in question is an agent-based simulation effort and that his personal background is strongest in the domain of database and mobile programming. At first, he tried to use the Unified Modeling Language (UML), based on his software background, to facilitate effective communication with the project members.

After realizing that UML did not fit his problem, and that it threatened effective communications between his customer and programmers, he experimented with SD modeling after discovering Business Dynamics (Sterman, 2000). Alas, Mr. Kwak (2002) declared that it was not that easy to model his customer's extremely complex business logic with SD. Hence he posed the following questions to the SD Listserve:

1. Is there a comprehensive list of Pros and Cons regarding using SD in business modeling?
2. What are the limitations of SD regarding cases where it should not be used?

Responses to these questions addressed modeling with UML and the efficacy of SD. Forest (2002) clarified that UML produces static, detailed, non-runnable flowchart-like models, e.g., Visio-like. To Forest (2002), UML looks like flow charting and is a language for listing the steps in a process that can be use as a blueprint to write software.

In contrast, SD produces dynamic, general, runnable simulation models in time series. Forest (2002) discussed, based on his experience developing custom software, that UML and SD complement each other. Regarding UML, he quotes from Grady Booch's UML User Guide:" [It] is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system". The UML gives you a standard means of writing the system's blueprints, covering conceptual items such as the business processes and system functions as well as concrete items such as classes written in a specific programming language, database schemas, and reusable software components.

As a process, Forest (2002) said that he uses UML after he has scoped a problem with a SD model to identify the critical process(es) that need improved implementations. The SD model is the basis of the reference mode of problematic behavior that needs exploration:

- Clarifies priorities and what the critical processes are for improvement
- Identifies different policy implementations and explores their effects on the system as a whole, and
- Builds a shared consensus between people about any or all of these things.

According to Forest (2002), he uses SD to help decide what is the critical process(es) for improvement and what the new policies and procedures in that process will be. To him, UML is useful to clarify what software objects will be needed:

- What properties/methods/events they should have, and
- How they should be organized to maximize software performance, flexibility and maintainability.

Hypercube (2002) advised strongly against trying to learn and apply SD at the same time as applying it. Similar to Joseph (2002), Hypercube (2002) recommended that Kwak (2002) either get some help or spend some spare time on the project learning and trying to use SD to generate insights for himself, not his client.

Regarding UML, Hypercube (2002) stated that it can be useful to pick bits and pieces out of that method, e.g., to generate generic use cases which map on to the flows from one stock to another. Otherwise, Hypercube (2002) does not advocate using UML, based on his experience, in any way for business strategy/planning work.

Hypercube (2002) acknowledged that using SD in a management situation is not necessarily straightforward, but a method by which to explore a "perceived reality" and generate some agreement on key aspects of the situation. It appears that what is useful are the concepts of stocks and flows and how they are affected by a variety of factors and the consequences of delays in feedback. This focuses on the important aspects of the situation and builds understanding of the dynamics of the situation.

A third responder on the listserve, Joseph (2002) posits that SD methods are suitable in general to the Kwak (2002) situation. However, he cautions against changing paths (e.g., UML to SD) in the middle of a project suggesting that might be more catastrophic than the current situation (Joseph, 2002).

Joseph (2002) believes that communication difficulties in UML have two typical sources:

- The facilitator doesn't know UML that well, and
- The participants don't really know the domain field in which they are expected to be experienced.

To Joseph (2002), the circumstances of Kwak's (2002) problem warrant considering canceling the project and cutting short the client's losses, or bringing in a facilitator for one of the two methodologies: UML or SD.

Introduction

Finding synergism between UML and SD is a win-win situation for the systems development and system dynamics communities. Organizations that are familiar with UML will have an opportunity to "see" the relationship of systems defined in static UML artifacts being described in SD stock and flow models that lead to dynamic simulation models. Likewise, dynamic simulations that focus on critical system capabilities as stocks and flows will be describable in UML artifacts that support the development of new system capabilities, whether as a new product or information system process.

Without a bridge to communicate between the information system development and dynamic modeling communities, significant strategies or policies may not be implemented in information systems. And, as information systems are built without a focus on critical capabilities that implement key strategies or policies, they will have little business impact and a poor return on investment.

