

Tim Brown

Phone: (027) 2089726
E-mail: tim@codelab.co.nz

IT319 RESEARCH PAPER

Guidelines for Managing Evolutionary Information Systems

18th June 2007

Author: Tim Brown

IT319 Research Paper

Guidelines for Managing Evolutionary Information Systems

Abstract

"A competitive advantage comes only with superior IT." - (Aetna Healthcare Chairman/CEO Richard Huber)

Organisations are faced with difficult challenges when strategic decisions are made to upgrade, migrate, integrate or even re-develop their information systems. While business processes are forever changing so does the need to keep their information systems functioning to the required specifications.

"Better! Cheaper! Faster!"

During the 1990's this was the theme for IT project management.

"To produce better and cheaper products or services and get them to market quicker requires better, cheaper, faster processes."

In today's information systems, the model has matured and we factor in the following themes:

- Do it right the first time
- Do only manageable portions at a time
- Do it in a reusable and adaptable manner

Agile Practices (AP) and Object Oriented Analysis and Design (OOAD) include these themes in managing the development of information systems.

In this paper we provide a solution to help IT managers execute an efficient and agile way of managing the process of upgrading/developing their information systems.

TABLE OF CONTENTS

<u>PROBLEM STATEMENT</u>	3
<u>SOLUTION</u>	4
<u>THE GUIDELINES</u>	6
<u>FINDINGS - IDENTIFYING COMMON PHASES IN SYSTEM DEVELOPMENT LIFE CYCLES</u>	21
<u>FINDINGS - COMMON EVOLUTIONARY INFORMATION SYSTEM CONSIDERATIONS</u>	26
<u>FINDINGS - MATCHING AGILE AND OOAD TO COMMON CONSIDERATIONS</u>	39
<u>APPENDIX A - REFERENCES</u>	41

PROBLEM STATEMENT

How to manage the future development of an organisation's evolutionary information system? Once the organisation's system has been implemented and deployed, what happens next? How do organisations manage the next project cycle? How can they improve in the way they manage the system in the future? These are questions many organisations have to face where their information system is the central point of business. The problem is in-fact how can an organisation improve on what they have already got?

Evolutionary information systems are part of an incremental approach where the system and the requirements evolve overtime. This is normally caused by rapid shifts in business processes, strategic directions change within the organisation or management do not fully understand their business direction.

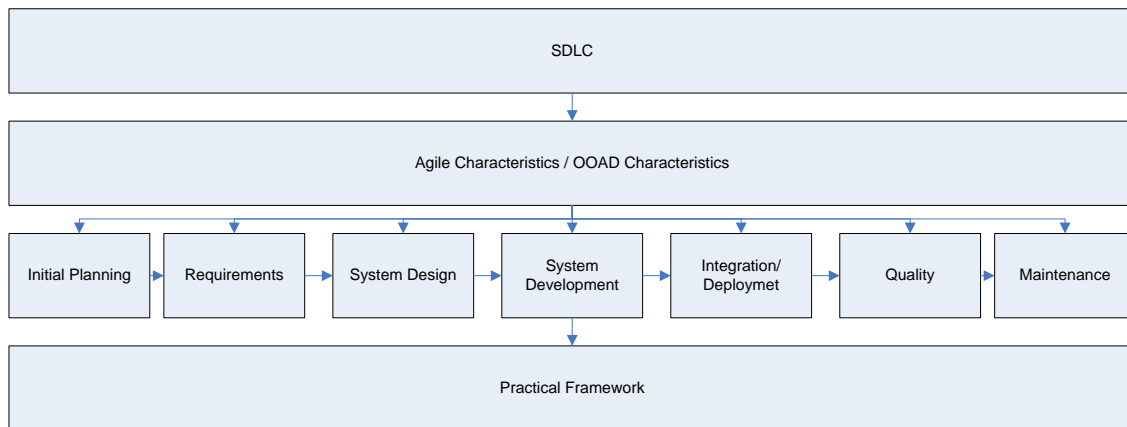
SOLUTION

Answering these types of questions require having a solid set of techniques to manage the development of evolutionary information systems. *“What is an effective approach to managing, modelling, designing, implementing, testing and deploying new releases of evolutionary systems?”*

This solution creates a set of guidelines to identify and resolve key considerations within organisations using Agile Software Development approaches and Object Oriented Analysis and Design (OOAD) techniques.

WHAT IS THE APPROACH?

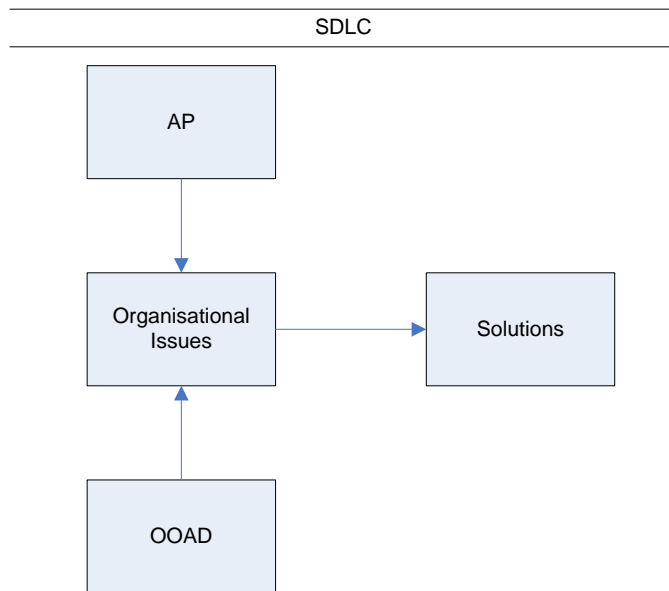
The bases for identifying our guidelines are a set of recommendations from the outcome of our research (*Figure 1 –Guidelines Overview*).



• Figure 1 –Guidelines Overview

The recommendations are determined by four elements (*Figure 2 shows the relationship between the four guideline elements.*):

1. Identifying common phases in any System Development Life Cycle (SDLC)
2. Organisational Considerations
3. Agile Practical (AP) Characteristics
4. Object Orientated Analysis and Design (OOAD) Characteristics



• Figure 2 shows the relationship between the four guideline elements.

1. The goal is to find the commonalities between each phase of key SDLC methodologies and divide the research and findings into the respective phase. I.e. several SDLC methodologies all include an initial planning phase therefore we can categorise parts of our findings into the initial planning phase.
2. Organisational considerations look at specific organisational issues across these common phases.
3. Agile Practices and Principles (AP) is a framework for managing evolutionary change through any SDLC. We relate this framework to the identified phases and organisational considerations.
4. Object Oriented Analysis and Design (OOAD) aims at modelling the problem domain by using such techniques as UML. The analysis is then turned into a logical solution using a variety of techniques including class diagrams, sequence diagrams and Use Cases.

THE GUIDELINES

After conducting our research to identify key common phases of any Software Development methodology (SDLC), key organisational considerations, applying various Agile Practices and Principles (AP) and Object Oriented Characteristics (OOAD), we have defined a set of guidelines that will help IT experts make decisions in managing information system projects.

Consider adopting the following guidelines when managing evolutionary information system projects:

- *Use a Agile Software Development Approach*
- *Model in an Agile way*
- *Understand the BIG picture*
- *Try Model Driven Development*
- *Unify with UML*
- *Expect and allow for change*
- *Plan the interface with UI Prototyping*
- *Make the test cases before you code*
- *Plan the deployment strategy early*
- *Make robust software using Test Driven Development and Full Life Cycle Object Oriented Testing (FLOOT)*
- *Start planning the next project cycle*
- *Keen to try another SDLC approach? Try the Agile Unified Process!*

USE A AGILE SOFTWARE DEVELOPMENT APPROACH

The first guideline is to understand the principles and practices between Agile Software Development. Agile Software Development is about developing software through evolutionary change. The following shows several key principles¹ in Agile Software Development.

"Achieving customer satisfaction through iterative delivered software." – Providing functional software to the client earlier gives opportunity for rapid feedback and the ability to review requirements earlier in the project. By releasing software in small increments eliminates issues such as scope creep.

"Encourage flexibility to requirement changes during a project." Managing requirements is a process through out the life cycle rather than being defined at the start of a project. Using such methodologies as Agile Model Driven Development (AMDD) gives the team the chance to review requirements and prioritise them during development. If projects tend to be over scheduled, the highest prioritised requirements have been completed still giving the client a functional system.

"Deliver working software frequently." Short-increments of software releases promotes flexibility in delivering software.

¹ Agile Manifesto - <http://agilemanifesto.org/principles.html>

"Collaboration between business and technical experts." This means encouraging a supportive and collaborative environment between technical experts and business experts. Make sure that all stakeholders are involved at the appropriate time.

"Trust the team to get the job done." Evolutionary projects are based on motivated team members. Having trust within the motivated team enhances agility.

"Encourage Face-to-face communication methods within the development team." The Agile Manifesto encourages more face-to-face communication between the project team and the use of several artifacts including white boards and paper prototyping etc

"Working software is the primary measure of progress." The ultimate test case is making sure software is meeting the client's expectations. Delivering software through small incremental approaches provides a benchmark for measuring quality software.

"Workloads are evenly distributed between users, developers and stakeholders in a project." All stakeholders, users and the project team should distribute the workload so everyone plays a significant part in completing the project.

"Good design and technical excellence."

Having structured software design practices and the ability to source technical expertise in software development through networks enhances agility across multiple projects.

"Use simplicity where ever possible." Agile Software Development is about developing software using simplistic approaches. Excessive documentation, using the wrong artifacts and not managing requirements are forms of complicated scenarios in software development.

