Dion Johnson

Dion Johnson is a senior test consultant for Pointe Technology Group, Inc. He's responsible for providing IT consulting services that focus on the overall system development lifecycle, with particular focus on the quality assurance and quality control elements. Prior to joining Pointe Technology Group, Dion was with Bell Atlantic, where he led a number of major testing efforts including an automated testing initiative.

POINTE TECHNOLOGY GROUP, INC.
8201 Corporate Drive, Suite 700 ♦ Landover, MD 20785 ♦ (800) 730-6171/(301) 306-4400 ♦ Fax (301) 306-4421

The Road to UML is Paved With Good Intensions

Presented by:

Dion Johnson
Senior Test Consultant

May 17, 2002

The Road to UML is Paved With Good Intentions

Dion Johnson
Pointe Technology Group, Inc.
8201 Corporate Drive
Landover, MD. 20785
djohnson@pointetech.com
diondjohnson@hotmail.com

1.0 Abstract

A picture is worth a thousand words. Does that mean that a model is worth a thousand requirements? A thousand test cases? Not exactly, but a model will tremendously aid in the development of requirements and test cases, and help facilitate inter-team communication of requirements and test cases; at least, that's always the intent. One way to help ensure that these good intentions come to fruition is to test the diagrams that the model is composed of, for 4C compliance – completeness, correctness, consistency, and clarity. There are different languages for producing models, but this presentation focuses on the Unified Modeling Language (UML), and methods of testing models that are created with UML.

2.0 Introduction

The Unified Modeling Language – better know by its acronym, UML – is fast becoming an industry standard for specifying, visualizing, constructing, and documenting the artifacts of software systems. In layman's terms, UML will be place on the list of major résumé buzzwords for the next couple of years. Interviewers will be pelted with UML laced statements such as, "I'm a hard worker, a quick learner and I know UML", or "I've got good communication skills – verbal, written and UML modeling". Well, maybe the phrases won't be constructed quite like that, but this simple three letter acronym will almost certainly be touted in many attempts at securing employment, simply because of the tremendous enthusiasm that UML is generating in the information technology universe. Joseph Schmuller, in his book entitled UML in 24 Hours, raves that "The Unified Modeling Language (UML) is one of the most exciting tools in the world of system development today." In Modeling XML Applications with UML, David Carlson writes that UML is "becoming the preferred language for describing business systems."

So what exactly does all of this mean? It means chaos and confusion that's what it means! UML is one of those technology crazes that has people excited. Many enthusiasts pick up UML books with good intentions for how it will improve the application development lifecycle by increasing traceability and opening the lines of communication among project members. Good intentions are great, but the road to UML is paved with good intentions – the vehicle for traveling that road, however, is the knowledge of the language. Most people realize this, but don't realize that vehicles need good alignment, so they end up running off of the road and never reaching their destination. Every development vehicle needs an alignment system, and the alignment system is known as quality assurance/quality control (QA/QC). UML is no exception. The UML alignment system should be made up of peer reviews that check for the following compliance areas: correctness, completeness, consistency and clarity (each defined in Table 1).

Area	Definition
1. Correctness	There are two types of correctness that need to be addressed when inspecting models: *syntactical* correctness and *logical* correctness. a. ***Syntactical Correctness*** addresses whether or not the diagram's syntax is correct based on modeling language rules. b. ***Logical Correctness*** addresses whether or not the logic and/or flow of the diagram is correct.
2. Completeness	This addresses whether the diagrams are missing anything or tell a sufficiently complete story of what is being modeled. Completeness also addresses whether the diagrams function as expected under all test conditions. This will involve stepping through the diagram and plugging in multiple real life scenarios and/or data, to verify functionality and identify scenarios that still need handling or can't be handled in the model.