Bridging

To facilitate an understanding of the hypothesis that synergism between UML and SD will benefit information system developers and strategist as well, the following will be discussed:

- The UML Activity Diagram (AD)
- The SD Stock and Flow Diagram, and
- The Activity Diagram and Stock and Flow (S&F) synergism.

Key parts of the AD and S&F diagrams will be presented graphically, labeled and discussed. An example of a bridge between an AD and an S&F diagram will be given as evidence of the feasibility of performing bridging. Carrying the S&F model through to a simulation model is left for another time.

UML Activity Diagram

Fowler (1997) says that ADs are useful for understanding workflow and describing the behavior that has parallel processing. The core of the AD is the "activity" whose interpretation depends on the perspective or intent of the drawer of the diagram. According to Fowler (1997) the conceptual perspective shows a task that needs to be done whether by a human or an information system. From a system specification perspective, the activity is a software implementation method.

To many readers, an AD will appear like a flowchart as one activity is followed by another activity. What makes an AD unique from a flowchart is the ability to describe parallel activities using "synchronization bars". Activities that occur between synchronization bars happen in parallel and have to complete before the next activity may start.

Fowler (1997, p 131) says that ADs are important to business modeling because: "A technique like this that encourages parallel behavior is valuable in these situations because it encourages people to move away from unnecessary sequences in their behavior and to spot opportunities to do things in parallel. This can improve the efficiency and responsiveness of business processes."

Rational's (Rational-1, 2003) definition of ADs encompasses the ordering of tasks, to accomplish business goals, to include satisfying commitments between external and internal actors. The ADs help with the following (Rational-1, 2003):

- Providing a rationale for introducing information systems into business

- Establishing information system objectives to implement business transformation initiatives, and
- Justifying investments in information systems based on detailed business process metrics.

An example of an AD is presented in Figure 1 below. The figure shows the following (Rational-2, 2003):

- Activity states – the performance of an activity or steps in the flow of events.
- Transitions – the activity states that follow one another.
- Decisions – the guard conditions that control which transition follows a completed activity.
- Synchronization bars – the parallel subflows or concurrent threads in the flow of events.