"Self-organising teams" The best software designs, architectures and requirements come from self-organising teams. Giving that flexibility for teams to manage their own agendas provides challenges, motivation and responsibility.

"Project teams spend time reviewing their progress to recommend changes and improvements." Having regular reviews of progress including communication with key stakeholders gives the team the opportunity to review and recommend changes and improvements for future releases and developments.

MODEL IN A AGILE WAY

The second guideline is to understand the values and advantages of Agile Modelling (AM) and how these set of values can enhance the software development process in an organisation.

AM follows a set of values² including communication, simplicity, feedback, courage and humility. Effective agile approaches include effective communication with stakeholders, taking simplistic approaches, obtaining rapid and regular feedback, having confidence in making decisions and encourage a supportive environment.

² <http://www.agilemodeling.com/values.htm>

AM is based on a number of key principles and practices.

Key principles³ include embracing change and assume simplicity so project teams can manage the change of requirements during a project. The incremental change of a system gives the opportunity for rapid feedback from key stakeholder members.

Model with a purpose, understand what to model and using multiple model methods help understand a problem.

Travelling light enables models to be discarded that are not suited for future documentation because agile modelling is based on the statement "*Content is more important than representation*".

Having open and honest communication with stakeholders and clients eliminates any "sloppy" work during the project.

To be an effective agile modeller there are several key practices. These include modelling in parallel, applying the right artifacts and iterating to another artifact so there is a steady pace to keep the project moving.

Modelling in small increments reduces any implications of development later in the project while proving the model with code also eliminates discrepancies between theoretical models and working examples.

Using simple modelling scenarios and tools helps stakeholders understand the system better.

Active stakeholder participation provides rapid feedback during iterative software releases for quality and testing purposes.

UNDERSTAND THE BIG PICTURE

Initial Planning includes identifying the project scope, stakeholder acceptance, project funding and identifying resources to the project. Key considerations include managing timeframes, budgets/costs, product knowledge, managing risks and stakeholder involvement.

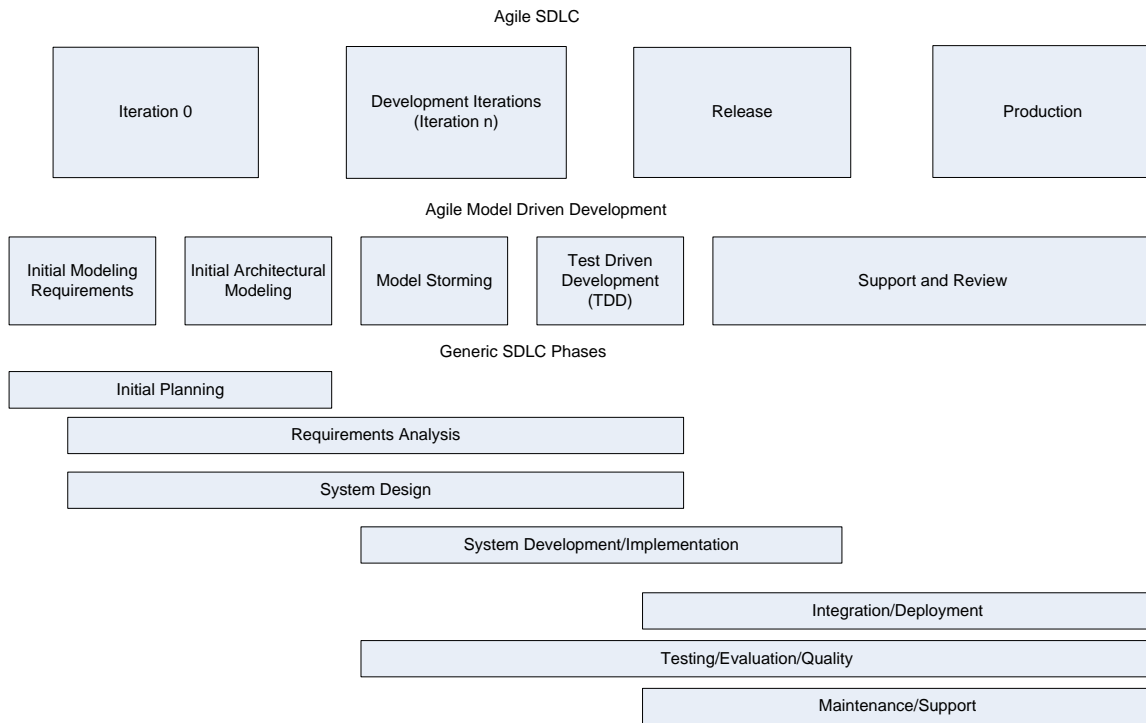
Follow these practices and principles below to help understand and manage the big picture:

- One of the agile principles is to maximize the stakeholders Return on Investment (ROI). Stakeholders invest their time and money in order to get projects completed. This should be managed appropriately to get the best ROI.
- In the initial planning of a project, make sure that all stakeholders are involved during the project provides several advantages including rapid feedback, better measuring of delivered software and reduce requirement issues like scope creep.
- "*Accurately plan in detail only for nearby tasks*" – Use such models as Just-in-time (JIT) planning.
- "*The people doing the work must be actively involved in scheduling*" – Make sure that all of the project team who are actively involved in the project have input into the scheduling so that everyone knows the timeframes and milestones.
- "*People should choose their work, they shouldn't be assigned it*" – A better approach is to let the project team do the planning, not just the project manager.

TRY MODEL DRIVEN DEVELOPMENT

³ <http://www.agilemodeling.com/principles.htm>

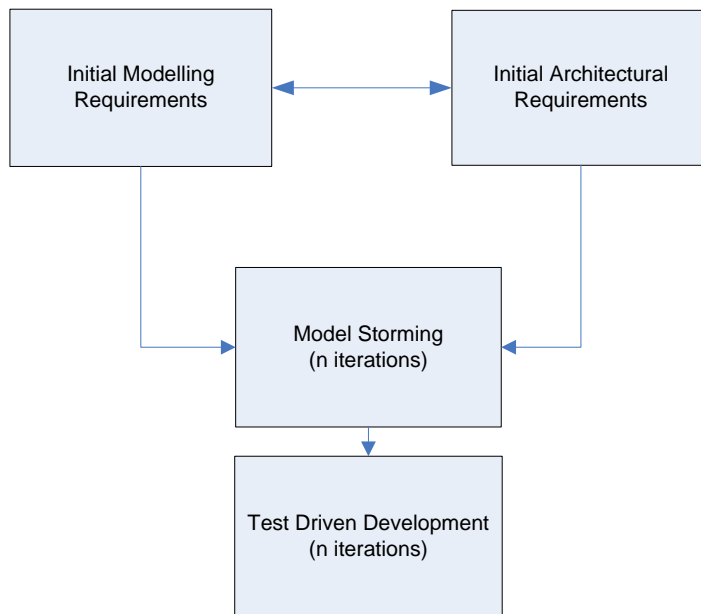
For the third guideline follow the Agile Model Driven Development (AMDD) approach⁴. *Figure 3 - Relationship between Agile Model Driven Development and SDLC Phases* shows how the AMDD relates to the common phases outlined in this paper.



• Figure 3 - Relationship between Agile Model Driven Development and SDLC Phases

AMDD is based on MDD (Model Driven Development). This approach requires extensive modelling before source code is written.

⁴ <http://www.agilemodeling.com/essays/amdd.htm>



• Figure 4 - Overview of Agile Model Driven Development

Illustrated on Figure 4 (Figure 4 - Overview of Agile Model Driven Development), iteration 0 involves initial modelling and initial architectural requirements. Model Storming and Test Driven Development (TDD) is grouped together into a number of iterations.

INITIAL MODELING REQUIREMENTS/ INITIAL ARCITECTURE MODELING

With evolutionary information systems, identify the high-level requirements for upgrades/re-development to the existing system before detailing the exact requirements during the initial part of the project. This is complemented with essential use cases on how the system interacts with external actors within the problem domain.

There are three types of models that should be developed:

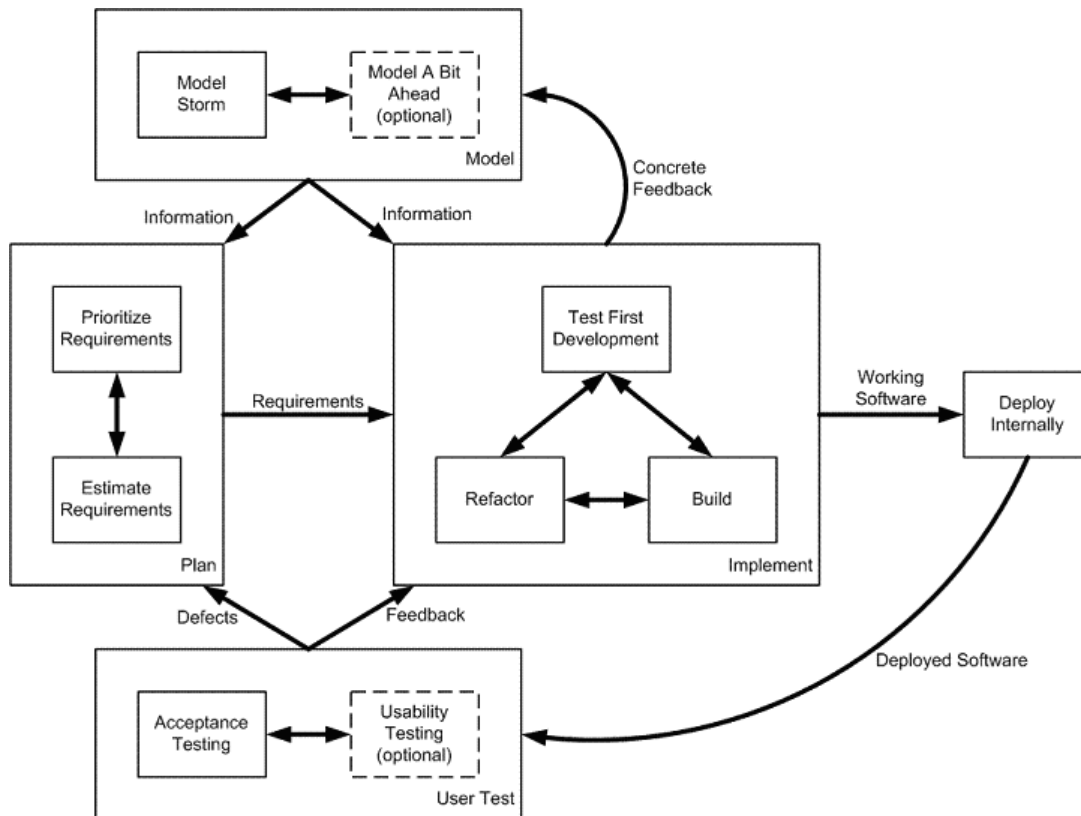
- Usage Model – Initial Use Cases, Feature Driven Development (FDD)⁵ etc
- Initial Domain Model – Identifying entities and business types for the project. This includes UML sequence diagrams, data modelling etc
- User Interface Model – Prototyping of how the system may look like in forms of several initial rough diagrams.

