

Agile Testing and the Banking/Finance Domain...should we do it?

Sharon Robson

Software Education Associates Ltd

21 July 2008

Banking/Finance and Software Testing have a lot in common – it is all about risk! Even more specifically it is about the clear understanding of the risks being undertaken. The Banking/Finance domain is focused on risk analysis of their clients and the investments that the institution makes, whereas Testing is about risk analysis and reporting on the Quality of software (applications) under test.

In the IT world, the Banking/Finance sector is perceived as being conservative; characterised by being not quick on the uptake of new approaches or techniques, very risk adverse, as well as being particularly price sensitive. This has led to a reliance on the older, more “traditional” development approaches such as the Waterfall or V-Model. These approaches have been considered to be robust and conservative, and although not cheap, have been used extensively in the Banking and Finance domain. However times are changing and now there is the push for getting more products to market and getting them there faster and much more cost effectively^[1].

To understand how to approach testing in the Banking/Finance Domain, there needs to be a model of the domain. There are many types of Banking and Financial applications within this industry vertical, so I will generalise (see Figure 1).

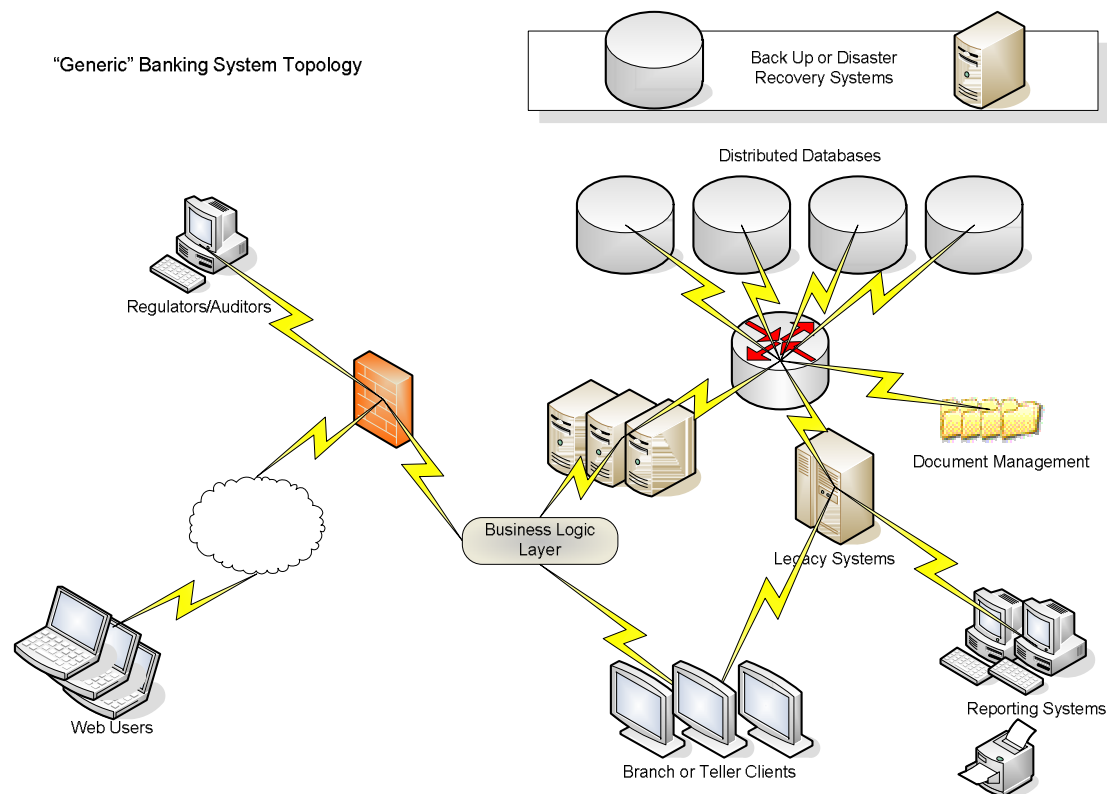


Figure 1

If I postulate that the banking and finance domain is typically characterised by multi-tier functionality^[2], which in itself implies a complex “system of systems”^[3]. The

systems themselves tend to be the result of bespoke development, maybe internal IT groups, or outsourced ^[4]. Either way, the systems are theoretically well documented, however the documentation is either grossly out of date (as with most documentation) or has been lost in the myriad of buy-outs, mergers or take-overs also typical in the Banking/Finance domain.

