RAPTOR QA Test Strategy & Results

Contents

Figures

Tables

Philosophy

The RAPTOR team is focused on applying best practices in architecture and development and coupling that with ongoing testing against newly developed code until quality software from a technologist's and end customer's perspective results. Our agile testing approach is built upon the philosophy that testing needs to adapt to rapid development/feedback cycles. The use of best practices at every step is built upon the philosophy that this brings known quality elements into the project.

In particular, the team believes that utilizing popular open source tools and frameworks such as PHP and Drupal brings a quality multiplier to the agile development process of RAPTOR. These tools and frameworks, their construction, and their implementation details, among others, are continuously being evaluated, examined, and discussed in open forums. The team regularly monitors activity and announcements that relate to the tools being employed in RAPTOR.

In general, the team subscribes to the notion that regression testing is appropriate and necessary whenever anything is changed in an implementation element or significant deployment factor. The extent of the regression testing is decided by key stakeholders through consultation with technical domain experts to assess risk depending on the phase of the project at that time. The RAPTOR system is implemented with an eye toward the expectation that regression testing is a critical part of establishing and maintaining confidence in the system.

Identifying and Categorizing Key Quality Risk Areas

Many elements of the RAPTOR system can be individually identified as crucial to a successful implementation and deployment; but as a clinical system, the accuracy of RAPTOR use case support is paramount. Categorizing risk areas is fundamentally driven by the requirements captured in the FRD and is supplemented by continuous insight refinement through agile process interaction with stakeholders.

Critical RAPTOR Features

Several features have been specifically identified as critically significant in the RAPTOR implementation and are listed in Table 1.

Table 1 - Quality features of RAPTOR identified as Critical

ID	Description
1	Only authorized users may use the system
2	Authorized users are only able to perform actions associated with their designated role
3	Patient data displayed by RAPTOR is an accurate reflection of what was provided to it by systems or record.
4	Data input by users to process patient orders is properly recorded by RAPTOR
5	All significant actions taken by users of RAPTOR are properly recorded for audit purposes
6	Patient history summaries are consistent with documented specifications
7	Patient allergy warnings are consistent with documented specifications
8	Patient medicine contra indications are consistent with documented specifications

Formally Categorizing Potential Enhancements

The RAPTOR team believes that clearly categorizing enhancements discovered during the agile system development process helps focus all team members channel their focus on delivering the highest quality system possible. Having unambiguous terms to categorize potential enhancements is fundamental to the testing process portion of quality assurance in that it helps the team appropriately prioritize discussion and remediation. The terms used by the team to categorize the nature of a potential enhancement are listed in Table 2.

Table 2 - Terms used to Classify Potential Enhancement Impacts

Impact Category	Description
S – Evolving out of the Sandbox	Listing of functionality potentially needed to help transition out of the sandbox.
I – Innovation Enhancement	Listing of additional innovative functionality requested by Dr. Medverd and Dr. Anderson to enhance existing RAPTOR functionality. Typically, these requirements require interfacing with new systems and incorporate new data and design into the existing RAPTOR framework. These requirements will be packaged into 2012 enhancement request and will require more resources to realize.
B – Bridge Enhancement	Listing of additional functionality requested by Dr. Medverd to improve existing RAPTOR functionality. Typically, these requirements are available within the current framework, and are enhancements to business logic that utilize the existing design. These requirements can be packaged into a smaller bridge request and can be realized with fewer resources.

Quality Assurance focus on RAPTOR specific Risks

The key risk areas of particular relevance to RAPTOR are summarized Table 3.

Table 3 - Key Quality Risk Areas and Impact on System Candidate

Quality Area	Impact	Risks
Use Cases Accuracy	Usability and applicability of system in target domain	• Incomplete or inaccurate use cases
Software Architecture	Complexity of ensuring performance, reliability, and maintenance	• Inappropriate or impractical architecture
User Interface Design	Usability and appeal	• Confusing to use • Not appealing to end users

Quality Area	Impact	Risks
Code Implementation	Performance, reliability, and maintenance; patient safety	• Runtime bugs • Undocumented code
Formal Technical Test Plan	Confidence at start of UAT; patient safety	• Incomplete • Inaccurate
UAT Testing Process	User acceptance; patient safety	• Incomplete • Inaccurate

Mitigating Quality Risks for RAPTOR Success

The quality risk areas specific to this project can be mitigated through careful attention to quality in the categories identified in Figure 1.

