User Community Modeling Language (UCML™) v1.1 for Performance Test Workloads

By Scott Barber

During the summer of 1999, I started trying to depict visually the often complex workloads that needed to be modeled for accurate performance testing. Because those early whiteboard sketches were so well received, I began using the same technique extensively to design and document performance test workloads. I later used the technique to demonstrate various workloads throughout my "User Experience, Not Metrics" series of articles. About the same time, I started jokingly referring to the technique as the User Community Modeling Language or UCML™. Much to my surprise, I received a lot of requests for the specification document for the language. Even more to my surprise, one of my readers (Nathan White, a retired military officer currently working for A. G. Edwards) wrote a specification document based on the articles and graciously sent me a copy.

Since time I have expanded my use of UCML™ to not just performance testing workloads, but to modeling all manners of system usage. From single user systems, to embedded system API's. This article describes UCML™ framework and discusses its use specifically for performance testing workloads. Additional examples can be found in many of my articles, presentations, and document samples on RDN and on my Web site. I hope you find this modeling framework as useful as I have over the years.

UCML™ Overview

UCML™ is a set of symbols that can be used to create visual system usage models and depict associated parameters. When applied to Performance Testing these symbols can serve to represent the workload distributions, operational profiles, pivot tables, matrixes, and Markov chains that performance testers often employ to determine what activities are to be included in a test and with what frequency they'll occur. The resulting diagrams can be used effectively for documentation, even of test plans, automated test designs, and as a basis for discussions and data-gathering workshops. They can be used to review and verify workflow and workload requirements with business analysts and users.

I developed UCML™ over the course of several years during my work on dozens of projects, with input from numerous clients, colleagues, and friends. My intent was to create an intuitive visual model of the often complex mathematical models that go along with performance-related testing, not as a replacement for the mathematical models but as a supplement to them. This visualization technique has proven to be at least reasonably intuitive to testers, application users, developers, managers, and business owners alike. In my experience, it has enabled productive discussions about important topics like "Is this activity really that common?" and "Why does activity B have a prerequisite of activity D? That's not right," rather than "How do I read this pivot table, again?"

While I draw most of my UCML™ models with SmartDraw, I still find myself hand drawing them in sketches, whiteboards, PowerPoint presentations, Microsoft Visio, and various other media. At this time, there's no "UCML™-compliant" drawing tool and I don't recommend any particular software product over any other to create these models – though I have created symbol libraries for Microsoft Visio and SmartDraw.

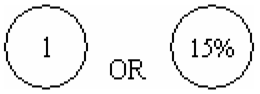
My sole intent in publishing this article is to provide access to a tool that performance test teams may find useful. This isn't an industry standard, nor has it been evaluated to comply with any industry standard or specification such as IEEE. You can use the symbols in whole or in part, and modify or improve upon them as you see fit. I ask only that if you have comments, improvements, or constructive criticisms, you send those to me by e-mail so that I can evaluate them for inclusion in any future documents about this technique.

Basic Symbols

UCML™'s basic symbols serve to represent possible usage through the system. In many cases, a usage path can also be thought of as an application session. One of the critical concepts of UCML™ is that the model represents system usage during a specific, finite period of time — often an hour. The fact that usage paths are the basis of UCML™ is part of what makes it powerful, because all members of the project team already have an understanding of how the system is used.

Quantity Circle

The quantity circle is used to answer the questions "How many…?" and "How often…?" Inside the circle is a number or a percentage.



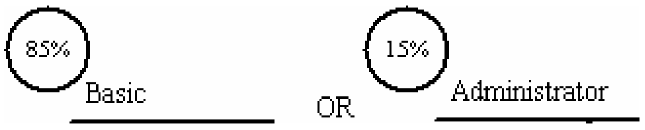
A number is used when an exact quantity is being modeled (for instance, one administrator). A percentage is used when a portion of the entire population of the user community, regardless of the size of that population, is being modeled.

