Interaction Process for ExamiNet
Development/Research Collaboration

February 22, 2002

Primary Contributors
Mark Bearden, Avaya Labs Research
Jake Garver, Avaya Labs Research Realization

Primary Contacts (Document Owners)
Bruce Hill, Avaya Services R&D
Jake Garver, Avaya Labs Research Realization

Other Contributors
Jeff Markel, Avaya Services R&D

Acknowledgments
The following people within Avaya contributed to the discussions that resulted in this document:
Matthew Agobozo, Harpreet Ahluwalia, Lorraine Denby, Bengi Karacali,
Chandra Kintala, Jim Landwehr, Jean Meloche, Karen Siebert,
David Stott, Ann Tvaroha
Table of Contents

1. Overview................................................................................................................................. 3
   1.1 Brief Project History ........................................................................................................... 3
   1.2 Two Team Structure .......................................................................................................... 3
   1.3 Scope of this Document ..................................................................................................... 3
   1.4 Related Documents .......................................................................................................... 3

2. Responsibilities Overview ......................................................................................................... 4
   2.1 ExamInet Development Group ......................................................................................... 4
   2.2 ExamInet Research Group ............................................................................................... 4
   2.3 Symmetries and Asymmetries in the Collaboration ........................................................... 4

3. Research-to-Development Migration Process ......................................................................... 5
   3.1 Principles of the Migration Process ................................................................................... 6
       3.1.1 Ensuring Platform-Compliance at Handover ............................................................. 6
       3.1.2 Providing Sufficient Documentation at Handover ....................................................... 6
       3.1.3 Collaborating to Determine Ownership and Schedules ............................................. 7
   3.2 Detailed Steps in Handover Process ................................................................................... 8
       3.2.1 Basic Research Stage (Pre-Migration) ...................................................................... 8
       3.2.2 Transition from Basic Research to Research Realization .......................................... 8
       3.2.3 Research Realization (RR) Stage ............................................................................... 9
       3.2.4 Transition from Research Realization to Handover ..................................................... 9
       3.2.5 Handover-(H) Stage .................................................................................................. 10
       3.2.6 Transition from Handover to Maintenance ................................................................. 10
       3.2.7 Maintenance (M) Stage ............................................................................................. 10
       3.2.8 Transition from Maintenance to Full Development Ownership .................................. 11

4. Development-to-Research Migration Process ......................................................................... 11

5. System Architecture ................................................................................................................. 11

   6.1 Distinct but “Orbiting” Code Bases ................................................................................... 12
   6.2 Ownership of Modules in Code Bases .............................................................................. 12
   6.3 Responsibility for Defining Bug Reports .......................................................................... 13
   6.4 Tracking Changes Made by the Other Group ................................................................... 13
   6.5 Support for Export of Documented Code Modifications .................................................. 14

7. Full Listing of Responsibilities ................................................................................................ 15
   7.1 All Development Group Members ..................................................................................... 15
   7.2 Bugzilla Liaison(s) in Development .................................................................................. 15
   7.3 All Research Members ....................................................................................................... 15
   7.4 Sablime Liaison(s) in Research ......................................................................................... 16

8. Documentation Requirements .................................................................................................. 16

Appendix A: Details of Change Management Process ................................................................ 18
Appendix B: Sablime System Reference for Research Members .................................................. 25
• Brief statement of the requirements satisfied by the module and/or references to other documents defining requirements
• High-level block diagrams of objects along with their interactions (UML graphical syntax is preferable)
• Formats of any messages used for interprocess communication
• Schema and semantics of any database tables created or relied upon
• Descriptions of any APIs exposed to other modules
• Description of the how the source files are structured, and where specialized documentation such as README files and javadocs, if they exist, can be found
• Instructions for how to compile and build the module code
• Description of any strategies employed, and code developed, for unit testing
• Dependencies on other modules or platforms (language versions, database versions, third party packages, etc.)
• Major design decisions that were inherent in the module, a brief justification for how these decisions were settled in the present implementation, and an identification of any design issues that might need to be resolved differently in the future
• Listing of known deficiencies and bugs, and missing features
• If the module has a GUI, the documentation should include screen shots of the major GUI panels along with brief functional descriptions of each major panel

Responsibility for converting documentation to a standard document format is left to the development group.

3.1.3 Collaborating to Determine Ownership and Schedules

To a changing extent, both groups have a committed stake in each migration item during the whole migration process. In contrast, the research group totally owns work items before they are classified (by both groups) as migration items, and the development group totally owns fully migrated code modules. At all intermediate stages, there is shared ownership and shared responsibility for resource scheduling and prioritization. The reason for making this clear is to avoid the assumption by development that prior to moving code into the development repository the research group are solely responsible for scheduling resources for and prioritizing the migration work. Conversely, the research group must not assume that once code has been transferred from research to development, only the developers are responsible for completing the migration work.