3. Consistency	This addresses the uniformity of the diagrams that compose the model, relating to their syntax, semantics, notations, and functionality. There are two main types of consistency: *intra*-diagram consistency and *inter*-diagram consistency. a. ***Intra-diagram consistency*** addresses whether or not consistency exists between all of the components within a single diagram. b. ***Inter-diagram consistency*** addresses whether or not consistency exists between all of the diagrams that make up a project model.
4. Clarity	This addresses how well the diagrams have been developed by examining two aspects of clarity: understandability and simplicity. a. ***Understandability*** addresses the readability and intelligibility of the diagrams that compose the model. This asks whether or not someone can sit down, with or without the author, and understand what the model's diagrams are attempting to convey. This is important because many of the people that use analysis diagrams are not UML literate. b. ***Simplicity*** addresses the straightforwardness of the model's diagrams. There are many different ways to model one thing, but it's important to try to model it in the simplest way possible. This is what assesses whether or not a diagram has been broken down to its lowest common denominator.

Table 1: 4C Definitions

This paper offers a 4C-compliance test methodology that is derived from standard peer review processes and customized to meet the needs of UML models. This paper is organized by:

- First, introducing the reader to different levels of peer reviews, then
- Introducing the reader to UML and different types of UML diagrams, and finally
- Revealing the methodology the results when UML meets the peer review process.

3.0 Peer Review Process

Peer reviews are a cost effective method of improving the quality of software. A peer review is a process in which the peers of a software work product's author examine that product to identify potential defects. There are many different levels of peer reviews ranging from very structured, to less structured. Four of the major types of peer reviews are:

1. Inspections
2. Team Review
3. Walkthrough
4. Ad hoc Review.

Figure 1 shows the four peer review levels on a formality scale, revealing Inspections as the most structured. Evidence shows that more formal reviews uncover the most bugs in work products.

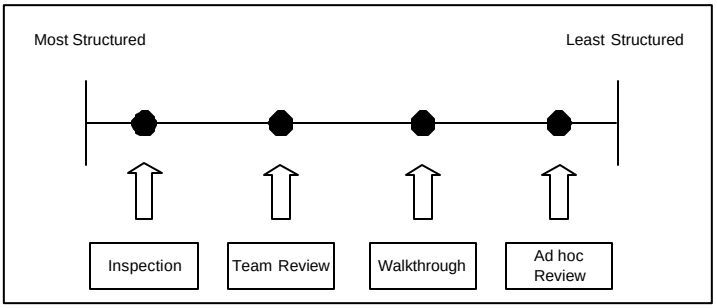

Figure 1: Peer Review Formality Scale

All of the peer review levels involve some sort of face-to-face meeting with the author and the reviewers, and this is because the personal interactions during the meeting can reveal new bugs as one person's

observation stimulates another's thinking. The stages that may be included in a peer review process are listed in Table 2.

Peer Review Stage	Description
Planning	Identifies work product to be inspected and sets the inspection schedule.
Overview	Optional stage where members of the peer review team who are unfamiliar with the work product to be inspected receive orientation. This stage is always optional.
Preparation	The work product is sent to the members of the peer review team, and they review it individually, looking for defects.
Meeting	Members of the inspection team meet to discuss possible defects in the work product.
Rework	The work product is revised based on defects that have been slated for resolving.
Follow up	The rework is verified, final peer review data is collected and summarized, and the peer review is closed.

Table 2: Peer Review Stages

Each stage usually has an identified purpose, entry criteria, a set of tasks to accomplish, and exit criteria. The following subsections provide a description of each of the four levels of reviews that are mentioned above, and reveal what stages are actually included in each respective level. For more detailed information, see the reference section at the end of this document.

3.1 Inspections

As shown in Figure 1, an inspection is the most systematic and structured type of peer review, and it is normally the peer review of choice for validating work products that are in a final draft state. It follows a well-defined process and commonly includes all six stages shown in Table 2.

An inspection meeting is usually carried out as follows: a participant leads the meeting (moderator), present the material to the reviewers (reader) and records issues as they're brought up (recorder). The holder of the moderator position may also hold the recorder or reader position, but participants other than the author must hold all three positions. Participants prepare for the inspection by examining the material on their own to understand it and to find problems. During the meeting, the reader presents the material a small portion at a time to the other inspectors, who raise issues and point out possible defects. At the end, the team agrees on what items are actually defects that will be reworked by the author.