Description Line

Description lines are solid horizontal lines that represent activities and user types. Each description line is labeled with one or more descriptors indicating the user type, activity, or navigation path represented. Enclosed in parentheses after the descriptor is the percentage of total users of that type, or the percentage of users performing that activity or following that navigation path over the period of time represented by the user community model. In some cases, it is more descriptive to use a quantity circle to depict the volume or frequency of a particular user type or activity. In general, use a quantity circle when multiple labeled activities or on the same line have the same volume or frequency and use parentheses when the volume or frequency applies to one specific activity.

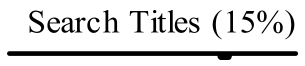
User Type Description Line

At the far left side of the model, a user type description line represents the type(s) of user(s) that will be accessing the application being modeled — for example, basic user, power user, administrator. The percentage associated with the user type is intended to indicate the proportion of all users who access the system represented by that type.



Activity Type Description Line

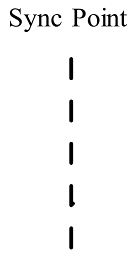
An activity type description line represents the activity or navigation path of each type of user through the system. The user actions/activities completed or the pages viewed on the navigation path are listed above the line. It's the modeler's choice whether to list each step of a particular activity or simply name the activity itself. The key point to note is that each horizontal line represents a straight-line path through that activity with no diversion.



This line may branch or merge (see below), representing a choice of navigation paths through the system. The percentage of users choosing a particular navigation path is indicated by a quantity circle at the beginning of the path. At any given point in the model, the aggregate of users on all navigation paths is 100% of the users visiting the site.

Synchronization Point

A vertical dashed line represents a synchronization (sync) point in the model. If the synchronization point is named, the name is shown above the line.



Synchronization points are used to depict convergence. There are two types of convergence that can be represented by the synchronization point symbol — convergence in time and in navigational location.

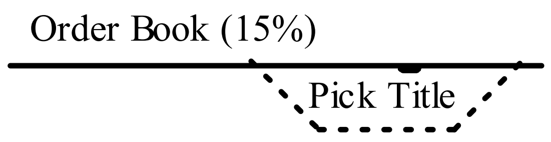
- A convergence in time means that the modeled user that reaches the symbolic sync point first will wait for all other modeled users in the system to also reach that point before proceeding. This is often useful when modeling messaging systems.
- A convergence in navigational location means that the modeled users all navigate past the same place in the application but not necessarily at the same time. A common example is the

login page of a Web site. In that example, all users must enter their credentials on the same page before being permitted to see the other pages. Identifying these navigational synchronization points often becomes useful when developing scripts, as they can serve as common start/stop points for scripts, scriptlets, functions, procedures, split scripts, and such.

Some people find it useful to annotate the type of synchronization point parenthetically after the name — for instance, "Sync Point 1 (time)" or "Home Page (nav)."

Option Box

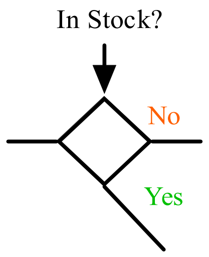
As users navigate through a system, they're often presented with an option or asked to provide data to the system that doesn't change their overall activity. These options are represented in the model by a dashed box suspended below the navigation path.



The option box should be labeled with a word or phrase that describes the option or data that varies from user to user. The specific data will be recorded in the spreadsheet below the model, described later in the section "Providing Supporting Information."

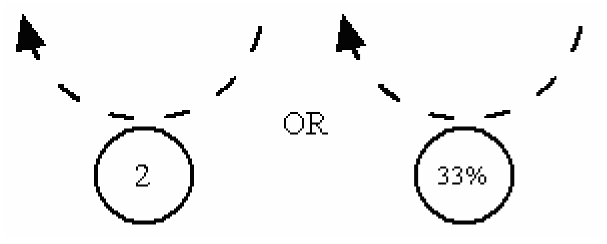
Condition

Conditions represent points in the model where users will change their navigation path based on the results displayed to them. In the example below, imagine a user trying to purchase a book. If after searching for the book, the user finds it to be in stock, the user follows the "Yes" path and purchases the book. If, however, the book isn't in stock, the user follows the "No" path and abandons the system.