Reviewing existing architectures, identifying the architectural vision and recommending solutions for the use of new and/or stable technologies is part of gathering the requirements of modelling the system architecture.

DEVELOPMENT ITERATIONS - MODEL STORMING/TEST DRIVEN DEVELOPMENT

⁵ <http://www.agilemodeling.com/essays/fdd.htm>

Development iterations (See *Figure 5 - Agile SDLC Development Iterations*) involves prioritising the requirements stack, modelling how the requirements will be implemented, implementing the requirements, testing and releasing software internally to stakeholders.



• Figure 5 - Agile SDLC Development Iterations

Key advantages:

- Being collaborative between stakeholders and developers. Reducing the risk of miss-communication.
- Implementing requirements in a specific order. The client/stakeholders choose which requirements get implemented first, giving control to the client and stakeholders.
- Ensuring quality. Using iterative development requires code to be re-factored and database schemas to be changed to make sure the best possible design is used.
- Testing! Each functional requirement is tested once been modified by the developer and again by stakeholders ensuring the quality of the software.

UNIFY WITH UML

The Unified Model Language (UML) is a standardised modelling language used in software development. It's a way of communicating how a system is designed and functions with stakeholders and with the project team. UML has several advantages⁶ including:

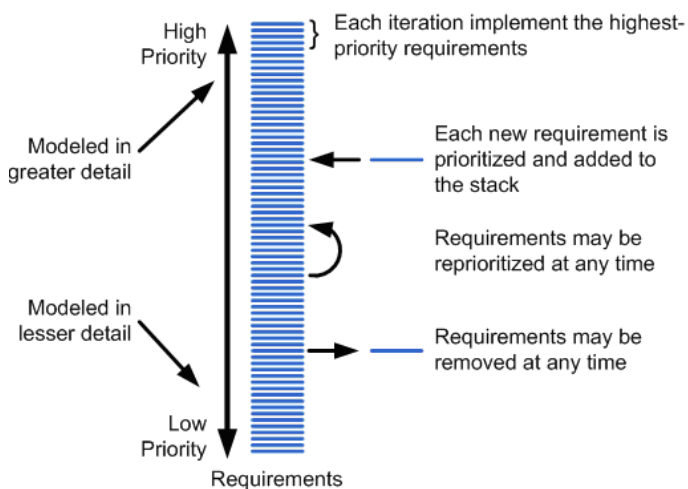
- "Being a formal language" – The notation of the language is highly defined
- "Its Concise" – Made up of straight forward notation
- "Comprehensive" – Illustrates all important aspects of the system
- "Scalable" – Can be used on small to enterprise scale systems

Agile Modelling (AM) approaches and UML complement each other and solve several key organisational considerations including documentation, communication, estimating, scheduling and understanding how the system works.

EXPECT AND ALLOW FOR CHANGE

Requirement Analysis (RA) includes identifying what the requirements are, identifying existing requirements vs. new requirements for business change. Key considerations include identifying requirements and managing how requirements are prioritised and implemented.

When the organisation requirements change indefinitely, use the Agile Change Management (ACM) process. The ACM process is a way of managing requirement changes during a number of iterations. As requirements change so often in software development, this process help keeps the project on-track but helps maximize the Return on Investment (ROI).



• Figure 6 - Agile Change Management Process

The prioritise stack (*Figure 6 - Agile Change Management Process*) shows that on the commencement of each iteration, high priority requirements are developed first. New requirements can be added and existing requirements can be removed or have their priority changed. Implement this process at the start of each iteration. It's recommended to freeze the prioritize stack so that developers have a stable period to implement requirements.

Consider these requirement practices:

⁶ Advantages cited from Learning UML 2.0 By Kim Hamilton, Russell Miles

- *Stakeholders actively participate* – Stakeholders are the key to defining and determining what the requirements are. They should be actively involvement and be approachable during the project.
- *Adopt inclusive models* – It is easier to include stakeholders in the requirement modelling and documentation areas of the project. We recommend using simple tools and methods to illustrate requirements to stakeholders. This may include paper prototyping, CRC cards and other artifacts.
- *Take a Breadth-First Approach* – It is proven that spending too much time developing requirement documentation and analysing the system in detail initially is not effective. Try just doing enough modelling to understand the big picture and then do detailed analysis of the system when appropriate (*In a just-in-time fashion*).
- *Smaller is Better* – Keep requirements short and small, try to avoid large requirements that these introduce risks such as unclear or misinterpret requirements.

PLAN THE INTERFACE WITH UI PROTOTYPING

System analysis/design includes designing software and hardware architectures. Defining components, design methods, class diagrams, data modelling and choosing the preferred Integrated Development Environment (IDE) is part of this phase. Key considerations include adopting the right methodology (agile, UML, data flows etc) for analysing system requirements and architecture.

As part of our guidelines, follow these practices and principles to help define a standardised approach to designing software.

User Interface (UI) Prototyping helps stakeholders and the project team understand how the system interacts with users. UI prototyping is an iterative technique used to mock-up the interface of a system. There are two types of prototyping, tradition UI prototyping and essential UI prototyping. Use essential UI prototyping because this prototyping approach focuses on how the user interacts with the interface rather than purely on system features. Use simplistic artifacts including white-boards and paper prototyping, we also recommend doing Use Case modelling in tandem. These types of techniques complement each other, Use Case modelling describes what the system should do with external actors, and UI prototyping illustrates what the user should do from the interface. Consider the following practices:

- *“Work with the real users”* – Involve users who will be using the system when the project has been completed
- *“Get stakeholders to work with the prototype”* –Involve key stakeholders in the prototyping phase so they know what to expect when software goes into production.
- *“Understand the underlying business”* – Understand the business processes and requirements before creating prototypes. This is critical for existing applications with understanding the existing system logic and how the existing prototype comparing with the new prototypes.
- *“Only prototype features that can actually be built”* – If a functional requirement cannot be delivered, there is no point prototyping it.
- *“Get an interface expert to help you design it”* – Another alternative is to get expert advice from an interface expert who knows how to design “easy-to-use” interfaces.

MAKE THE TEST CASES BEFORE CODING USING TEST DRIVEN DEVELOPMENT

System development/implementation is defined as interpreting the requirements into a working product using software engineering techniques. The key considerations include adopting standard development practices and how to incorporate the right technologies into a project.

Follow these practices and principles to formulate a standard in implementing the system requirements:

- *Prove the model with code* – When implementing requirements, make sure that the conceptual implementation model can be proven with code (*the final result*).
- If a developer does not understand a requirement, they should model with stakeholders on a just-in-time (JIT) manner to get the information required.
- Developers do not need to wait for a specification design document to be created. They can create models which are just barely good enough for the situation.
- Implement the Test-Driven Development (TDD) approach. Develop enough code to pass suitable tests, and then if tests fail, re-factor the code to pass any failed tests.

PLAN THE DEPLOYMENT STRATEGY EARLY

Integration and Deployment involves releasing stable versions of the system onto platform environments for stakeholders to use. Key considerations include having the right environments to develop, test and deploy software and the suitable practices for releasing software.

Follow these practices and principles to help develop a strategy to deploying software⁷:

- *Understanding the deployment audience* – who is involved in each part of the deployment process, e.g. support staff, operations and end-users.
- *Making a deployment strategy early* – Have in-place a strategy plan for releasing new software or systems into a production environment, but also have plans in-place for releasing to pre-production and development environments.
- *Testing installations scripts* – Test the installation scripts, e.g. if the organisational is using a build automation application, make sure they have tested the installation onto a pre-production environment first (*Figure 7 - Shows System Deployment Architecture*).