3.2 Team Reviews

Team reviews, also known as “structured walkthroughs”, follow several steps found in an inspection. The participants receive the review materials several days prior to the meeting, and are expected to study the materials on their own and the team collects data on the review effort and the number of defects found. The overview and follow-up inspection stages are simplified or omitted, and some participant roles may be combined. Also, the author may lead the team review. The reader role is optional. If the reader role is omitted, the moderator simply asks the participants if they have any questions or comments about any part of the work product.

3.3 Walkthroughs

A walkthrough is an informal review in which the author of a work product describes it to a group of peers and solicits their comments. Walkthroughs are different from inspections in that the author takes the dominant role; there are no other specific reviewer roles. Walkthroughs typically don't follow a defined procedure.

3.4 Ad hoc Reviews

Ad hoc reviews typically have no major structure or planning. They are typically called at the spur of the moment and consist of the author walking through a work product with one or more participants in an

effort to solve a particular issue. Several of these types of reviews may occur throughout the development of the work product, just to ensure that the product is on the right track.

4.0 Introduction to UML

UML is a language for modeling systems using object-oriented concepts, but it is not the first of its kind. This type of language first began to take shape in the 1980s, and developed into a force in the mid-1990s with notable methods created by Booch, Rumbaugh, and Jacobson. UML was formed when these notable methods were joined to create one unified method that is now known as UML. UML defines a standard language and graphical notation for creating systems and business models that enables builders of systems to create blueprints to capture their visions and communicate them to others. Many diagrams may be included in a model, but most types fit in one of the following categories:

1. Use Case Diagrams
2. Static Structure Diagrams
3. Interaction Diagrams
4. State Diagrams
5. Activity Diagrams
6. Implementation Diagrams

Note: All, except implementation diagrams, are considered analysis & design diagrams. The paper's scope is to introduce these diagrams, not provide details. Sample diagrams that have been included, will be used later to illustrate some concepts.

4.1 Use Case Diagrams

A use case is a description of a system's functional behavior from the standpoint of the system's users. For example, if the system allows a user to order books online, the use cases may allow a user to view a list of books, view a shopping cart list, modify a list of ordered books, submit an order, and cancel an order. The use case representation for this Online Books System may look like the diagram shown in Figure 2.

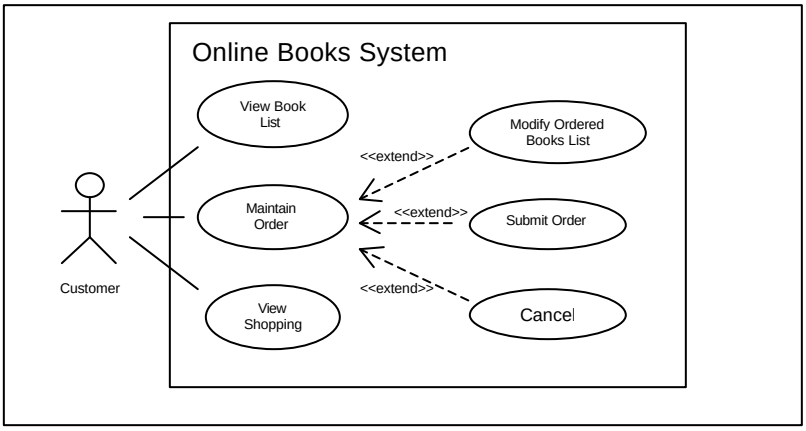

Figure 2: Online Books Use Case Diagram

4.2 Static Structure Diagrams

Static structure diagrams represent static information and include two different types of diagrams: class diagrams and object diagrams. Classes are groupings of things that have common attributes and behaviors. An object is an instance of a class, which means it's simply a class with specific values assigned to attributes and/or behaviors.