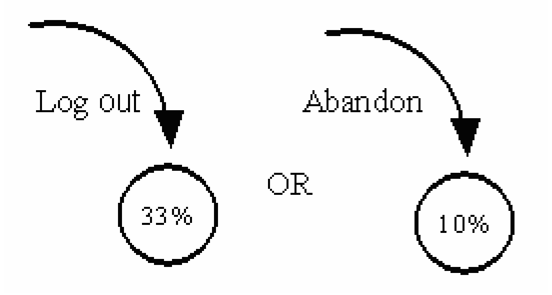
Loop

A dashed semicircle represents a loop on the navigation path, where the activity encircled by the semicircle is repeated. Either the number of iterations to be completed or the percentage of users to repeat the activity (not the percentage of all users represented by the model) is indicated in a quantity circle.

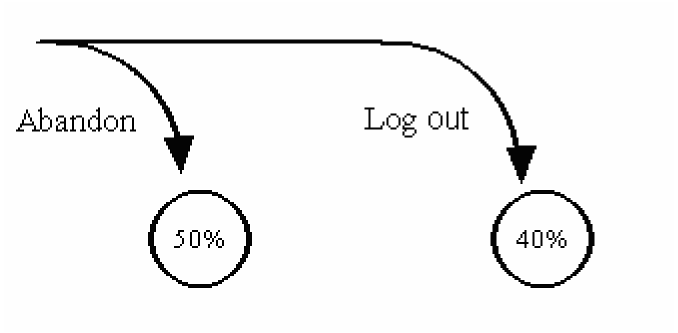


Exit Path

It’s often important to model users exiting the system at points other than the end of a navigation path. A downward-curving line pointing to a quantity circle indicates the percentage of users leaving the system at a specific point on the path. Remember that all users entering the system must be shown to leave the system. The type of system exit should be noted by a label — for instance, “Log out” (if the user leaves the system via the preferred means) or “Abandon” (if the user doesn’t leave the system via the preferred means).

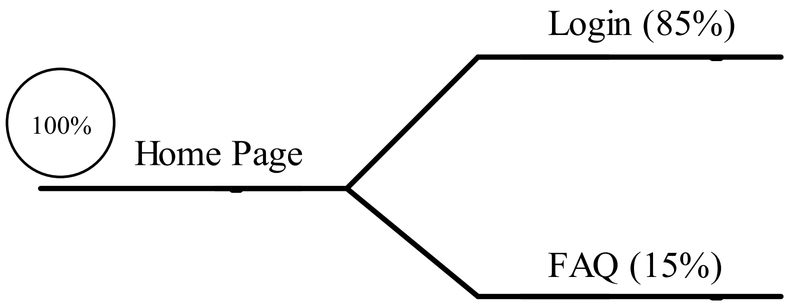


The example below depicts multiple exit paths from one navigation path. In the example, 50% of the users are abandoning the system and 40% of the users are logging out and thus exiting the system in the preferred manner. (The other 10% are exiting from another navigation path, not shown.)