⁷ Cited from <http://www.ddj.com/dept/architect/198800565?cid=Ambysoft>

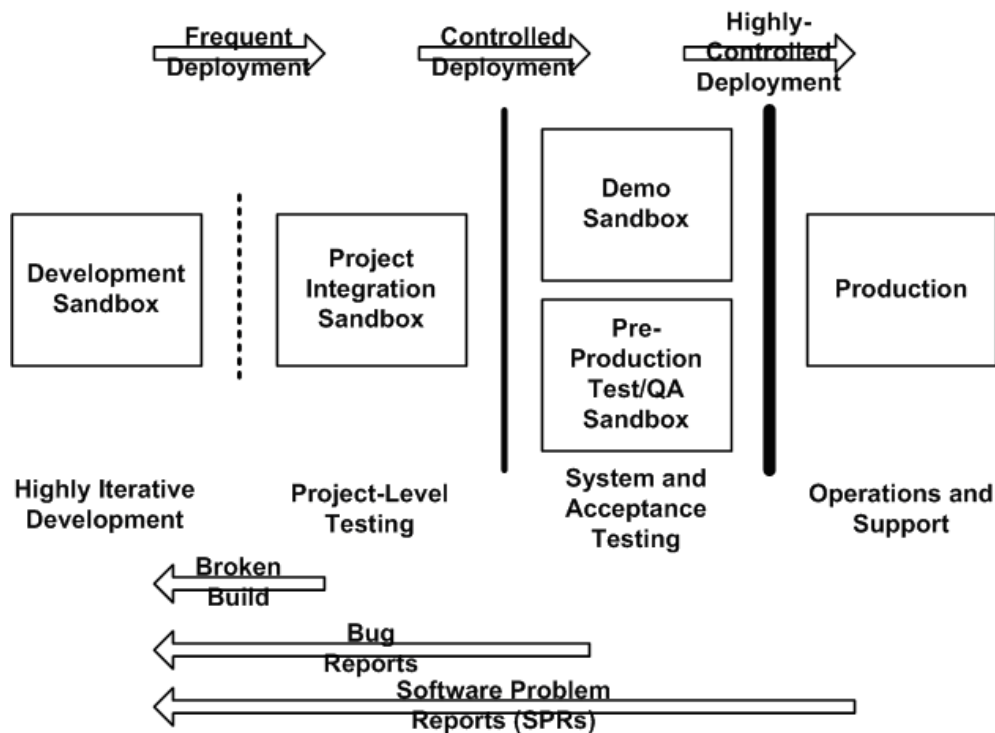


Figure 7 - Shows System Deployment Architecture

- *Regular software releases* – at the end of a single iteration, a release should be made onto the pre-production environment. This then can go through Quality Assurance (QA) and other testing procedures before being released onto the production environment.
- *Agile Project Planning approach* – keep in mind the high-level plans, but only work on specific details on a Just-in-time (JIT) basis.
- *Develop de-installation scripts* – It is critical being able to rollback or back out of an installation if anything unexpected happens during builds.
- *Training* – Training is a key role in new releases of software. Users should be aware and be trained for new features in existing software to avoid confusion and frustration in the long-term.

MAKE ROBUST SOFTWARE USING TEST DRIVEN DEVELOPMENT AND FULL LIFE CYCLE OBJECT ORIENTED TESTING (FLOOT)

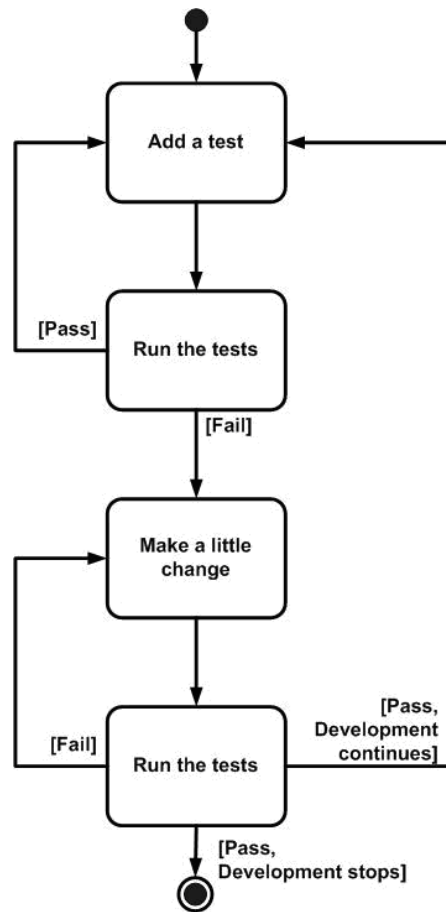
Testing, producing quality software and evaluating software have elements in all software methodologies. This includes “when does testing occur during the project? How do we measure the quality of the system? Users perception vs. perception of quality, how do we know?”⁸ Key considerations include using the appropriate type of testing methods (unit testing, integration testing, test driven development etc), and standardising testing types across the organisation.

Follow these two testing methodologies Test Driven Development and Full Life Cycle Object Oriented Testing (FLOOT) methodology:

⁸ Agile Testing Strategies - <http://www.ddj.com/dept/debug/196603549?cid=Ambysoft>

Test Driven Development

Test Driven Development (TDD) is an approach to developing quality software. The approach combines test-first development (*Figure 8 - Steps for Test-First Design*) and re-factoring.



• Figure 8 - Steps for Test-First Design

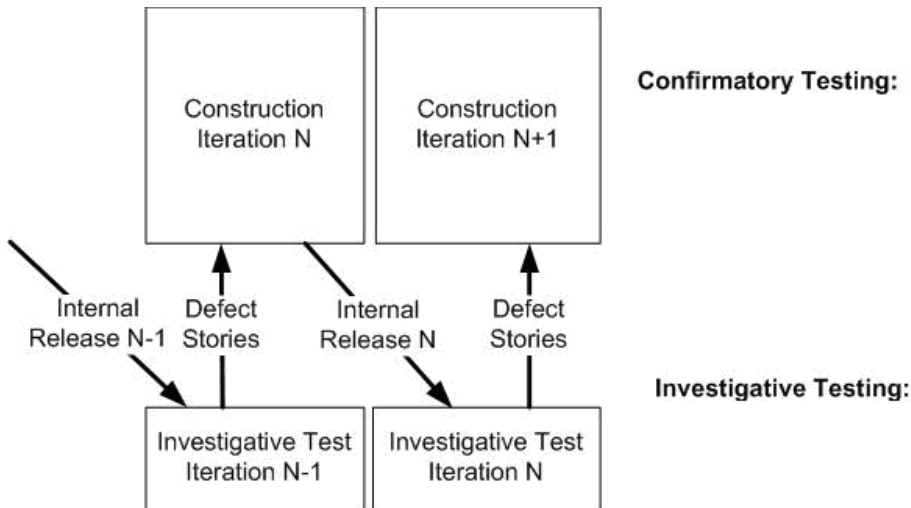
The idea is to develop an initial test. Then develop enough code to fail the new test.

Change the code until the test has been passed. Create a new test and go through the same procedures.

TDD should be used in conjunction with the Agile Model Driven Development (AMDD) approach. AMDD helps model requirements in an iterative way to stakeholders while TDD helps create clean and quality code, hence the outcome is overall quality software.

There are two types of testing in TDD, one being Confirmatory Testing (*developer/unit testing*) and Investigative Testing (*system testing, user testing and other testing*).

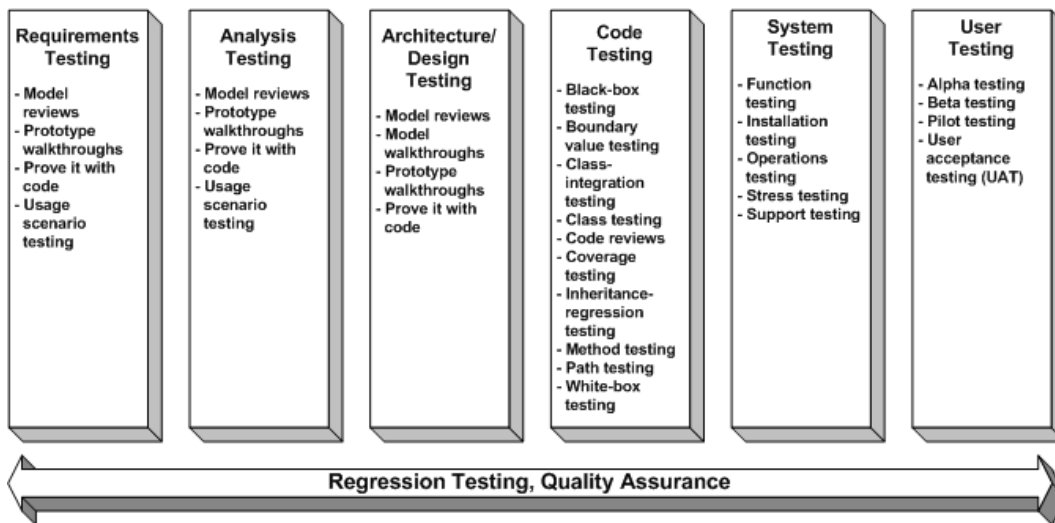
Investigative Testing should be conducted on a new internal release. After identifying any defects, these should be fixed and tested using Confirmatory Testing. This process happens during a single iteration (*Figure 9 - Testing during development iterations*).



• Figure 9 - Testing during development iterations

Full-Life Cycle Object Oriented Testing (FLOOT) methodology is a collection of testing techniques to validation object oriented software. FLOOT although is not limited to object oriented software, it can be applied to other types of software methodologies. The whole idea is to use a wide range of testing techniques against each phase in any system development life cycle (SDLC).

It's recommended to identify various testing techniques against a chosen SDLC (*Figure 10 - FLOOT Life Cycle*).



• Figure 10 - FLOOT Life Cycle

START PLANNING THE NEXT PROJECT CYCLE

Maintenance/Support includes changes requested by the client that should be made to the system. This normally involves a MAC(s), bug fixing and support agreements that is formally signed and delegated to the developers to implement. Key considerations include how to manage support/maintenance to the system after system is in production.