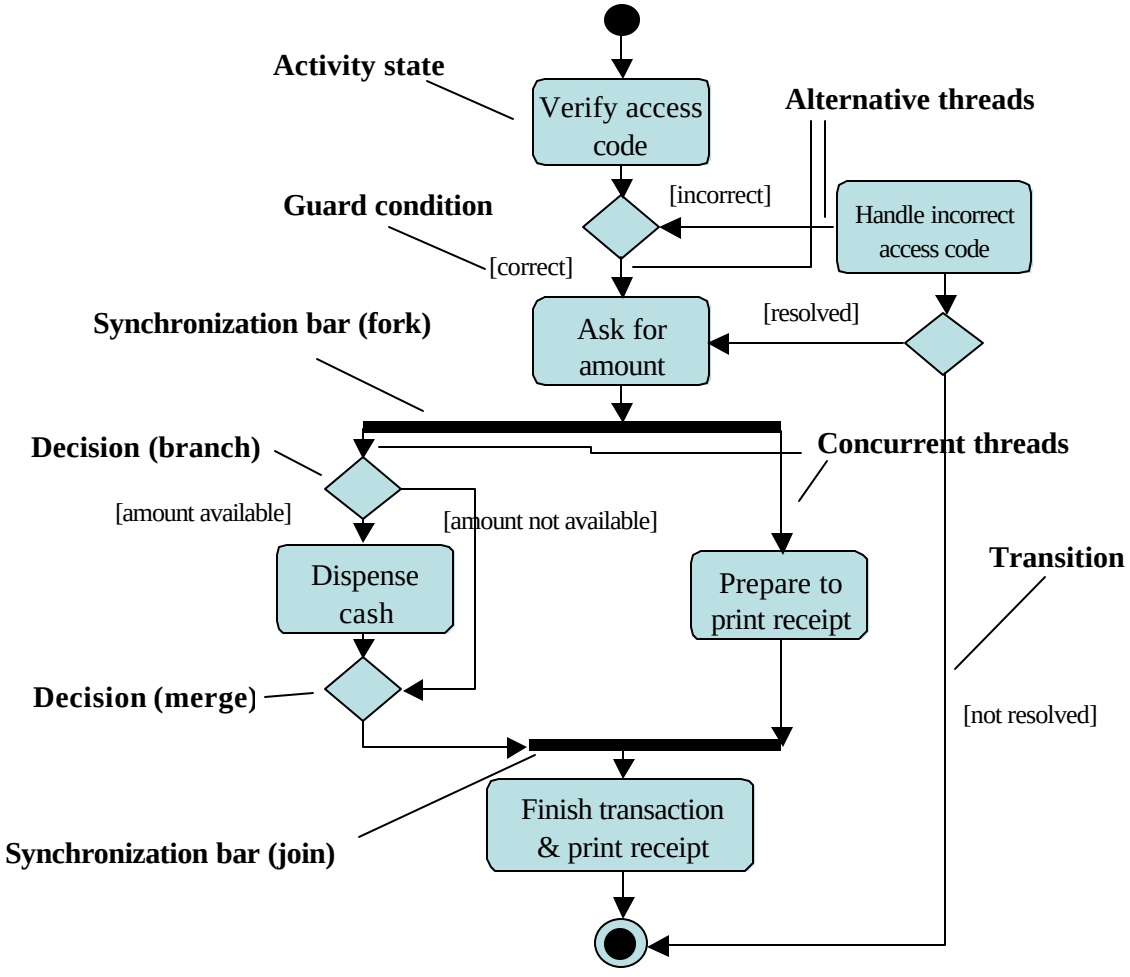

Figure 1 Annotated Activity Diagram © Rational Unified Process 2002.05.00.25

Stock and Flow Diagram

Like UML, SD has a diagramming notation. SD uses the diagramming notation described below and illustrated in Figure 2. The primary elements of the SD diagramming notation are as follows (Sterman, 2000, p. 192):

- Stocks, a container or accumulator of some thing, appear as rectangles.
- Inflows, adding to the stock, appear as a pipe with an arrow head pointing into the stock.
- Outflows, subtracting from the stock, appear as a pipe with an arrow head pointing out of the stock.
- Valves control the flows.
- Clouds, defining the boundaries of the model, represent the sources and sinks for flows.

The structure of all SD models consists of stock and flow elements, see Figure 2.

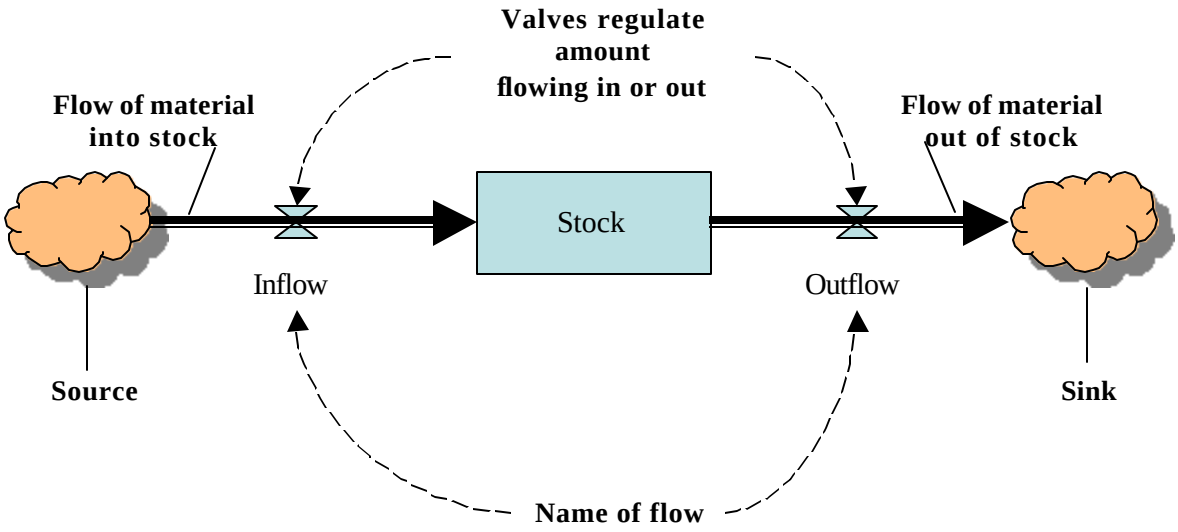

Figure 2 SD diagramming notation, (Sterman, 2000, p. 193)

Bridged Activity, and Stock and Flow Diagrams

For a recent assignment, a team prepared ADs to define business activities in order to identify business flows for an information system development project. As part of the project, identification of business process that should be developed first, critical process with high impact leverage points, were sought.