In the early part of the migration process, the research team takes a primary role of providing implementation and scheduling support (i.e. making time-to-completion estimates) while the development team plays a secondary role of evaluating and prioritizing the work item. As the migration to development goes proceeds to the later stages, the development team progressively takes over the primary role of implementation and scheduling, until they eventually own all aspects of the item. In the middle of migration, there are times when both groups must contribute to and approve time-to-completion estimates. During these times both groups must explicitly schedule and commit the time and personnel resources needed to transfer code ownership.

During the whole migration process, the resources devoted to any item will be communicated to both groups, as well as best estimates for completion of milestone events. The whole list of migration items must be tracked on some type of formal, or informal, schedule that can be accessed by both groups, e.g. on project web pages. For each tracked item, there should be an indication of the item status (e.g. How many requirements does the code satisfy? How much testing has been done?), estimates of time needed for the steps identified as necessary for module handover, and an indication of who is responsible for each handover step.
3.2 Detailed Steps in Handover Process

Figure A-1 presents the migration process as a three-stage pipeline. The sections that follow discuss the stages and transitions shown, from the left-hand side to the right. The stages are characterized by work that goes on during that phase, and the transitions are characterized by the preconditions for moving to the next stage.

3.2.1 Basic Research Stage (Pre-Migration)

This pre-migration stage includes work done by researchers to explore useful features for the product, including by talking with the development team, interacting with present and potential customers, and performing experimental investigations. In general, the output of this stage is not code that is ready for use in a product, and the output of these activities may not include any useable implementations at all. Items worked on in the basic research stage are periodically considered for their potential contributions to future ExamiNet product releases. Items that seem to have promise but which are not suitable to be owned initially by the development team can be considered for promotion into the migration pipeline (research realization stage), once resources can be identified to carry them through the early handover stage in the migration process pipeline.

3.2.2 Transition from Basic Research to Research Realization

When all of the following conditions are satisfied, and only then, will the work item be classified as a migration work item (and as a research realization module) in shared planning documents.

- The item has been defined in writing, at a high level, as a module that is understood by both research and development.
- The development team has communicated agreement that the module is worth their effort to evaluate for handover, and that they will accept ownership of the module when it is ready for handover (contingent of course on a successful prototype implementation and the continued availability of anticipated project resources).
- Research has identified a primary owner in research who is responsible for defining the requirements and implementation.
- Optionally, a secondary owner in development is identified (e.g. for consulting on platform and integration issues).
- One of the following two conditions must be satisfied:
  - Research commits to hand over, at the end of the realization phase, a realization implementation that is compatible with the current development platform and architecture
Figure 1. Workflow for bug fixes that are discovered by research group.

Figure 2 shows the other side of the process, that is, how bugs discovered by the development team propagate to the research code base. This is simpler than Figure 1, as the developer group is not required to handle any “pushing” of MRs to the research group. Rather, the research group takes full responsibility for reviewing MR reports, which must be made available on a regular basis, and then replicating the desired MRs in the Bugzilla system.
Figure 2. Workflow for bug fixes that are discovered by development group.

Figure 3 shows the standard MR life cycle, for reference. At the two states highlighted in gray, “accepted” and “assigned”, a decision can be made by the development group to delegate responsibility for an MR to an appropriate party in the research group. The process that is followed in that case is shown in Figure 4. If the member of research performing the fix is a user of Sablime, then the MR is simply reassigned as shown by the right-hand half of the flow. If not, then the MR is assigned to the research group’s Sablime liaison, who manages a research-internal reassignment to the research member taking responsibility for the MR. This second case is shown by the left-hand side of the flow in Figure 4. (Since this reassignment within the research group will not be automatically visible through MR status, it would be helpful if the MR record is annotated to indicate which researcher is really making the fix.)

If the delegated fix for the MR is one that also needs to be applied to the research code base, then the research member who has been reassigned the MR carries out a “splitting” of the work item into a Bugzilla entry (which might already exist as an independently reported bug). The lower part of Figure 3 details this flow into the separately defined “Bugzilla/CVS Life Cycle” subprocess. After the bug fix is completed and the entry is closed in Bugzilla, then the original MR owner in the development group is referred—via a change in MR status made by research—to the Bugzilla entry that contains the fix. The developer must then extract the fix and merge it into the Sablime code base in order to close the MR.

---

5 Simplified Sablime state diagram was obtained from http://sj.usae.avaya.com/~sad/sablimehelp.html.
There is one other case that is detailed by the central part of the flow diagram in Figure 3. This is when the MR fix is delegated to research but the fix is irrelevant to the research code base—it needs to be applied *only* to the development base, that is. In this case the Bugzilla splitting is bypassed as shown.
“dcreate” calls “fcreate” and automatically assigns

d create (Any User with PTSid)

created

accept (MRA)

accepted

activate (MRA/GA)

spawnmr (GA)

assigned

activate (MRA/GA)

assign

spawned

submit (AD)

subtome (GA)

nochanged

deferred

subtodef (GA)

Optional Test States

reject (AT)

submit (AD)

reject (AT, TT)

approve (AT)

(MRA)

approved

closemr (MRA)

closed

Figure 3. MR Life Cycle
Figure 4. Workflow for Development-to-research delegation of MR fix.