Another characteristic of the Banking/Finance systems is the preponderance of interfaces; bank to bank, bank to branch, bank to web-users, just a few obvious examples. These days there is highly probably a Web Interface as well as many specific user interfaces. The result is many users on distributed systems each with a different business need and approach to the system.

To further compound the complex situation, the transactions at both the data level, and the User Interface (UI) can be complex, often requiring business logic to be applied at the client side and server side and, as is happening more and more now, in a Service Layer or Business Logic Layer ^{[1][4]}. Most if not all of the activity through the system will require Secure Transactions, both batch and synchronous (real-time).

Another key characteristic, especially when it comes to the sale of various Banking/Finance “products” is the huge number of Business Rules that must be maintained and monitored. There is little margin for error, as this is the fundamental requirement of a banking system! This is reinforced by the need for extensive regulation and auditing functionality. Reporting is often the way that audits and confirmation of adherence to regulations are implemented and is always a key component of Banking systems, even more so now that there are more stringent requirements as a result of the implementation of regulations such as Sarbanes-Oxley and Basel II. Effective Document Management is key to the retention of important data and is usually implemented to assist with reporting and auditing requirements, as well as implementation of the Business Rules.

At an infrastructure level, Banking/Finance systems tend to have robust back up procedures, disaster recovery systems, and high availability requirements. Coupled with this are massive storage systems, requiring speedy transactions and rapid data transfer. The complicating factor with this infrastructure is that it often involves integration of myriad disparate legacy systems as the result of mergers and take-overs which are typical of this industry ^[5].

To summarise the generic Banking/Finance system, I see the following key characteristics:

- Multi-tier Functionality
- Systems of Systems – Large Scale Integration
- Bespoke Development
- Lacking Documentation
- Many Interfaces
- Distributed Systems
- Disparate Use Cases
- Complex Transactions
- Secure Transactions (Batch and Real Time)

- Speedy Transactions
- Complex Business Logic (Client and Server)
- Auditable
- Document Management
- Back Up/Recovery/Disaster Recovery
- Massive Storage Systems

Now the application domain has been defined, let us consider how we can approach testing it.

“Agile” development methods have been developed to try and eliminate the issues associated with the more traditional development approaches. The Agile Manifesto [6] focused on many aspects of software development, but its terms of quick delivery it can be highlighted by the following two principles:

- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.

For implementing new systems in any domain, even the Banking/Finance sector, an Agile approach to the testing makes a lot of sense. No matter what Software Development Lifecycle (SDLC) is followed, testing continues to be about analysis and reporting on the Quality of the System Under Test (SUT). In Agile development approaches this lends itself to a risk-based approach from the business perspective, contrasting with more traditional development approaches. This risk analysis is done with the Business Users, by the Business Users. This seems to be ideally suited to risk adverse, yet economically constrained domains, such as Banking/Finance.

Agile test techniques should not be taken on without due consideration to the dynamics of the specific project. I will consider the generic characteristics of the Banking/Finance systems as defined above and if, in general, they are suitable for an Agile Testing approach. With this knowledge you can then consider the particular project you are focusing on.