The business domain for the project is the U.S. federal government, which provides an information services to the public, to industry partners, and to other federal and local government entities.

One of the ADs developed addressed the "request for information" process and the activities performed by the government to satisfy the request. Figure 3 illustrates the activities performed. Within the AD, there are divisions of the activities by vertical bars that separate what is performed by the requestor of information, the provider of the information and the receiver of information for review and adjudication before providing an official response to the requestor.

In the parlance of ADs, the vertical separators presented on Figure 3 are called "swimlanes". An AD without swimlanes is excellent at telling the story of what happens in a business or system process flow, but does not readily indicate who or what is performing the activitiy(s) – the actor. A swimlane is the means of indicating who or what is the actor (Fowler, 1997, p.138). From an information system perspective the swimlanes may represent the software "class" responsible for each activity, where a class describes the objects in the information system and the various kinds of static relationships that exist among them (Fowler, 1997, p. 53).

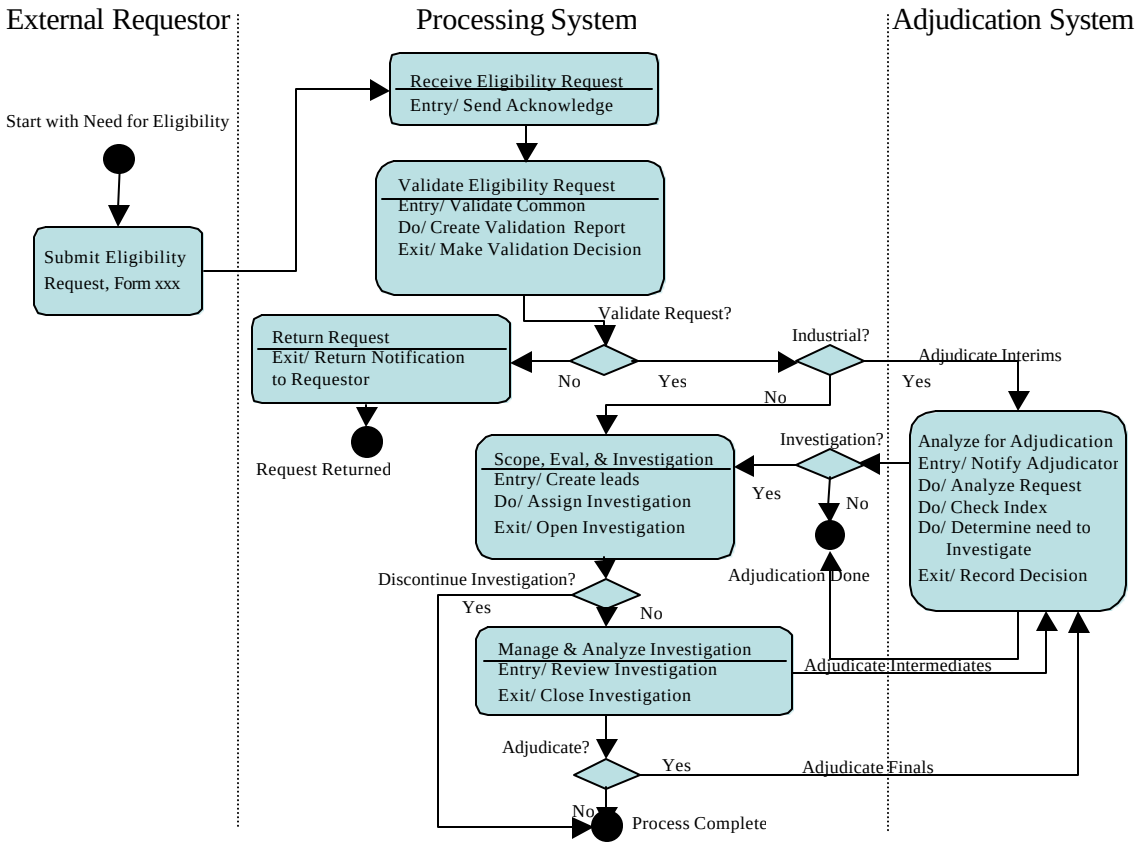

Figure 3 Activity Diagram with swimlanes

From the AD, an example SD stock and flow model was constructed as shown in Figure 4. The boundary of the SD model starts with the External Requestor and ends with the Adjudication System actors. The SD model focuses on the Processing System.