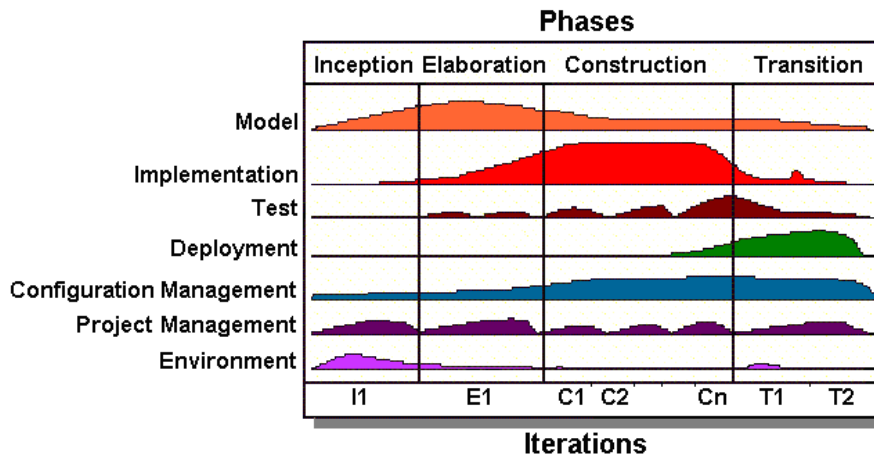
The guideline includes following these practices:

- *Release Automation Software* – This helps automate the release procedures for new builds and future version of the software.
- *Support Agreements* – Document and determine support agreements with the client to make sure defects are fixed and resolved.
- *Next Project Initiation* – Start to plan the next project cycle, this might include new requirements, changes and bug fixes.

ADOPTING THE AGILE UNIFIED PROCESS

Our last guideline recommends that if the organisation does not currently have any standardised system development life cycle approaches. They should consider the Agile Unified Process (AUP) methodology (*Figure 11 - Agile Unified Process Overview*).

This methodology is a modification of the IBM Rational Unified Process. The Agile Unified Process focuses on evolutionary projects.



• Figure 11 - Agile Unified Process Overview

The AUP incorporates all of our guidelines including using Agile Values, Principles and Practices, Incremental Approaches and Object Oriented Analysis and Design techniques.

Here is a summary table showing the functions of each workflow (*left axis*) for their respective phase (*top axis*).

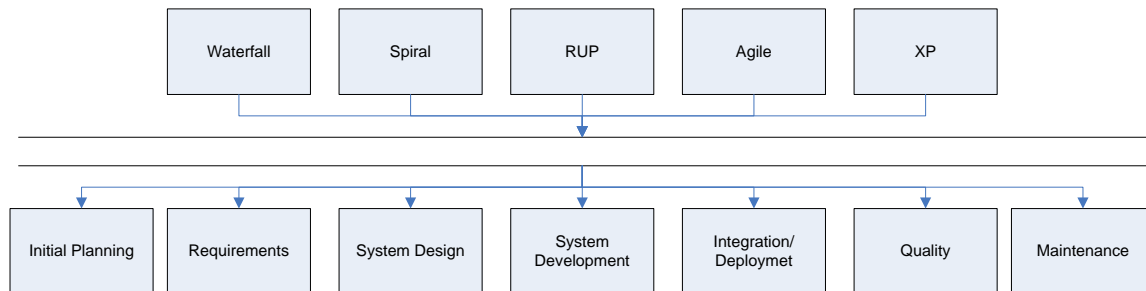
	Inception	Elaboration	Construction	Transition
Model	High-Level Requirements Modelling Business Rules Use Case Development Initial Prioritization	Identify Technical Risks Architectural Modelling User Interface Prototyping/Modelling	Analysis Model Storming Design Model Storming	Finalise system overview documentation Finalise other documentation

	Stack			
Implementation	Technical Prototyping User Interface Prototyping	Prove the architecture	Test first Build continuously Evolve the domain logic Evolve the user interface Evolve the data schema Develop interfaces to legacy assets Write data conversion scripts	Fix Defects
Test	Initial Testing Planning Review Initial Models	Validate the architecture Evolve your test model	Test Software Evolve your test model	Validate the system Validate the documentation Finalise your test model
Deployment	Identify the potential release window High-level deployment planning	Update your deployment plan	Develop (de)installation scripts Develop release notes Develop initial documentation Update plans Deploy into pre-production environments	Finalised deployment plan Finalised Documentation Training
Configuration Management	Setup Configuration Environments	Put all work under CM control	Put all work under CM control	Put all work under CM control
Project Management	Start building the team	Building the team	Manage the team	Manage the team Close out the

	Build relationships with stakeholders Project Feasibility High-Level schedule Iteration Plan Manage Risks Stakeholder support and funding Close Phase	Protecting the team Obtain resources Manage Risk Update Project Plan Close Phase	Manage risk Update Project Plan Close Phase	phase Initiate next project cycle
Environment	Setup the working environment Identify the project category	Evolve the working environment Tailor the process materials	Support the team Setup training environments	Setup operations/support environments Recover software licences

FINDINGS - IDENTIFYING COMMON PHASES IN SYSTEM DEVELOPMENT LIFE CYCLES

To determine the basis of what the guideline framework is, we identified the similarities between several well known system development methodologies. (Figure 12 - Similarities between System Development Life Cycle (SDLC) methodologies.)



• Figure 12 - Similarities between System Development Life Cycle (SDLC) methodologies.

We conducted research into several methodologies including the Rational Unified Process, Traditional Waterfall, Spiral, Agile Unified Process and Agile Driven Development⁹.

We determined the following common phases of any System Development Life Cycle (SDLC) methodology:

Common Phases of System Development Life Cycle Methodologies	
Common Phases	Influences
Initial Planning	<p>Initial Planning includes identifying the project scope, stakeholder acceptance, project funding and identifying resources to the project.</p> <p>Initial planning is used within the methodologies below:</p> <p>Waterfall – The initial planning is included in the Requirements Specification phase. This phase includes eliciting, analysing and recording requirements.</p> <p>Spiral – Initial planning is the first phase using the spiral method. This phase includes project scope and interviewing key stakeholders.</p> <p>Agile Unified Process – Inception phase is initial planning including estimations, time, and stakeholder acceptance.</p> <p>Rational Unified Process - Inception phase is initial planning including estimations, time, and stakeholder acceptance.</p> <p>Agile Driven Development – Initial Modelling includes initial planning, high level planning, risk analysis and identifying the</p>

⁹ System Development Process - http://en.wikipedia.org/wiki/Software_development_process

	requirements stack.
Requirements Analysis	<p>Requirement Analysis includes identifying what the requirements are, identifying existing requirements vs. new requirements for business change.</p> <p>Waterfall – The requirements specification phase includes analysing the client’s needs and requirements that are transformed into a solid functional requirement document before design and implementation phases are started.</p> <p>Spiral – The spiral methodology involves defining a large portion of the requirements before design and prototyping phases are started.</p> <p>Agile Unified Process – The elaboration phase includes design the architecture of the system. During the elaboration phase, the model workflow is the most important part. The model workflow defines the organisation’s problem domain. A series of small releases are implemented and requirements are always reviewed and prioritised using Agile Change Management processes.</p> <p>Rational Unified Process – The requirements discipline is to agree on the scope of the project by working closely with project stakeholders understanding their needs, exploring business rules, user interface designs and identifying requirements through out the project. This discipline happens through out the project but mostly in the inception and elaboration phase of the project.</p> <p>Agile Driven Development – Initial modelling requirements involves defining the requirements stack. This stack can be changed during the project using prioritisation methods. Base on iterative development cycles, requirements are refined after each release.</p>
System Design	<p>System design includes designing software and hardware architectures. Defining components, design methods, class diagrams, and data modelling and choosing the preferred Integrated Development Environment (IDE) is part of this phase.</p> <p>Waterfall - This step consists of "defining the hardware and software architecture, components, modules, interfaces, and data...to satisfy specified requirements"¹⁰.</p> <p>Spiral – Normally a preliminary design is creating using prototyping methods then follows a series of refined designs of the system. The design is implemented and the design process is refined.</p> <p>Agile Unified Process – The modelling workflow includes the designing of the system and most of the system design is during the elaboration phase of the project.</p> <p>Rational Unified Process – The Analysis and Design workflow</p>

¹⁰ http://en.wikipedia.org/wiki/Waterfall_model

	<p>covers most of the system designing during the project under the elaboration phase.</p> <p>Agile Driven Development – System design is included in the initial modelling and during iterations of design and implementation using Test Driven Development (TDD).</p>
System Development/Implementation	<p>System development/implementation is defined as interpreting the requirements into a working product using software engineering techniques.</p> <p>Waterfall – The implementation phase involves constructing the system to produce a solid working system.</p> <p>Spiral – The development phase as managed in small iterations which includes the design and implementation phases.</p> <p>Agile Unified Process – The implementation workflow manages the development of the system. This workflow is managed under the construction phase where a number of iterations are deployed to produce the final stable version of the system.</p> <p>Rational Unified Process – The implementation workflow is managed mostly under the construction phase but also in the later stages of the elaboration phase.</p> <p>Agile Driven Development – Agile Driven Development is followed by developing in small iterations. The iterations have initial modelling storming, Just-in-time (JIT) requirement decisions and Test Driven Development (TDD).</p>
Integration/Deployment	<p>Integration and Deployment involves releasing stable versions of the system onto platform environments for stakeholders to use.</p> <p>Waterfall – The Installation/Verification phase shows the system has now been tested and certified to be released onto a working environment.</p> <p>Spiral – Small iterations are developed to stakeholders who provide feedback that triggers refinement of requirements. A solid robust system is developed based on small iterations of development and releases.</p> <p>Agile Unified Process – The deployment workflow is mostly carried out during the transition phase of the project. Most cases the final version of the system is released to users.</p> <p>Rational Unified Process – The deployment workflow is jointly carried out during the end of the construction phase and during the transition phase.</p> <p>Agile Driven Development – Small iterations of the system are developed which go through processes of User Acceptance testing and usability testing. Normally the last release of the system is deployed onto the production environment.</p>
Testing, Evaluation and	Testing, producing quality software and evaluating software have