Table 4 - Key QA Categories

Process Category	Direct Quality Impact Area	Quality Assurance Actions
Use Cases	Design	• Develop initial cases from evaluation of existing processes • Engage subject matter experts in refinement of use cases
Software Architecture	Complexity of ensuring performance and reliability	• Employ established principles • Leverage quality open source tools • Prototype to validate approaches
User Interface Design	Usability and appeal	• Employ trusted principles • Get feedback from stakeholders
Code Implementation	Performance and reliability	• Best practices in development • Unit testing • Code reviews
Formal Technical Test Plan	Confidence at start of UAT	• Provide clear repeatable steps that prove correct functioning in critical areas • Document technical metrics and measured performance against those metrics
UAT Testing Process	User acceptance	• Customer driven and initiated processes to validate the system in the designated UAT environment

Use Cases

The use cases are captured in the FRD which was formally accepted in the summer of 2011; but have continued to improve through meetings with subject matter experts. As the software design has matured, the refinements of the use cases have focused on clarifying ambiguities and recognizing opportunities for delivering a stronger system within the resource and time constraints of the project. The following test plan description is based on the format described in IEEE 829-1998 Test Plan template.

2.1 DESCRIPTION

The Test Case description describes the test case and the individuals involved in the testing. It may include diagrams depicting the interaction between individuals and the different elements being tested.

It may include Resources, Inclusion/Exclusion points, and Test Conditions. Resources describe the individuals involved in the testing, their responsibilities, and their association with the test case. A precondition is the state of the system that must exist before a test case can be performed. A post condition is a list of possible states the system can be in immediately after a test case has finished. Inclusion/Exclusion points describe other test cases that may be included or excluded in the act of executing this test case.

FLOW OF EVENTS

The flow of events that would be expected in normal conditions as well as any potential alternate flow of events, and exceptions/errors that may be expected.

Software Architecture

The RAPTOR team collectively has decades of practical and academic experience in delivering software from inception to implementation in small and large scale environments. There are many aspects to quality software architecture; some of the key principles emphasized in RAPTOR are listed in Table 5.

Table 5 - Key Architecture Principles in RAPTOR

Key Principle	Description
Modularization	Functionality is logically grouped into modules which can be independently described, unit tested, and developed. This is a materialization of abstraction concepts
Minimal Coupling	Good modularization has only those couplings between modules which were necessary and appropriate. Every additional cross dependency becomes an additional element of development and maintenance complexity.
Paths for Extension	During the agile development process team members and consulted SMEs may discover and identify feature areas for possible future expansion. The team facilitates enhancement in such areas wherever possible by supporting clear APIs and simple plug-in architectures.
Portability	The team chooses mature tools and framework components

	wherever possible which are not locked into one vendor specific operating system or product line.

Modularization

A key principle in the fundament architecture of any system is to organize the system into blocks of manageable effort. The design should take into account the available tools and frameworks, technical resources, and SMEs available on the project. The fundamental modularization employed in RAPTOR is illustrated in Figure 1.

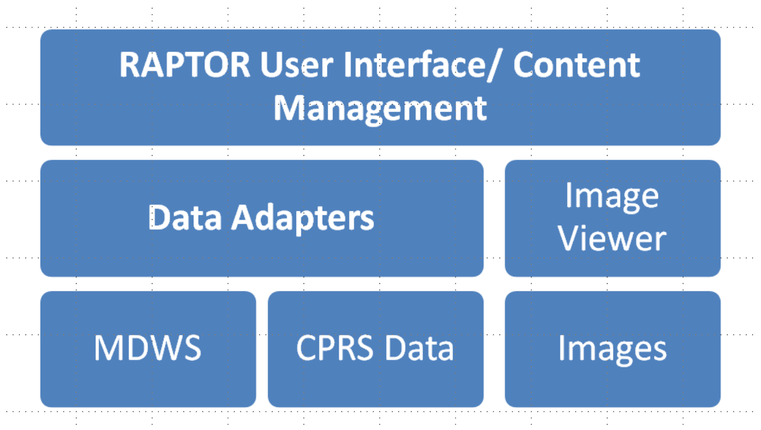

Figure 1 - Fundamental Modularization of RAPTOR

Modularization is applied as a recursive technique within the RAPTOR project to organize subparts logically as such organizations make themselves apparent to the developers and architect.