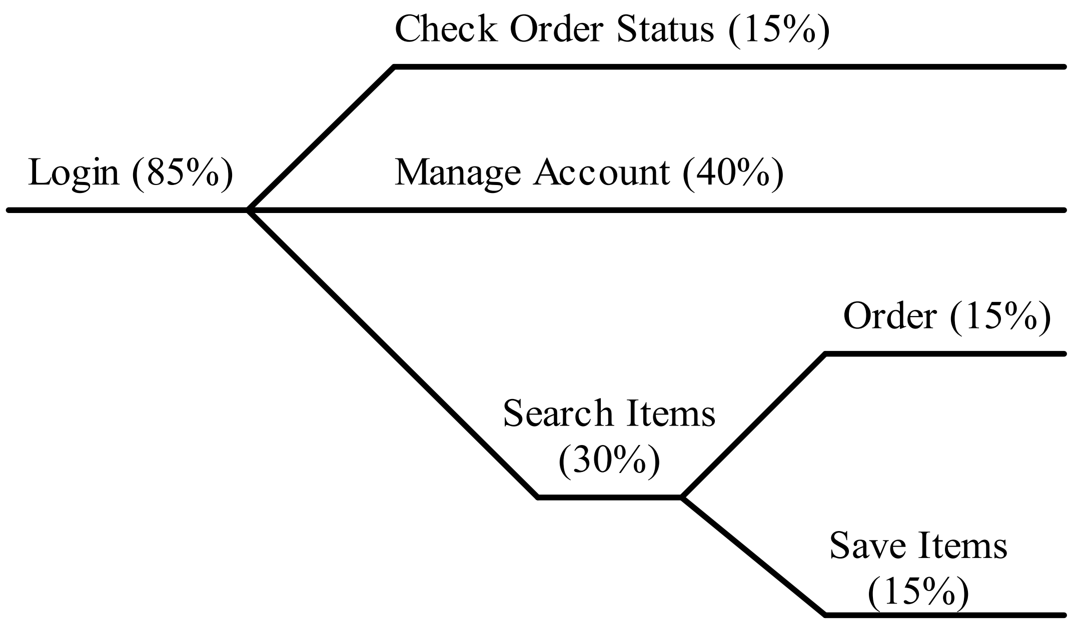


Branch

Most applications aren’t limited to straight-line navigation paths. A group of users may start out performing the same activity only to later branch off into different navigation paths. If we think of an activity line as a road on which a car is traveling, then a branch represents an intersection where the driver can choose which direction to turn. The example below shows a single branch where one of two paths can be selected off the home page. The sum of the percentages of users on the branches must equal the percentage of users on the page leading into the branches. In the example, the home page has 100% of the users and the sum of the percentages on the branches is 100%.

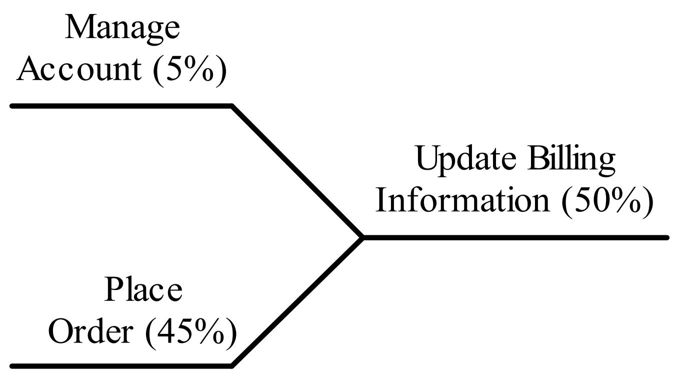


A single description line can have any number of branches, as shown below.

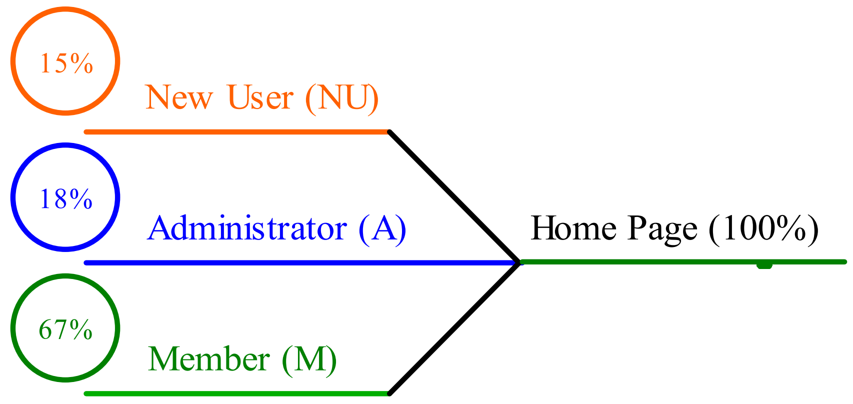


Merge

Applications with branching navigation paths often also have instances where different navigation paths merge back together. In keeping with the traffic analogy, this is similar to cars from different roads merging onto the same road. The example below shows two different paths converging on the Update Billing Information activity. Once again, the total percentage of users before the merge must equal the total percentage of users after the merge.