Quality	<p>elements in all software methodologies. This includes asking questions like when does testing occur during the project? How do we measure the quality of the system? And Users perception vs. perception of quality.</p> <p>Waterfall – The testing phase usually happens directly after the implementation phase where test cases are generated and compared against the code developed in the implementation phase. This phase is where bugs are identified against the functional requirements and fixed before a final release.</p> <p>Spiral – Testing happens after each development phase. After a build has been release, regression testing and other testing procedures are used. If defects are identified they are resolved and put into the next build.</p> <p>Agile Unified Process – The test workflow covers testing the quality of the system, identifying defects and fixing defects. Once defects are fixes, small builds are released before the final build to the client to ensure quality. The test workflow is normally performed during the construction and transition phases.</p> <p>Rational Unified Process – The test workflow covers testing the quality of the system, identifying defects and fixing defects. Once defects are fixes, small builds are released before the final build to the client to ensure quality. The test workflow is normally performed during the construction and transition phases.</p> <p>Agile Driven Development – Agile Driven Development uses Test Driven Development to produce quality software. Test Driven Development starts by developing test codes which passes testing criteria. Once each criterion is met, development progresses until tests are complete.</p>
Maintenance/Support	<p>Maintenance/Support includes changes requested by the client that should be made to the system. This normally involves a MAC(s), bug fixing and support agreements that is formally signed and delegated to the developers to implement.</p> <p>Waterfall – The maintenance phase includes making changes to the system that includes change requests, defects and normal system administration tasks.</p> <p>Spiral – Normally routine maintenance is carried out after the project has finished. Routine maintenance is carried out in a similar spiral motion of gathering requirements, design, implementation, testing, builds and deployment.</p> <p>Agile Unified Process – The Deployment and Configuration management phases cover change requests, managing change, maintenance builds etc.</p> <p>Rational Unified Process – The Deployment and Configuration management phases cover change requests, managing change,</p>

	<p>maintenance builds etc.</p> <p>Agile Driven Development – Agile Driven Development (ADD) is driven on a number of small iterative releases, once a system goes into a production phase, change requests are handled by developing a number of small builds and patches to be released onto the production environment.</p>
--	--

FINDINGS - COMMON EVOLUTIONARY INFORMATION SYSTEM CONSIDERATIONS

INDIVIDUAL PROJECT CONSIDERATIONS

By identifying the common phases for a collection of system development methodologies, we can then break up the evolutionary information system considerations into their respective phase.

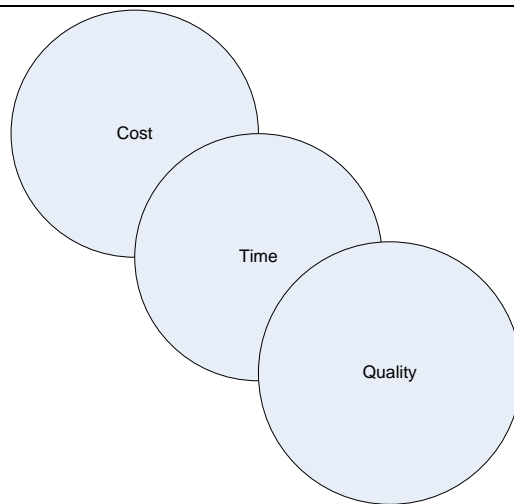
We looked at several projects that have been successfully developed, implemented and deployed in the past but are reviewed periodically for either regular upgrades or re-development. We looked at specific areas of their systems where considerations can be made. We can match the similarities between each project to formulate a common census of considerations.

NOTE: Before continuing to read the following section, we have contacted the respective companies and obtained approval to use their projects in our study. For confidentiality purposes, we have suppressed the company names.

COMPANY A

The following table shows the evolving considerations from the Company A Administration System. The Administration System provides an interface that allows staff to administrate the core platform for their clients. The system controls all the data that is being accepted and transferred to their clients.

Common phases	Considerations
Initial Planning	<p data-bbox="548 1079 808 1104"><u>Budget Considerations</u></p> <p data-bbox="548 1150 1382 1283">Company A's administration system is the framework to maintaining the whole core business including entities as billing, support, operations and administration. The company has taken the following into consideration when upgrading their system (<i>Figure 13 - Company A's Budget Model</i>):</p> <ul data-bbox="597 1329 1365 1535" style="list-style-type: none"> <li data-bbox="597 1329 1260 1392">• <i>Time to market</i> – The time to upgrade the system vs. developing new products to generate income <li data-bbox="597 1398 1365 1461">• <i>Quality</i> – Generating new income vs. making sure core system is robust <li data-bbox="597 1467 1289 1535">• <i>Costs</i> – Costs of enhancing existing systems vs. costs in developing new products



• Figure 13 – Company A's Budget Model

Skill Set Considerations

Company A have considered conducting research in how to implement the latest technologies into their existing architectures. Hence they have considered:

- Skill set of existing staff
- Wanting to be a technological leader in their industry
- Specialised Certification for staff and Quality Certifications

Licence / Escrow Considerations

Company A have considered licensing parts of their system to external members, i.e. charging for accessing real-time information.

Hardware Infrastructure Considerations

Considering hardware/networking infrastructure changes to improve the quality of the information and system factors into the initial project plan of costs and time.

Documentation Considerations

One of the major risks that effects the development of the system is the amount of documentation available about how to system works. All knowledge is available within the minds of developers. They have considered what happens when developers move on to new jobs? Spending time developing just enough documentation to cover all areas of the system?

Requirements Analysis

Identifying the client considerations

	<p>Company A started as a Research and Development (R&D) company, it was difficult to understand who the client was and what their actual product was. They need to consider who is the first point of contact? Operations? Administration? Support or do we have a dedicated Analyst to determine what the requirements are? Who is responsible for the overall requirements of the system from both a business and technical perspective?</p>
System Analysis and Design	<p><u>Standardised Design Patterns Considerations</u></p> <p>Company A considered:</p> <ul style="list-style-type: none"> • Using Modelling tools and techniques, e.g. encouraging modelling techniques between developers to understand the requirements • Structure approach to designing software, e.g. have a standardised way of designing software across projects • Documentation (Functional and Service Specifications) • Prototyping (User Interface Designs)
System Development	<p><u>Testing Considerations</u></p> <p>Company A considered implementing testing patterns and strategies across projects and their system to record and measure the quality of their systems. This includes unit testing, test cases and test plans.</p> <p><u>Technologies</u></p> <p>Company A have a well defined architecture. They are considering what type of technologies could enhance their existing architecture.</p>
Integration/Deployment	<p><u>Staging Environments</u></p> <p>Company A have considered having different types of environments:</p> <ul style="list-style-type: none"> • Development • Testing • Staging • Production <p>They are considering looking at their procedures and deployment tools to help automate the processes of releasing new upgrades/versions of their system.</p>
Quality, Evaluation and Testing	<p><u>Testing techniques</u></p> <p>Company A is considering using different types of testing methods. Their company relies on accurate data, methods like regression, performance and unit testing.</p>

	<p><u>Support</u></p> <p>Company A are considering:</p> <ul style="list-style-type: none"> • Having more documentation and well defined support processes • Implementing support fault management to monitor critical parts of the system • Define key performance indicators to provide quality reports <p><u>Performance</u></p> <p>Because the system relies of data integrity, data optimization and performance, there is a major risk of having incorrect data or slow retrieval of critical client data. They are considering looking at expert advice in database optimisation and maintenance.</p>
Maintenance/Support	<p><u>Support System</u></p> <p>Company A are considering how they will manage the support agreements with their clients but also how they manage their internal work between staff.</p> <p><u>Procedures</u></p> <p>Company A are also considering the structure to manage bug/defect fixing for their clients systems and for their own system. They are considering who is responsible for handling the support operations (i.e. helpdesk etc)</p>

COMPANY B

Company B provides satellite and wireless solutions to NZ rural community. The Company B Information System handles the day-to-day operations, billing, support and sales for the company (*A small scale ERP system*).