4.3 Interaction Diagrams

While static structure diagrams represent static information, interaction diagrams represent the interactions that these different objects have with one another. Interaction diagrams have two different

diagram types: sequence diagrams and collaboration diagrams. Both show interactions among objects, but sequence diagrams show these interactions over time.

4.4 State Diagrams

State diagrams also relate to static structure diagrams. Interaction diagrams normally deal with several objects at a time, however, while state diagrams normally deal with one object at a time. State diagrams show the dynamic behavior of a single object. For example, in the Online Books example used above, an order is considered "pending" while a user is in the process of completing it, "submitted" once a user electronically sends it, "modified" once a user changes it and "canceled" once a user cancels it. Each of these adjectives enclosed in quotations is considered a state, and Figure 3 illustrates these states.

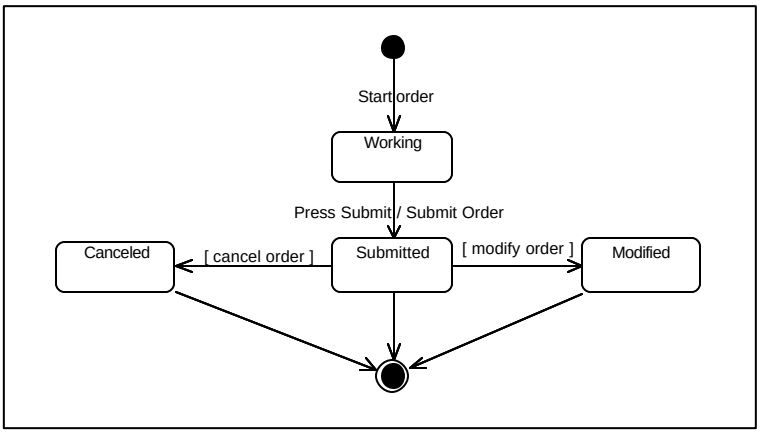

Figure 3: Orders State Diagram

4.5 Activity Diagrams

Activity diagrams commonly illustrate how activities work together to accomplish a business workflow.

4.6 Implementation Diagrams

Implementation diagrams show the physical architecture of a computer-based system, and include two different types of diagrams: component diagrams and deployment diagrams.

5.0 UML meets Peer Review

The meeting between UML diagrams and the peer review process is not unlike the meeting between the peer review process and any other document, aside from the fact that diagrams are more visual than plain text. The visual aspect makes diagrams behave more like systems. While reading validates text, a system, calls for inputs, holds variables and produces outputs. And it is this distinction that moves the meeting between peer reviews and UML from being simply a reading assignment to being a small-scaled system test effort. Do not panic! Notice that the term "small-scaled" is used. Peer reviews should be cost efficient. If too much time were spent validating diagrams, it would not have been worth it. The term "system test effort", is only used to help reveal how unique UML model reviews are.

Most testing efforts include test planning, test design, test execution, and defect tracking stages. Although UML model peer reviews are peer reviews, and should follow the basic peer review structure described in the Peer Review Process section, the "system test" aspect allows the stages included in a testing lifecycle to be used to describe the UML model peer review effort. The following sections reveal procedures for carrying out the UML model peer review process using the stages listed in Table 2, and the testing stages listed above.

5.1 Planning Stage

5.1.1 Purpose

The purpose of this stage is to complete planning and design of cases to be implemented in later stages.

5.1.2 Entry Criteria

The entry criterion is the UML Model. The peer review level determines the Model's required state.

5.1.3 Tasks

The tasks executed during this stage, relate to test planning and test design, and are listed below.

1. Test Planning
   a) Verify that the entry criteria has been met
   b) Identify peer review types
   c) Identify the peer review team(s)
   d) Determine if an overview is needed
   e) Schedule peer reviews
2. Test Design
   a) Develop or locate peer review checklists
   b) Develop test cases
   c) Determine test cases to implement
   d) Distribute peer review work pack.

5.1.3.1 Test Planning Tasks

5.1.3.1.1 Verify Entry Criteria

To begin the planning stage, the entry criteria must be verified.