Another common use of the merge is to show different user types entering an application at the same point, as shown below. In this case each different user type is represented by a different color. This is done so that later in the model, activities that are specific to a particular user group can be identified by color. Each user type is followed by a parenthetical abbreviation that can also be used later in the model. This is particularly useful for models that must be represented in black and white.



Combining Symbols into a User Community Model

Combining the basic symbols is what gives UCML™ its power. It's through combining the symbols to represent all the possible user navigation paths through a system that visual models of entire communities are formed. Using the traffic analogy again, you can envision the lines in the model as roads and each modeled user as a car driving those roads. You can think of the places where two or more lines meet as intersections, synchronization points as traffic signals, and exits as off-ramps. Quantity circles show how heavily a section of "road" is traveled, thus creating a "map" of how users will traverse the application.

I'll build a UCML™ diagram for an online bookstore to demonstrate. Let's assume this is a new application so we have no existing usage data to analyze. Through a series of interviews we've collected the following information about the activities users conduct on the site and their anticipated relative volume:

- There will be four user types:
  - New Users (20%)

    - Members (70%)
    - Administrators (4%)
    - Vendors (6%)
- All user types enter through the home page.
- New Users and Members can conduct the following activities:
    - Search books by
        - Title
        - Author
        - Keyword
    - Add one or more books to their cart
    - Save their cart to order later
- New Users will also be able to choose from these activities:
    - Create an account (after account is created, become Members)
- Members will also be able to choose from these activities:
    - Log in
    - Update their account
    - Order items
    - Check order status
- Administrators and Vendors must log in from the home page and then start on the administration page.
- Administrators can choose from the following activities:
    - Add new books
    - Check order status
    - Update order status
    - Cancel orders
- Vendors can run the following reports:
    - In stock
    - Sales last week
    - Sales last month

The UCML™diagram in Figure 1 takes a first cut at representing this information and uses all of the UCML™ symbols.