Common phases	Considerations
Initial Planning	<p><u>Stakeholders</u></p> <p>Company B do not have the right resources to make decisions on their information system requirements. They know about how the business runs but do not know how the information system should run to cater for their business processes. They are considering:</p> <ul style="list-style-type: none"> • If there is a need to define a analyst role • Outsource • Consultants/Analysts for expert advice

	<p><u>Budgeting</u></p> <p>Currently development is done on a week by week basis, a list is created on what needs to be developed next, time frame and hours are identified and confirmation is made by Company B.</p> <p>Company B are considering creating projects with defined scope, time and budgets.</p> <p><u>Project Identification</u></p> <p>Company B knows about the implications of not defining a project around their work. A major risk is there is no defined project around what needs to be developed for the system. It's an on-going development process that has no defined ending. A major risk for both the software development company and Company B.</p> <p><u>Stakeholder involvement</u></p> <p>Currently there is no centre point of support for the information system. Users of the system contact developers directly and does not go through a structured process of identifying the problem, finding a solution then giving the solution to developers for implementation. They are considering a central contact point.</p>
Requirements Analysis	<p><u>Defining requirements</u></p> <p>Currently the outsourced software development company determines what the requirements are for Company B's information system. They are considering who has control over the final decisions (and specifications) of the changes to the system.</p> <p><u>Documentation</u></p> <p>What documentation is already available about the system? Currently there is a small spreadsheet outlining some requirements, should there be an application specification? Or a previous service specification on what originally was developed for the system? They are considering implementing more documentation about their business processes and technical specifications to help understanding what the system does.</p>
System Analysis and Design	<p><u>Documentation</u></p> <p>The application specification document is over a year out of date due to the system analyst leaving Company B. There is no structured way to designing the system. They are considering what type of documentation is needed to help Analysts determine the specifications of the system.</p>

	<p><u>Why not ERP?</u></p> <p>Perhaps having a off the shelf ERP like SAP might reduce the development of a customized information system, but they may cause other issues like change management and data migration. There will still be a need for a role of an analyst to make sure SAP is aligned with their business processes.</p>
System Development	<p><u>Testing</u></p> <p>There are no testing procedures/standards used apart from simple unit testing once a piece of logic has been developed. There is no structured testing procedures that include such scenarios like peer testing/programming.</p> <p><u>Stakeholder involvement</u></p> <p>Company B are considering how much involvement should stakeholders have into testing the system? Users within the company tend to raise issues regarding the usability of the system. Does the software development company collate these issues to improve the usability of the system? There could be political control over who should make these decisions.</p>
Integration/Deployment	<p><u>Staging Environments</u></p> <p>The system has a development environment, but no pre-production or staging environment. When new updates are applied to the live environment, User Acceptance Testing (UAT) and normal testing is then performed on the live environment. This could be a potential risk of releasing software that hasn't been completely tested (quality issues). Company B are considering having a staging environment, but this also depends on their decision to outsource or host their environments in-house.</p>
Quality, Evaluation and Testing	<p><u>Performance Indicators</u></p> <p>There are no defined performance indicators for the system. There are also no testing procedures to include Company B's staff (User Acceptance Testing) or formal testing procedures within the software development company. This may cause issues in the long term due to lack of testing documentation when system needs to be audited. This depends on their decision to outsource the development or develop in-house. The decision will lead to either implementing their own standards or the standards of another company.</p> <p><u>Technology</u></p> <p>They are considering the possibilities of researching into how new</p>

	technologies can enhance their system from a quality and performance view.
Maintenance/Support	<p><u>Service Level Agreements/Procedures</u></p> <p>There are currently no service level agreements available, users contact developers directly, no response times or agreements in place. Developers are not obligated to fix any problems. Company B are considering putting a service level agreement in place.</p>

COMPANY C

The following table shows the evolving considerations for Company C's information system. Company C provides medical training to companies throughout New Zealand. The Company C information system handles the day-to-day administration for the company. It also includes a learning management system to handle online learning and document management.

Common phases	Considerations
Initial Planning	<p><u>Managing</u></p> <p>Company C are considering:</p> <ul style="list-style-type: none"> • Developing projects and putting budgets against projects to suit their requirements • Use a range of open source technologies to reduce costs • Source expert knowledge about technological areas in online learning and database management • Managing risk
Requirements Analysis	<p><u>Requirements/Strategic Directions</u></p> <p>There was no strategic direction on where they want to position their company in terms of their technology and systems. The system was originally developed to resolve their traditional paper issues. Therefore it was not developed to handle future capabilities when business processes change or the business strategic direction changes.</p> <p>Company C are considering:</p> <ul style="list-style-type: none"> • Reviewing their strategic direction for their system • Increase documentation (modelling/artifacts) • Gaining technical knowledge to better understand their own requirements
System Analysis and Design	<p><u>Documentation</u></p> <p>Company C have considered developing service and application specifications. Currently there is no service specification or application specification documentation. No artifacts used to demonstrate</p>

	<p>requirements and no prototyping. The system has been developed on some initial requirements with no follow up documentation. There is no structure to designing the system, e.g. no flow charts etc</p>
System Development	<p><u>Testing</u></p> <p>No defined testing procedures. No testing documentation or any documentation involving User Acceptance Testing (UAT). There has been no client interaction for feedback or testing of the existing system. Company C are considering having more involvement in functional testing of the system.</p> <p><u>Standards/Technology</u></p> <p>Company C have considered using rapid development technology that helps reduce cost in maintenance and requirement changes to the system.</p>
Integration/Deployment	<p><u>Environments</u></p> <p>No pre-production environment.</p> <p>The use of Open Source software for their learning management system and a custom made website for their marketing/administration may cause issues of communication between the two applications. Company C are considering how they can merge all systems into one system or making the systems interoperable between them.</p>
Quality, Evaluation and Testing	<p><u>Performance/Evaluation/Testing</u></p> <p>Company C are considering:</p> <ul style="list-style-type: none"> • Implementing performance indicators on system performance and usability • Implementing types of testing methods • Gathering client feedback and evaluations to trigger new requirements (improvements) to the system.
Maintenance/Support	<p><u>Service Level Agreements</u></p> <p>There are no contracts or support agreements put in place to handle user issues with any of the systems available through Company C. Company C are considering how to manage support for their systems.</p>

COMPANY D

The following table shows the evolving considerations for Company D. Company D has a web based information brokerage system for trading grain and seed products nationally and internationally.

Common phases	Considerations
Initial Planning	<p data-bbox="548 205 669 231"><u>Feasibility</u></p> <p data-bbox="548 275 1308 338">Company D started as a “start-up” company, currently there is no income stream so they are considering:</p> <ul data-bbox="597 380 1360 583" style="list-style-type: none"> <li data-bbox="597 380 1149 405">• Feasibility of making changes to the system <li data-bbox="597 415 1338 478">• Determining strategic direction/marketing to align with their system <li data-bbox="597 489 1247 514">• Risk Management with future changes to the system <li data-bbox="597 525 1360 583">• Documentation (Service and Functional Specifications, Budget Plans)
Requirements Analysis	<p data-bbox="548 634 831 659"><u>Managing Requirements</u></p> <p data-bbox="548 703 1360 940">Company D have considered managing future changes to the system and the risks involved from a technical and business view. A structured System Development Life Cycle (SDLC) methodology was used in initial development, but scope creep was introduced and requirements changed during development, causing huge problems. They are considering reviewing their processes to try and improve on future developments.</p>
System Analysis and Design	<p data-bbox="548 957 847 982"><u>Processes/Documentation</u></p> <p data-bbox="548 1026 1344 1155">Company D are considering have formal design processes and documentation including artifacts and prototyping methods. There have been no formal design processes, there was some Just In Time (JIT) prototype designs, but no solid documentation.</p>
System Development	<p data-bbox="548 1171 808 1197"><u>Affordable Technology</u></p> <p data-bbox="548 1241 1383 1369">Company D now have considered the use of new technology for system development. Because of budget constraints, open source software was used to develop the system but problems with performance and security have made them consider other approaches.</p> <p data-bbox="548 1419 646 1444"><u>Skill set</u></p> <p data-bbox="548 1488 1344 1551">After reviewing the developers skill set. Company D are considering how to up skill their developers.</p>
Integration/Deployment	<p data-bbox="548 1566 734 1591"><u>Version Control</u></p> <p data-bbox="548 1635 1383 1764">No formal version control implemented. There was no pre-production environment and no development environment. They only used the live environment for development. Company D are considering how they can manage their source code around version control.</p> <p data-bbox="548 1814 711 1839"><u>Environments</u></p>

	<p>There was only a live environment, no development or pre-production environments. This was a huge risk to quality but also bad practice for deploying releases/builds.</p> <p>Company D are now considering how they can setup testing/pre-production environments.</p>
Quality, Evaluation and Testing	<p><u>Standards</u></p> <p>Company D are considering how they can change their system to comply with web standards (W3C) and other ISO standards. This was triggered from the feedback of several clients who use their system.</p> <p><u>Testing</u></p> <p>Testing sessions and procedures were identified and used, but in most cases, the client was not involved enough in the testing sessions (UAT). There was no testing procedures defined, e.g. UAT, release testing, unit testing or peer testing. Company D are reviewing their approach to testing new and existing changes to their system.</p>
Maintenance/Support	<p><u>Support Agreements</u></p> <p>Because their system is a "start-up" project, the decision was made to not to have any on-going support or ability to request changes (MAC) until the company was making enough income to cover the costs.</p> <p>Company D are considering what impact does this have on the system and the users.</p>

COMPANY E

The following table shows the evolving considerations surrounding the Company E's Learning Management System. The learning management system caters for online learning, document management and managing distance students.