5.1.3.1.2 Identify Peer Review types

The Peer Review Process section identifies peer review ***levels***, while the Introduction section identifies the UML compliance areas. Combining these elements forms the 5 basic ***types*** of UML peer reviews.

Peer Review Type	Description	Suggested Reviewers
Clarity (simplicity)	This may be one of the first peer reviews conducted, and will probably be an ad hoc review or a walkthrough. The purpose of the review is to point out redundancies and simplification methods, before nearing completion.	UML "experts"
Completeness and Correctness (Logical)	Checks the diagrams for completeness and *logical* correctness, which naturally go hand in hand. It is very difficult to examine logical correctness without addressing completeness, and vice versa, because a diagram that is not complete, will often not make sense logically.	Developers, Req. Analysts, Testers, Project Manager
Correctness (Syntactical)	Checks the model diagrams to ensure the diagram syntax follows the modeling language rules to an acceptable degree, and maintains intra-diagram consistency (which often affects how syntax is assessed).	UML "experts"
Consistency Peer Review	Checks for inter-diagram consistency.	UML "experts"
Clarity (Understandability)	Checks the analysis diagrams for understandability to see how well people that aren't familiar with UML syntax will be able to understand the model diagrams. diagrams may not be proficient with the UML.	UML illiterate reviewers

Table 3: Peer Review Types

Combinations of the peer review types may be combined into a single peer review. The reason it's important to identify each type individually is because each of the basic types has its own purpose, and its own set of recommended participants. Advantages of holding different types of peer reviews at different times include more efficient use of resources, ability for greater focus to be given to each compliance

area, shorter peer review meetings, and the fact that diagrams can be partially reviewed while other diagrams in the model await completion.

The planning stage is used to determine what's best for the model. The coordinator plans the order for reviewing diagrams, and what types of peer reviews will be used. The suggested reviewers are based on analysis and design diagrams. For implementation diagrams, the clarity review types may be omitted, while the completeness and logical correctness review may only have developers in the reviewer roles.

5.1.3.1.3 Identify Peer Review Team(s)

The roles of moderator, reader, reviewers and recorder need to be identified.

5.1.3.1.4 Determine If An Overview is Needed

The peer review coordinator determines if the members of the peer review team have sufficient knowledge of the model. If not, an overview needs to be set up.

5.1.3.1.5 Schedule Peer Reviews

Once the peer reviews have been determined, they can be scheduled.

5.1.3.2 Test Design

5.1.3.2.1 Develop Or Locate Peer Review Checklists

The checklists contain brief statements that help to ensure the diagrams are being checked for compliance.

5.1.3.2.2 Develop Test Cases

Test cases for analysis and design diagrams may be produced from use cases. Implementation diagram test cases may also be produced from use cases and related design specifications. If the diagram being tested is a use case model, business requirements can be used to create test cases.

For peer reviews that verify consistency, it is helpful to identify the set of relationships among the model's diagrams that must be maintained. Figure 4 illustrates these relationships.