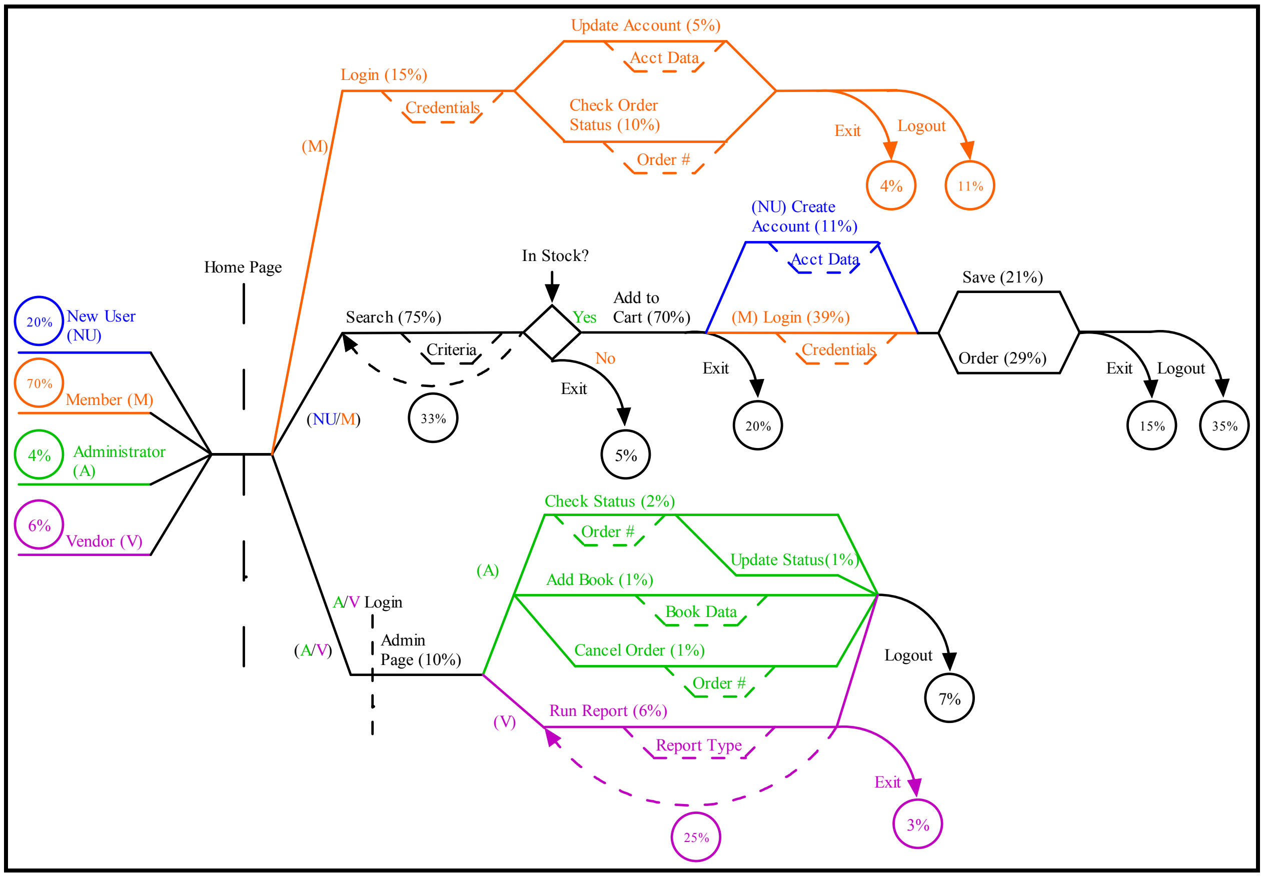


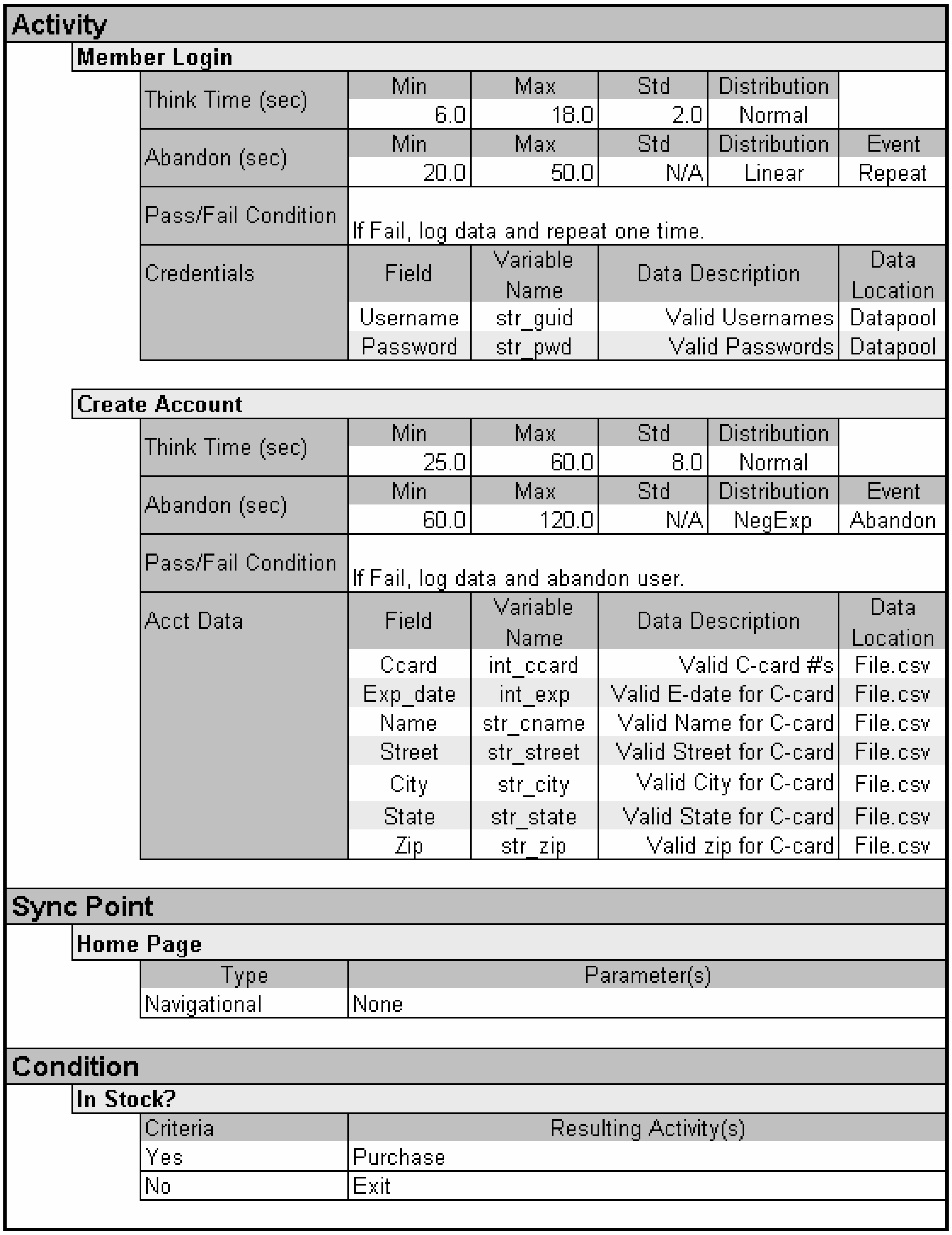
Figure 1: UCML™ diagram for an online bookstore

It's obvious that this diagram contains information not mentioned in the interview summary. For the purposes of this first draft, the information was estimated. In some cases the first draft will contain notes to reviewers or question marks to indicate that more information is needed. A close look will also reveal that not all information from the interview summary is included. This is also typical for a first draft. The diagram can now be circulated back to the people who were interviewed so they can validate or correct navigation paths and user volumes and add or remove activities.

Providing Supplementary Information

Unfortunately, the UCML™ diagram alone doesn't provide all of the information required to implement the depicted workload. To fully implement the user community model, several more pieces of information are needed. This information includes how long users may spend on a page, what data may need to be entered on that page, and what the actual conditions are behind a condition symbol. The information itself is mostly useless without a model, and the model can't be implemented without this supporting information.

Instead of trying to cram all of this information onto the diagram, it makes sense to supplement the diagram with a spreadsheet. This spreadsheet organizes supporting information by model component (activity, sync point, condition, and so on) so that it's easy to relate it back to the model. Figure 2 is a subset of the spreadsheet that contains the supporting information for the diagram in Figure 1. You can download the template for this spreadsheet if you want. The nature of much of this information — such as user think time, abandonment, and data variance — is discussed in more detail in the "User Experience, Not Metrics" and "Beyond Performance Testing" series.