Characteristics of Banking/Finance Systems	Suitable for an Agile Testing Approach?
Multi-tier functionality	Yes – for new functionality, depending on what aspects are under test.
Systems of Systems – Large scale Integration	Depends on degree of rigour required by the Testing
Bespoke Development	Yes
Lacking Documentation	Yes
Many Interfaces	Yes – when User Stories are well defined
Distributed Systems	Yes
Disparate Use Cases	Yes – User Stories defined and agreed.
Complex Transactions	Yes – automated Unit and Integration Tests
Secure Transactions (Batch and Real Time)	Yes – Automated testing

Characteristics of Banking/Finance Systems	Suitable for an Agile Testing Approach?
Speedy Transactions	Yes- Automated Unit and Integration testing.
Complex Business Logic (Client and Server)	Yes – Automated Unit and Integration Testing
Auditable	Depends on the testing standards that may be applied: No – light documentation may not work for this. Yes – if the unit and integration tests are automated, traceable and under source control.
Document Management	Depends on degree of rigour required by the Testing
Back Up/Recovery/Disaster Recovery	No – complete and structured test coverage must be available
Massive Storage Systems	No – complete and structured test coverage must be available

As noted previously, Agile is all about access to the Business User (customer), and rapid turn around of software which can be done via incremental and rapid (small batches of functionality) development cycles. Incumbent in this approach is design collaboration and review, less documentation, automated testing, early and often code delivery, unit and smoke tests before “independent” testing, and importantly, business experts defining and participating in Acceptance tests ^[7].

So, the key question is “What are we testing?” or more specifically “What is the objective of this test?” If we are focusing on Large Scale Integration ^[8] for example, which is often a characteristic of Banking/Finance applications, theoretically this is technically difficult and requires extensive knowledge of all the systems at the interface definition level. This information may not be available for all the systems in the Integration scope; thus an Agile testing approach could work very well here. An Agile testing approach would define the User Stories which in turn define how the various systems will be required to behave in specific circumstances – without needing the in-depth documentation.

The Testing mindset of applying positive and negative test conditions will assist in driving out many undefined requirements, so by working with the Business at all levels the requirements for the interfaces and integration can be defined and tested. Although there are technical integration requirements, the key “needs” still have to be defined by the Business. The tests should and could even be defined in such a way that they are automated and retained for ongoing regression and integration testing.

If the project is adding functionality to an existing system, testing the new functionality under an Agile Development approach is straight-forward, but how do we make allowances for the potentially extensive Regression testing? The big question here is when will the Regression testing occur in the Agile lifecycle? What should be within the iteration, and what should be excluded from the iteration? Of

course, the standard testing answer is “it depends” – the key point is to identify what the dependencies are.

New functionality could be developed under an Agile approach, but then integrated using more traditional methods, or even the reverse, traditional methods for the initial development and Agile testing for the Integration and Regression test activities. For example if the project was about installing a Service Oriented Architecture (SOA) infrastructure, which is increasingly common in the Industry, and incorporating legacy systems with new “Commercial Off The Shelf” products (COTS) [12]. The business logic layer may be done in an Agile approach, or the new functionality could be developed using the more traditional methods. Either way, what about the Integration and Regression testing? There are two types of regression testing to consider; regression testing within the Iteration and regression testing when the new functionality is integrated within the system as a whole, or when the new system is integrated within the “system of systems”.

Applying an Agile Testing approach to the regression testing is a risk, the main risk being that the right people are not made available or consulted on what is important to the business for the regression User Stories. Sometimes this could be due to the merger or take over of an existing system, where the knowledge has not been retained. However when there is little or no documentation, surely this is the ideal time to involve the Business and have them understand the risks, the testing approach, and define the Acceptance Criteria; which is a fundamental tenant of the Agile Testing approach.

Another risk with the Agile approach is coverage, can we ensure we achieve sufficient coverage for the Business and the new implementation? This comes back to the Business and working closely with the Development Team and the Test Team to clearly define the User Stories that are associated with regression. The Agile approach allows the discussion and analysis of the risks and implications of the new development, and any further testing that must take place.