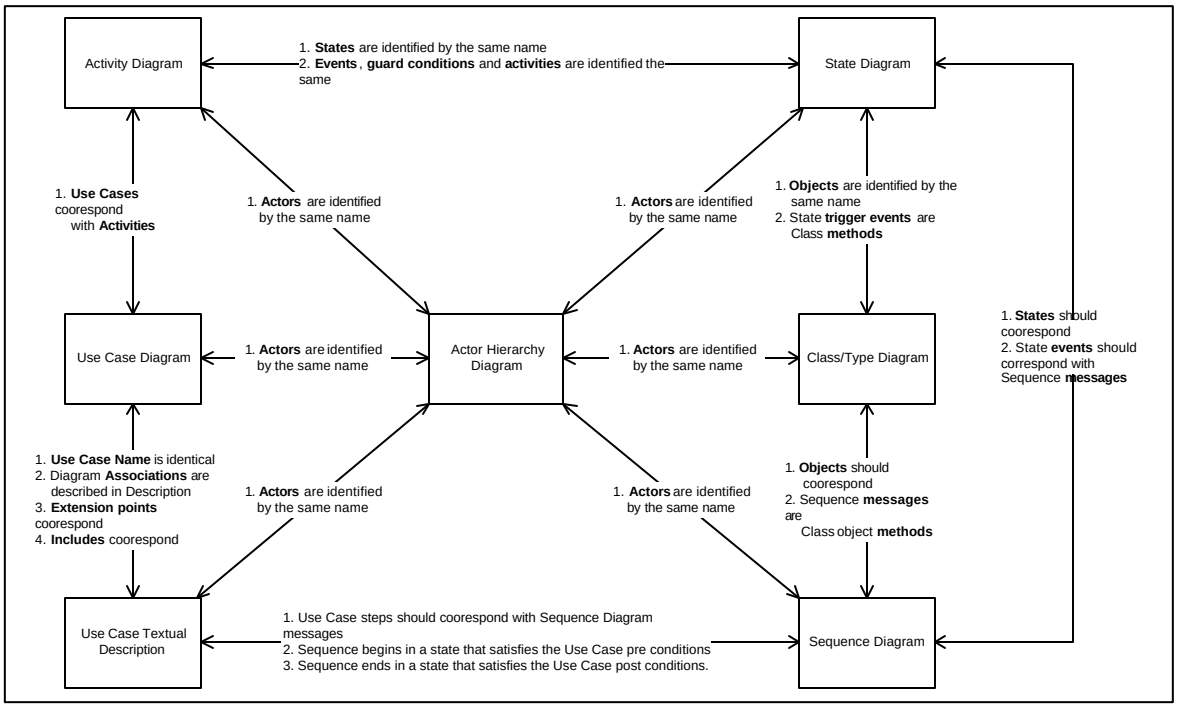

Figure 4: Diagram Relationships

5.1.3.2.3 Determine Test Cases To Implement In Review

Many test cases may be identified, and even sent out to reviewers for the preparation stage, but not all of them will be implemented in the peer review meeting. The planning stage is used to identify which ones will be implemented in the review meeting.

5.1.3.2.4 Distribute Peer Review Work Pack

Once the peer review types have been identified and scheduled, and the teams have been identified, the work pack can be distributed to the team members. A work pack is sent out prior to each peer review, and includes:

- The diagrams to be reviewed
- Appropriate checklists
- Defect list
- The test cases that the reviewers may use during the preparation stage.

5.1.4 Exit Criteria

The exit criteria for exiting the planning stage includes:

- A test plan composed of a list of peer review types, peer review teams and a peer review schedule.
- Peer review checklists
- Test cases
- Peer review work packets

5.2 Overview Stage

5.2.1 Purpose

To provide the peer review team with a high level description of the UML model.

5.2.2 Entry Criteria

The entry criterion is that decision to have this stage, which is made in the planning stage.

5.2.3 Tasks

The task to complete is the overview of the UML model.

5.2.4 Exit Criteria

The primary exit criterion is the sufficient understanding of the UML model, by the peer review team.

5.3 Preparation Stage

5.3.1 Purpose

To complete the execution of any test cases that are part of the work packet and to track defects that are found. Also, the reviewer is to record an approximate amount of time that it takes to complete the review.

5.3.2 Entry Criteria

The entry criterion for the preparation stage is the peer review work packet.

5.3.3 Tasks

The tasks that are executed during this stage include:

1. Test execution
2. Defect tracking
3. Record preparation time.

5.3.3.1 Test Execution

Reviewers review the UML diagrams that have been sent to them. If test cases have been sent to them as part of the work packets, the reviewers may focus on reviewing the document with those cases to help them identify potential defects.

5.3.3.2 Defect Tracking

Defects found by the reviewer during test execution, should be recorded on the defect list. The recorded issues are not official defects yet, but should be brought to the peer review meeting for discussion.

5.3.3.3 Record Preparation Time

During this process, the reviewer should record the amount of time that it takes to complete the review.

5.3.4 Exit Criteria

The exit criteria for this stage include the completed defect list and the recorded preparation time.