The SD model equates the AD activities to SD flows. The results of the AD activity/SD flow processes are the SD stocks representing accumulations:

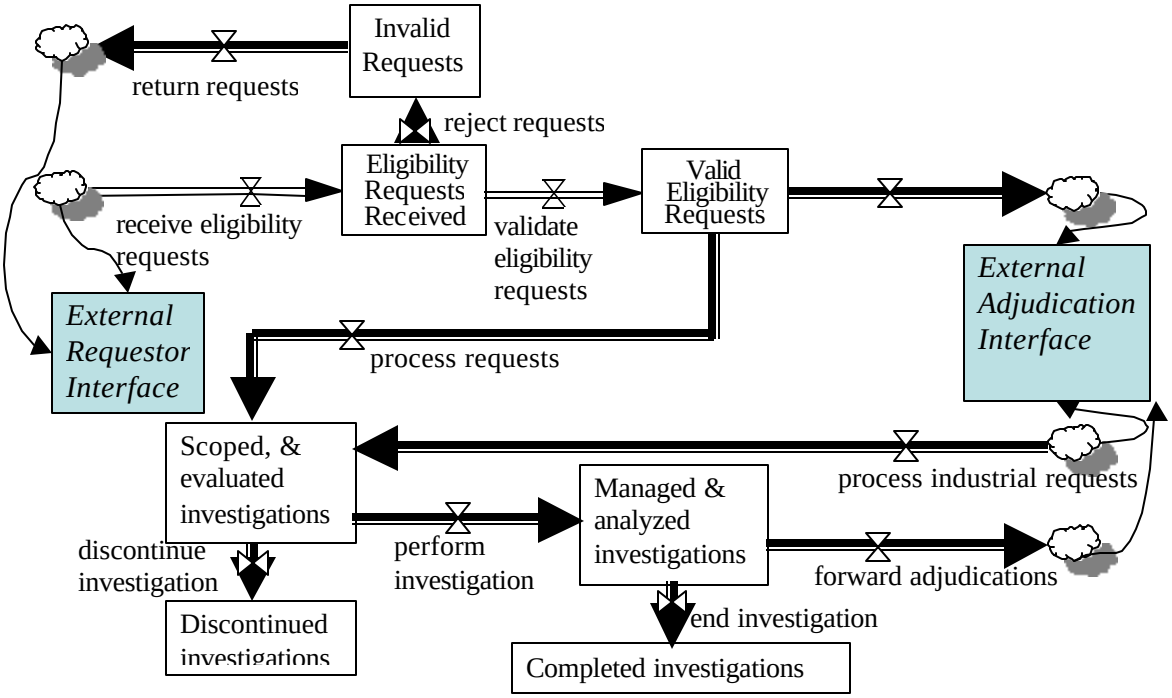

Figure 4 SD Stock and Flow Model based on UML Activity Diagram

Once the fundamental SD Stock and Flow model is established based on the AD, the remainder of the SD model needs to be developed in order to provide a capability to perform a simulation and analysis of the Processing System. From the SD perspective, the feedback loops, if any, and auxiliary variables, if appropriate, that help govern the flow in and out of the Stocks need to be established both graphically and algorithmically; this exercise is outside the scope of this paper, which focuses on the AD to SD Stock and Flow model.

For example, in the case of the model presented in Figure 4, the auxiliary variables that affect the flows in and out of Eligibility Request Received and Valid Eligibility Request are items such as the number of requests received from all sources, the typical delay in validating the requests, the rate of rejection of the requests, and the amount of staffing required to process the request. Figure 5 illustrates the SD Stock and Flow model that adds the auxiliary variables to the Eligibility Request Received and Valid Eligibility Requests stocks.