Another characteristic of the Banking/Finance domain is the heavily regulated environment, which implies lots of regression testing – this is ideal for an Agile testing approach as the regression tests are an integral part of the development lifecycle. Unit Testing and Automated Test Suites are part of the Agile approach, and when done correctly provide a traceable and re-usable framework for all development and testing. Of course this only applies to the systems that have been developed using an Agile approach. When considering the Integration or Regression testing, an Agile testing approach allows the ongoing development and retention of these frameworks, independently of any specific development lifecycle.

The Agile testing approach advocates a high degree of tool usage which means that the performance and security aspects of the testing can be included in the standard System and Acceptance test suites. These can be moved from the domain of the specialists, once written, and incorporated into the standard suite of Regression and Acceptance tests, meaning that this area is no longer over looked or considered “too hard” . Agile approaches with their inherent risk analysis enable risk mitigation by focusing on the “hard” things earlier in the development cycle, and by doing so,

escalating them in the order of development and implementing them as early as possible.

Additionally when the Agile approach is taken to testing, the Business User is able to define, via their User Stories, the importance and/or risks associated with these aspects of the system. Agile testing approaches mean that the Business User is also there to participate in the testing, or demonstrations, so that they can say when it is “good-enough” instead of assigning arbitrary and sometimes un-testable numbers to performance characteristics. They are gathering a better understanding of their system, earlier in the development cycle.

An Agile testing approach is excellent for other non-functional testing; with continual integration, we can focus on the non-functional or Quality aspects of the system (providing we have the appropriate test environment) and have continual testing and re-factoring (if required) of the architecture and code, potentially without the use or “interference” caused by Browsers or other distracting UIs ^[8]. The use of stubs and harnesses, and other test tools coupled with frequent customer demonstrations means that they are able to focus on the performance without the influences of any UI driving their understanding of the system.

An Agile approach to testing also allows the acceptance criteria to be clearly defined, as soon as they become part of any specific User Story – creating the story forces us to consider how it will be tested. These factors are often overlooked or worse, poorly or incorrectly defined, in more “traditional” approaches. These tests are then able to be incorporated into the new or existing system as soon as possible. This also contributes to a substantial knowledge base about the system being established very quickly.

When considering the question of if Agile Testing approach should be used on Banking/Financial domain applications the usual testing answer “it depends” applies. With the move towards more dynamic applications and greater speed to market there is a need for Agile techniques across the board in this domain. Additionally there is the need for very traditional structured approaches to testing due to the regulatory and auditory requirements of the industry. Thus what then occurs is a more “hybrid” approach to the development and testing. We can use different, less structured, techniques during the development phase, almost a purist Agile approach (although Agile is highly structured), and then once the project has been delivered the testing can stand alone, with moderate support, to continue the regression testing. The testing however, could continue to use the fundamentals of Agile Testing to enhance the understanding of the SUT, and to work with the Business in undertaking their risk analysis. Agile test techniques should be seriously considered for the Banking/Finance domain. As with all development and testing approaches, they should be implemented after a period of consideration and analysis. The most important consideration is to ensure that the system is Tested!

References

1. Colonial First State reduces time-to-market for core applications
<http://www.computerworld.com.au/whitepaper/216155?rtid=1&rid=457&location=featured>
2. Nancy Feig, Pursuing the Promise of SOA, June 24, 2008,
<http://www.banktech.com/printableArticle.jhtml;jsessionid=RBPIDUTZ5LFSKQSNLPCCKH0CJUNN2JVN?articleID=208701020>
3. ISTQB Advanced Level Syllabus 2007, www.anzta.org
4. http://www.eclipse.org/community/casestudies/jp_morgan_final.pdf
5. Linda Newsom-Ray, Software Quality Improvement in Financial Services after the Merger
An MKS Whitepaper
<http://www.computerworld.com.au/index.php/id;671849297;relcomp;1>
6. <http://agilemanifesto.org/>
7. Lisa Crispin and Tip House, Testing Extreme Programming, Addison-Wesley, 2003
8. Paul Gerrard, E-Business Testing: Test Techniques and Tools, Risk-Based E-Business Testing, Part 2, Nov 1, 2000