Activity

Member Login

Think Time (sec)	Min	Max	Std	Distribution	
	6.0	18.0	2.0	Normal	
Abandon (sec)	Min	Max	Std	Distribution	Event
	20.0	50.0	N/A	Linear	Repeat
Pass/Fail Condition	If Fail, log data and repeat one time.				
Credentials	Field	Variable Name	Data Description	Data Location	
	Username	str_guid	Valid Usernames	Datapool	
	Password	str_pwd	Valid Passwords	Datapool	

Create Account

Think Time (sec)	Min	Max	Std	Distribution	
	25.0	60.0	8.0	Normal	
Abandon (sec)	Min	Max	Std	Distribution	Event
	60.0	120.0	N/A	NegExp	Abandon
Pass/Fail Condition	If Fail, log data and abandon user.				
Acct Data	Field	Variable Name	Data Description	Data Location	
	Ccard	int_ccard	Valid C-card #'s	File.csv	
	Exp_date	int_exp	Valid E-date for C-card	File.csv	
	Name	str_cname	Valid Name for C-card	File.csv	
	Street	str_street	Valid Street for C-card	File.csv	
	City	str_city	Valid City for C-card	File.csv	
	State	str_state	Valid State for C-card	File.csv	
	Zip	str_zip	Valid zip for C-card	File.csv	

Sync Point

Home Page

Type	Parameter(s)
Navigational	None

Condition

In Stock?

Criteria	Resulting Activity(s)
Yes	Purchase
No	Exit

Figure 2: Part of the spreadsheet that supports the UCML™ model in Figure 1

If done carefully, the UCML™ diagram and supporting spreadsheet are all you need in order to plan, design, and document any given workload distribution. And along with actual data files, this is all of the information you'll need to implement the workload distribution using your load-generation tool.

Summing It Up

The User Community Modeling Language (UCML™) is a simple yet powerful way to visually depict the workload distribution models and associated data necessary to create an effective performance test. The models created using UCML™ are easily understandable by users, managers, analysts, developers, and testers alike with little or no explanation. This allows for an ease of conversation and results in more accurate user models than simply presenting complex spreadsheets of data, thus increasing your confidence that your performance tests will accurately predict performance in production.

Revision History

- Ver. 1.1 created solely by Scott Barber in March 2004 and publicly released at StarEast in May 2004.
- Ver. 1.0 created solely by Scott Barber in November 2003 and publicly released in an article written for IBM-Rational on the Rational Developer Network in January 2004
- Ver 0.9 documented by Nathan White in August 2003 based on examples from *User Experience, not Metrics* articles publicly released on the Rational Developer Network throughout 2003.
- Ver 0.8 (undocumented) used solely by Scott Barber for various testing projects and *User Experience, not Metrics* articles throughout 2001, 2 and 3.
- Initial creation, 2000 with assistance from Chris Walters for a particular client engagement.

Acknowledgments

Thanks go to Nathan White for sharing with me the UCML™ specification he wrote for his company. Many of the examples in this article are largely unmodified from his specification. Thanks, Nate — I probably never would have formally defined UCML™ if not for your assistance.

About the Author

Scott Barber (WOPR Co-Founder) is the CTO of PerfTestPlus, Inc. (www.perftestplus.com). PerfTestPlus is a software testing consultancy that focuses on technical and otherwise challenging software testing, mentoring, methodology development and effective test resource utilization. Scott's particular specialties are testing and analyzing performance for complex systems, developing customized testing methodologies for individual organizations, embedded systems testing, testing biometric identification and security systems, group facilitation and authoring instructional materials. Scott is the author of the widely recognized article series' "User Experience, not Metrics" and "Beyond Performance Testing." Scott has presented at such venues as STPCon, WSE, WOPR, AWTA, PNSQC, StarEAST, the Rational User's Conference, SQE Testweek, local users' groups, and was invited to be a guest lecturer at MIT based on his article "Automated Testing for Embedded Devices". Additionally, he has numerous papers and articles published through IEEE, Better Software Magazine, Software Test and Performance Magazine, IBM Developerworks and numerous informational websites. He is a member of the Association for Software Testing, IEEE, American MENSA, the Context-Driven School of Software Testing and is a signatory to the Manifesto for Agile Software Development. You can view Scott's resume and a sampling of his work at www.perftestplus.com.