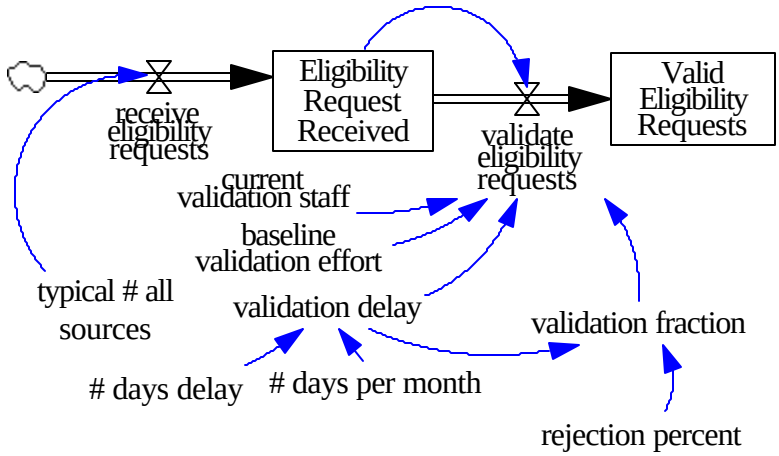

Figure 5 Illustration of SD model with auxiliary variables and feedback loop.

Once the SD model is developed to this point the simulation model needs to be formed to study the dynamics of the stocks and flows – the reference model and the algorithms. This exercise is not within the scope of this paper but identified as the next step once the SD stock and flow model is derived based on the AD.

Summary

Kwak (2002) announced via the SD Listserve that he was struggling to find some way to apply SD to a commercial project relative to Humanex, Inc., Korea, where he had begun modeling using UML. Responses to his request for help were posted by Forest (2002), and Hypercube (2002) Joseph (2002). Forest (2002) said that UML produces static models in contrast to SD that produces dynamic ones. He further elaborated that his preference is to use UML after he has scoped a problem with a SD model to identify critical business processes. In contrast to Forest (2002), Hypercube (2002) indicated his preference to pick bits and pieces out of the UML that map to the SD flows. The third listserve responder, Joseph (2002) generally thought that SD was appropriate to Kwak's (2002) problem but felt the best course of action under the circumstances was to not switch from UML to SD, but to consider canceling the effort to avoid further losses to the client.

This paper discusses the synergism between UML and SD as a win-win situation for system developers and system dynamacists. UML practitioners will have the opportunity to see static UML rendered into dynamic SD models. Likewise, SD modelers of information systems will have the opportunity to see high payoff processes and strategies implemented as new information system capabilities.

To facilitate an understanding that there is synergism between UML and SD, the following was discussed:

- The UML Activity Diagram (AD)
- The SD Stock and Flow Diagram, and

- The Activity Diagram and Stock and Flow (S&F) synergism.

Once the SD stock and flow model is developed from the AD, the simulation model needs to be formed – the reference model and the algorithms. This exercise is not within the scope of this paper but identified as the next step once the SD stock and flow model is derived based on the AD.

Conclusions

This paper illustrates by example that there is synergism between UML and SD that presents a win-win situation for information system developers, system dynamacists and their clients.

References

Forest, T (2002). REPLY Limitations of System dynamics? (SD3965). (October 17, 2002, 12:07:56). Retrieved November 11, 2003 from, http://www.vensim.com/sdmail/archives/

Fowler, M., & Scott, K. (1997). UML distilled: applying the standard object modeling language. Reading: ADDISON-WESLEY.

Hypercube, L (2002). REPLY Limitations of System dynamics? (SD3979). (October 31, 2002, 18:28:26). Retrieved November 11, 2003 from, http://www.vensim.com/sdmail/archives/

Joseph, R. (2002). REPLY Limitations of System dynamics? (SD3960). (October 14, 2002 19:31:33). Retrieved November 11, 2003 from, http://www.vensim.com/sdmail/archives/

Kwak, W (2002). QUERY Limitations of System dynamics? (SD3953). (October 14, 2002 12:30:02). Retrieved November 11, 2003 from, http://www.vensim.com/sdmail/archives/

Rational-1 (2003). Guidelines: activity diagram in the business object model. (n.d.), Retrieved January 14, 2003 from Rational\RatioanlUnifiedProcess\process\modguide\md_bactdb.

Rational-2 (2003). Guidelines: activity diagram in the use-case model. (n.d.), Retrieved January 14, 2003 from Rational\RatioanlUnifiedProcess\process\modguide\md_actdb.htm

Sterman, J. D. (2000). Business dynamics: system thinking and modeling for a complex world. Boston: Irwin McGraw-Hill.

Back to the Top