5.4 Peer Review Meeting Stage

5.4.1 Purpose

The purpose of this stage is to uncover defects in the UML diagram(s) being reviewed.

5.4.2 Entry Criteria

The entry criteria for the peer review meeting stage are the appropriate test cases and the completed defect lists.

5.4.3 Tasks

The tasks for this stage are fairly the same as the tasks for the preparation stage, except in this stage a group is involved, so there will be a delegation of activities. The tasks include:

1. Test execution
2. Defect tracking

5.4.3.1 Test execution

The execution of the test will involve walking through the given diagram guided by the appropriate use case scenario. For example, let's look use the Online Books system used in Introduction to UML section of this paper. If the diagram that we are testing is a state diagram that shows the states that an order can hold, we may want to use the Maintain Order use case, shown in the Figure 2, as the basis for the test case. Since this sample diagram is only one page, and is so small, it would make more sense to combine multiple peer review types into one peer review, if all of the desired resources are available. When dealing with larger diagrams, however, it may make more sense to hold different peer reviews for the different peer review types.

Although the case comes from the use case diagram, unless the peer review is checking for consistency, the reader would not execute the case by walking through the use case diagram, but rather by walking through state diagram using the Maintain Order use case scenario. The reader should present the diagram. Let's take a snap shot from what the peer review meeting may entail:

Reader – While looking at the state diagram says, "The user of the system is in Books Online

application, and starts filling out the online order form. At that point, as in the state diagram, the order is in a 'Working' state. After filling out the order form, the user submits the form electronically by pressing the submit button. At that point the order is in 'Submitted' state. Does everyone see that?"

Reviewer 1 – "Well, I see it, but there seems to be a slight syntactical error in the diagram. In the diagram, there is a line that reads, 'Press Submit / Submit Order'. 'Submit Order' is an event that occurs that causes the transition to the submitted state. 'Press Submit' is an action that the user performs to cause the event, right.

Author – "Right."

Reviewer 1 – "I think UML syntax says that when using events and actions with the transition arrow, the event comes **before** the backslash, while the action comes **after** the backslash". You have in the opposite order."

Author – "You're right. Thanks."

Moderator – To the recorder, "Can you note that as a defect?"

Recorder – "I've got it"

Moderator – "If there are no more issues with that, let's continue."

Reader – "Ok. After submitting the order, the order is in submitted status. Later on, the user may decide that they want to add more books to the order, so they reopen the order, then modify it, and resubmit it, at which time the order is in modified status. Any comments on this piece?"

Reviewer 2 – "Well, it makes sense, but since the order is submitted again, shouldn't there be an arrow going back to the submitted state, after the Modified state?"

Author – "Well, the Modified state is defined as being the same as the submitted state, except that the order has been changed from what it originally was, and resubmitted. Because of this definition, which will appear in the glossary of the high level requirements document, I didn't think that an arrow needed to go from the Modified state to the Submitted state."

Reviewer 2 – "Ok. That sounds good. As long as modified is defined the way you say it is, I guess this isn't a problem."

Reviewer 3 – "I've got a question. When the order is reopened, that seems like it is a state in itself. Shouldn't there be a Reopened state? Or if the reopen is no different than the Working state, shouldn't there be an arrow going from the Submitted state back to the Working state?

Author – "That's a good question. I'm not sure if the system is supposed to handle the Working state and a Reopen state differently, but you're right in saying that one of them needs to come from the Submitted state. I'll check on that."

Moderator – Says to the recorder, "Could you note this as a defect, and also note that the author needs to find out if there is a difference between the Working state and a possible Reopened state?"

Recorder – "Ok."

Moderator – "Let's continue."

Reader – "Ok. We can start from the Submitted state. If the user decides to cancel the order, the order then goes into Canceled state, and is not completed. If the order is not canceled, the books are sent out, and that is the end of the process. Any questions?"

Reviewer 4 – "I have a question. What happens to a Submitted or Modified order once it has been

completed and the books have been sent out?"