The high level RAPTOR modularization strategy was prototyped in the mockup delivered and accepted in August of 2011. The iterative nature of agile development provides for a continuous improvement and refinement of both high level and subsystem modularization as development progresses.

Minimal Coupling

The effort required to validate the correct interaction between modules and blocks is impacted by the number of couplings that exist between them. With fewer couplings, there is less complexity that needs to be tested and thus results in a higher level of confidence that communication between blocks and modules is properly testable and is functioning to specifications.

Paths for Extension

The Drupal framework provides a clear architecture for extension which fundamentally consists what they label “modules” and what they label “themes”. Users of the system can simply extend the framework in innovative ways by plugging in their own modules and themes in an API compliant manner. The RAPTOR system is implemented in Drupal by leveraging this feature of the framework; and within those module and theme elements provides RAPTOR specific areas for extension.

Both Drupal and RAPTOR leverage a folder hierarchy approach to simplify the extension process.

Portability

The RAPTOR system is engineered to run on any popular operating system that supports the tools and frameworks integrated into the design.

The two key operating systems which have specifically been targeted by the team are Unix/Linux and Windows. Customers are free to install the RAPTOR system on either operating system which is best supported and available within their operational environment.

- Open source and proprietary versions of Unix and Linux are both popularly available. RAPTOR can be installed and configured to run on these operating systems.
- Windows is proprietary and has a licensing cost. However, it is already installed and popular in many environments. RAPTOR can be configured and run on these operating systems.

As part of the RAPTOR team philosophy that leveraging popular mature open source tools and frameworks can produce a higher quality innovative product at lower cost, the RAPTOR system itself will not be encumbered with any embedded proprietary tool or component. It does not exclude proprietary plug-ins, but does not require them for fundamental operation of the system use cases.

NOTE	The RAPTOR system interacts with external systems that are themselves not necessarily open source or license free. For example RAPTOR interacts with the VA data stores through VA provided open model web services (MDWS). However, MDWS itself interacts directly with systems that are not open source and not license free. A key notion in RAPTOR is to buffer RAPTOR from proprietary systems via an open model buffer layer. This buffer allows proprietary system to be enhanced or replaced while only impacting the buffer system and not RAPTOR itself.

Code Implementation

The developers of the RAPTOR system have decades of industry and academic experience in the design and development of code using best practices. Coordinated application of best practices at the code level is one of the quality assurance strategies employed during the agile development process of RAPTOR. This technique improves analysis and maintainability of the code.

Developers of subsystems and modules are responsible for development and execution of unit tests to validate that those subsystems and modules are operating to specifications. The Drupal framework itself has a unit test paradigm which is also leveraged in the RAPTOR system.

A key aspect of ensuring quality in the agile process is to employ periodic peer code reviews and integration sessions. Elements of the development process become collaborative as an outcome of this process. The collaboration and review increase the opportunity to identify weak areas early and improve them while progressing forward.

Since the RAPTOR system is being collaboratively developed, the following points are of particular consideration to the programmers during the implementation cycles:

- Establish clear APIs between developers

- Implement redundant safety checks between modules where practical
- Throw errors when patient safety would otherwise be compromised
- Create re-usable unit and integration tests

User Interfaces

User interfaces involve elements of art and can be very subjective. For this reason early involvement of customers is key to delivering a successful user interface into RAPTOR.

The RAPTOR team demonstrated a navigable RAPTOR mockup in August of 2011 and with feedback arrived at an understanding of basic principles and user interactions appropriate to this system in a clinical environment. Continuous stakeholder involvement is the key to assuring a successful reception.