Author – "A Books Online employee goes into the system and closes out the order. I think I see your point. There needs to be a state that reads, 'Completed' or 'Closed Out state'."

There are many important things, about executing test cases, and about peer review meetings in general, that can be taken from the example above. The important points are as follows:

- The way issues are brought up is very important. Notice that in the example, the reviewers were very non-assuming. They made their points, but made them in a way that allowed the author to reach the same conclusion on his own. Asking questions instead of making blunt statements is a good technique to use. This will prevent the author from getting defensive. Plus if the statement is wrong, as was the case with Reviewer 2, the author will feel no reason to try to make an example out of them, and scare others from making comments.
- By executing test cases, the peer review team now has a way of verifying what's in the diagram, and what's ***not*** in the diagram. By simply walking through the diagram with no test scenario, items like the ones brought up by Reviewer 3 and Reviewer 4 may have been over looked. By using a test scenario to walk through the diagrams, however, the reviewers were truly able to verify the completeness of the diagram.
- Not all of the problems will have an immediate solution, such as the problem found by Reviewer 3. That is fine. As a matter of fact, an issue should be discussed for about 1 to 2 minutes at the most. If it takes more time than that to resolve the issue, it should probably just be noted, and addressed at another time outside of the peer review meeting.

When the peer review is checking for consistency, a walk-though may not be necessary. If it's not necessary, the peer review can consist of discussing issues found by reviewers, when comparing one diagram to another in the preparation stage. If it is necessary, the walk-through should be conducted by using one diagram to drive the walk-through, while looking at the related diagram as an output to the driving diagram. This will allow the established relationships to be verified (see Figure 4 for an example of some diagram relationships).

5.4.3.2 Defect Tracking

Many issues were found in the example above. The issues that are found should be recorded as they come up during the review. At the end of the review, the issues need to be revisited, and it needs to be determined which ones will become defects that need to be addressed in the rework stage. One of the following three peer review determination needs to then be chosen:

1. ***Complete Acceptance*** – Accept the diagram(s) as complete, with no defects to rework.
2. ***Conditional Acceptance*** – Accept the diagram(s) with the condition that defects will be reworked and verified by the Moderator.
3. ***Repeat Review*** – Hold another peer review following the rework of the defects.

5.4.4 Exit Criteria

The exit criteria include completed master defect list and peer review determination.

5.5 Rework Stage

5.5.1 Purpose

The purpose of the rework stage of the UML model peer review is to fix defects in the UML model that were marked for rework in the peer review stage.

5.5.2 Entry Criteria

The entry criterion for the rework stage is the set of defects from the peer review meeting stage.

5.5.3 Tasks

The tasks for this stage revolve around the author resolving the appropriate defects, and notifying the moderator upon completion.

5.5.4 Exit Criteria

The exit criterion for this stage is the revised diagram(s).

5.6 Follow up Stage

5.6.1 Purpose

The purpose of this stage is to complete the peer review process.

5.6.2 Entry Criteria

The entry criterion for this stage is the revised diagram(s).

5.6.3 Tasks

The tasks for this stage are going to be dependent upon the peer review determination from the peer review meeting stage. If the determination was Conditional Acceptance, then the follow up will simply be the moderator verifying the changes. If the determination was Repeat Review, then the peer review meeting stage is repeated, with an emphasis around the revised parts of the diagram(s).

6.0 References

Carlson, D. (2001) Modeling XML Applications with UML. Addision Wesley

Frankovich, J. “The Software Inspection Process”.
[http://sern.ucalgary.ca/courses/seng/621/W97/johnf/inspections.htm]

Major, M. “Using Guided Inspection to Validate UML Models”. [www.stickyminds.com]

Schmuller, J. (1999). Teach Yourself UML in 24 Hours. Sams Publishing

Weigers, K. (2001) “Peer Review Process Description”. [www.processimpact.com/pr_goodies.shtml]
http://www.izdsw.org/projects/FTR/dogma/

Weigers, K. (2001) “When Two Eyes Aren’t Enough”. [www.sdmagazine.com/print/documentID=16595]