Table 6 - Key UI Principles Employed in RAPTOR

Category	Description
Clear identification of user	Take all reasonable visual cue measures to ensure that there is no question the user currently sitting a workstation is the user recognized by the system as interacting with the system from that workstation.
Clear identification of patient	Take all reasonable visual cue measures to ensure that there is no question which patient is currently being viewed and processed through RAPTOR.
Clear identification of Study	Take all reasonable visual cue measures to ensure that there is no question which study is currently being viewed and processed through RAPTOR.
Same thing in same place	For those data elements that appear across various pages, ensure that all reasonable efforts are taken for that data to appear in the same style and relative location as it did before.
Common things easy	Take all reasonable steps to ensure navigation and expected user actions are intuitive and simple for those things which are expected to be common occurrences and user needs.
Uncommon things possible	Take all reasonable steps to provide a mechanism for users to take actions which are recognized as necessary and appropriate but have been deemed as infrequent or unlikely to occur.
Minimal navigation clicks	As a general strategy, users should be able to navigate to desired action targets through the minimal number of clicks necessary to get there in a clear an unambiguous manner.
Unambiguous labeling	Data presented in RAPTOR should not be ambiguous. User interface conventions employed to disambiguate data include the following: • Clear labels on tables from the perspective of expected users • Clear labels on fields from the perspective of expected users • Provide additional hover-text when space is constrained or additional full text on the regular display is generally not needed for the average user
Critical Confirmations	Where critical decisions are made, the user should be asked to clearly confirm that is the action they are requesting

Formal Technical Test Plan

The formal technical test plan will cover the following two requirement areas:

- Functional Requirements
    - Demonstrate expected output occurs given known inputs
    - Demonstrate actions are properly constrained by assigned roles
- Non-Functional Requirements
    - Demonstrate only authorized users can use the system
    - Demonstrate expected performance criteria is satisfied under reasonable conditions

Security Testing

RAPTOR is currently only accessed via an encrypted laptop. It is protected by Guardian-Edge Authenti-Check responses. Once the collaborative "sandbox" environment has been established, it will be maintained and secured by the innovation group.

User Acceptance Testing

The User Acceptance Testing (UAT) is driven directly by the radiologist/innovator with input, as requested, from the RAPTOR technical team. The UAT Results will be documented in the table below.

Table 7 – UAT Results

FRD #	Functionality	Pass/Fail	
		#	%
2.1	Non-Functional (Technical, Usability) Requirements		100
2.2	Users		100
2.3	Access Roles		100
2.4	RAPTOR Workflow Status		100
2.5	Approving a Protocol		100
2.6	Protocol Worklist		100
2.7	Protocol Listing		100
2.8	Patient based Protocoling		100
2.9	Contrast		100
2.10	Patient Pre-Scan Preparation Regimens		100

Enhancements

All UAT enhancements will be categorized a SEV level as shown in the table below.

Table 8 - Enhancement Severity Chart

SEV Level	Enhancement Severity Level
S	Evolving out of the Sandbox – Listing of functionality potentially needed to help transition out of the sandbox.
I	Innovation Enhancement – Listing of additional innovative functionality requested by Dr. Medverd and Dr. Anderson to enhance existing RAPTOR functionality. Typically, these requirements require interfacing with new systems and incorporate new data and design into the existing RAPTOR framework. These requirements will be packaged into 2012 enhancement request and will require more resources to realize.
B	Bridge Enhancement – Listing of additional functionality requested by Dr. Medverd to improve existing RAPTOR functionality. Typically, these requirements are available within the current framework, and are enhancements to business logic that utilize the existing design. These requirements can be packaged into a smaller bridge request and can be realized with fewer resources.

System and Regression Testing Enhancements

Table 9 - UAT Testing Enhancements

System and Regression Testing Enhancements	Total Opened	Total Deferred	Total Closed
Critical	0	0	
Major	0	0	
Average	0	0	
Minor	0	0	
Enhancement	0	0	
Totals	0	0	

Performance Tuning

Performance tuning of the RAPTOR system will occur at the module level as modules arrive into a state of significant feature completeness.

Glossary of Abbreviations

UAT	User Acceptance Testing
QA	Quality Assurance
SME	Subject Matter Expert

Revision History

When	Who	Description
2011-10-30	Frank Font	Initial draft