Common phases	Considerations
Initial Planning	<p><u>Defined Budget</u></p> <p>Company E are considering defining formal projects to help understand their budget requirements. They are considering how they can manage projects better within the department.</p> <p><u>Skill sets</u></p> <p>Company E are reviewing their department to identify the necessary skill set to continue development changes to their system.</p>
Requirements Analysis	<p><u>Identified Client</u></p> <p>Company E are considering managing the requirements by creating a</p>

	<p>role where they are responsible for making decisions on the system requirements and specifications. Although their requirement documentation was good, there really was no client for the size of the organisation.</p>
System Analysis and Design	<p><u>Right Methodology</u></p> <p>There was no defined methodology used when building the software, e.g. uses a traditional waterfall method, could have been improved by using a different approach. Company E are considering defining their own practices and standards across multiple projects.</p>
System Development	<p><u>Technology</u></p> <p>Company E have considered the different types of technology for developing their system. This is determined by budget requirements, available staff and existing architecture.</p> <p><u>Testing</u></p> <p>There was a lack of stakeholder participation when testing sessions were arranged. This had an impact of the quality of the system. Company E are reviewing their testing procedures and creating a framework of testing for both software and academic relating areas.</p>
Integration/Deployment	<p><u>Environment</u></p> <p>There was no pre-production environment. No structured processes to release the software to the public. Company E is working towards implementing processes for release their own software and processes in upgrading existing open source software.</p>
Quality, Evaluation and Testing	<p><u>Stakeholder Participation/Performance Measures/Testing Methods</u></p> <p>Company E are considering:</p> <ul style="list-style-type: none"> • Formulating a special interest group to help participate in the quality of the system • Involve stakeholders more for evaluation of the system • Implementing standardised testing procedures that can be used across multiple projects
Maintenance/Support	<p><u>Support Requirements</u></p> <p>Company E are reviewing their requirements for technical support and academic support throughout the system. They have a IT services department that doesn't directly support the system as its part of a different department. They are considering how the support can be driven into the same services.</p> <p>Company E are considering defining service level agreements (SLA) and</p>

	<p>support measures to improve the quality of the system.</p> <p>Company E are also considering the processes of requesting changes to the system.</p>
--	--

SUMMARY OF CONSIDERATIONS

To determine what the consideration similarities are between each project, we used a visual text algorithm called Document Arc Diagramming that illustrates similarities between common words or phrases through a visual medium.¹¹ The common words or phrases were taken from the organisation considerations for each common phase.

Common Phases	Summary Considerations
Initial Planning	<p><u>Time</u></p> <p>A common consideration between each project is timeframes. This is time between planning and upgrading a system. This also includes time versus available budgets.</p> <p><u>Budgets/Costs</u></p> <p>Defining costs over time, costs over quality.</p> <p><u>Right Team</u></p> <p>Outsource? Dedicated software team? Over time and quality and cost?</p> <p><u>Knowing the product</u></p> <p>Know the product first versus developing a system then after knowing the product.</p> <p><u>Managing/Identifying risks</u></p> <p>Considering how to manage risks versus changes to the system from technical and business views.</p> <p><u>Stakeholder involvement</u></p> <p>Considering how much stakeholder involvement is needed during a project.</p>
Requirements Analysis	<u>Identifying/Managing Requirements</u>

¹¹ You can look at Appendix B (*APPENDIX B – DOCUMENT ARC DIAGRAM RESULTS*) for the actual diagram results using an online Document Arc tool.

	<p>Considering how to manage new requirements, and how they fit into the existing system.</p>
System Analysis and Design	<p><u>Standards/Processes</u></p> <p>Using standard procedures across organisations to analyse and translate requirements into business and technical specifications.</p>
System Development	<p><u>Standards Testing Methods</u></p> <p>Considering using a range of different testing methods to testing functionality from different angles.</p> <p><u>Technologies used</u></p> <p>Considering using latest technology to reduce development time and cost.</p>
Integration/Deployment	<p><u>Environments</u></p> <p>Considering having several environments to develop, testing and deploy systems.</p> <p><u>Release Standards</u></p> <p>Having standard procedures on deploying systems across multiple projects.</p>
Quality, Evaluation and Testing	<p><u>Standards Testing Methods</u></p> <p>Considering using a range of different testing methods to testing functionality from different angles (i.e. regression testing, performance testing).</p> <p><u>Measuring quality techniques</u></p> <p>Considering having processes to measuring quality in forms of customer and stakeholder feedback and evaluations.</p>
Maintenance/Support	<p><u>Support Agreements</u></p> <p>Considering having support agreements in place (i.e Service Level Agreements) to cover support requirements.</p>

FINDINGS - MATCHING AGILE AND OOAD TO COMMON CONSIDERATIONS

We have now isolated the considerations by categorising them into specific common phases from a collection of system development methodologies. We can then identify the relationship between these considerations and specific Agile and OOAD characteristics to formulate the recommendations.

The following table identifies Agile and OOAD characteristics against the common system development methodology phases and the information system organisational considerations:

Common Phases	IS Considerations	Agile Characteristics	OOAD Characteristics
<u>Initial Planning</u>	Timeframes Budgets/Costs Product Knowledge Managing Risks Stakeholder Involvement	Maximize ROI practice Agile Model Driven Development Agile Approach to Budgeting and Scheduling Prioritise Stack Active Stakeholders	Iterative Development Model Driven Approach
<u>Requirements Analysis</u>	Identifying Requirements Managing Requirements	Just Barely Good Enough modelling Just in Time Modelling Agile Requirement Best Practices Agile Requirements Change Management Agile Model Driven Development (AMDD) Prioritising Stack	Problem Domain UML Business Processes Modelling Notation RUP Principles (Manage Requirements) Cluster Charts
<u>System Analysis and Design</u>	Standards/Processes	Just Barely Good Enough (JBGE) Freeform Diagrams, Use of the right artifacts UI Agile prototype process	Component Based Architectures (RUP) Use of UML Code Generators
<u>System Development</u>	Standards	Test Driven Design	RUP – Verify Quality

	Technologies	Prove your model with Code Just Barely Good Enough (JPGE) modelling Stakeholder Involvement Practices	Comparing UML Models with Developed code Code Generators Object Oriented Approaches, Service Oriented Approaches
<u>Integration/Deployment</u>	Environments Release Standards	Agile SDLC JIT Small iterations	Control Changes (RUP)
<u>Quality, Evaluation and Testing</u>	Standards testing methods Measuring quality techniques	Turning JBJP into quality modelling? Test Case Management Test Driven Development Prove it with Code Agile Testing Strategies	Use Case verification against requirements
<u>Maintenance</u>	Service Level Agreements/Support Agreements	Support Releases/Support System	Component Driven Development

APPENDIX A - REFERENCES

- Scott Ambler, December 12 2006. Dr. Dobb's Portal, *Agile Testing Strategies* - <http://www.ddj.com/dept/debug/196603549?cid=Ambyssoft>
- Scott W. Ambler, March 03 2007. Agile Modelling, *Agile Model Driven Development* - <http://www.agilemodeling.com/essays/amdd.htm>
- Scott W. Ambler, March 03 2007. Agile Modelling, *Agile Modelling Principles* - <http://www.agilemodeling.com/principles.html>
- Scott W. Ambler, January 2007. Dr. Dobb's Agile Modelling Newsletter, *Rethinking How You View Requirements Management* - <http://www.ddj.com/dept/architect/196902703?cid=Ambyssoft>
- Scott Ambler, October 11 2004. Dr. Dobb's Portal. *Agile Requirements Management: Less is More* - <http://www.ddj.com/dept/architect/184415221?cid=Ambyssoft>
- Scott Ambler, February 15 2006, Dr. Dobb's Portal. *Supersize Me: Scaling Agile Software Development* - <http://www.ddj.com/dept/architect/184415491?cid=Ambyssoft>
- Scott W. Ambler, March 03 2007. Agile Modelling, *Agile Architectural Modelling* - <http://www.agilemodeling.com/essays/agileArchitecture.htm>
- Scott W. Ambler, Ron Jeffries (February 01, 2001) – *Wiley Publisher*. Agile Modelling, Effective Practices for Extreme Programming and the Unified Process by Scott W. Ambler, Ron Jeffries
- Kim Hamilton, Russell Miles (April 2006) – *O'Reilly Publisher*. Learning UML 2.0
- Kurt Bittner, Ian Spence (June 27 2006) - *Addison Wesley Professional Publisher*. Managing Iterative Software Development Projects
- R. Dennis Gibbs (July 27 2006) – *IBM Press Publisher*. Project Management with the IBM® Rational Unified Process®: Lessons from the Trenches
- Martin Wattenberg (IBM Research).
- Arc Diagrams: Visualising Structures in Strings -<http://www.research.ibm.com/visual/papers/arc-diagrams.pdf>
- Scott W Ambler, March 2007. Agile Modelling, *Agile Unified Process* - <http://www.ambysoft.com/unifiedprocess/agileUP.html>
- Scott W Ambler, March 03 2007. Agile Modelling, *Agile Model Driven Development* - <http://www.agilemodeling.com/essays/amdd.htm>
- McConnell, Steve (1996). *Rapid Development: Taming Wild Software Schedules*, 1st ed., Redmond, WA: Microsoft Press. ISBN 1-55615-900-5.
- Wiegers, Karl E. (2003). *Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle*, 2nd ed., Redmond: Microsoft Press. ISBN 0-7356-1879-8.
- Andrew Stellman and Jennifer Greene (2005). *Applied Software Project Management*. Cambridge, MA: O'Reilly Media. ISBN 0-596-00948-8.
- Contributor Melonfire (September 22 2006). *TechRepublic*, Understanding the pros and cons of the waterfall model of software development - http://articles.techrepublic.com.com/5100-3513_11-6118423.html?part=rss&tag=feed&subj=tr#
- Barry W. Boehm. A spiral model of software development and enhancement - <http://www.sce.carleton.ca/faculty/ajila/4106-5006/Spiral%20Model%20Boehm.pdf>
- Walter Maner (Revised March 15, 1997). Rapid Application Development - <http://csweb.cs.bgsu.edu/maner/domains/RAD.htm>

Benjamin Kok Swee Khoo. A survey of major software development methodologies - <http://userpages.umbc.edu/~khoo/survey2.html>

Reed Sorensen (1994). A comparison of software development methodologies - <http://www.stsc.hill.af.mil/crosstalk/1995/01/Comparis.asp>

Dan Brandon (2006) - *IBM Press Publisher*. Project Management for Modern Information Systems

Susan Snadaker (September 2005) - *Syngress Publisher*. How to Cheat at IT Project Management

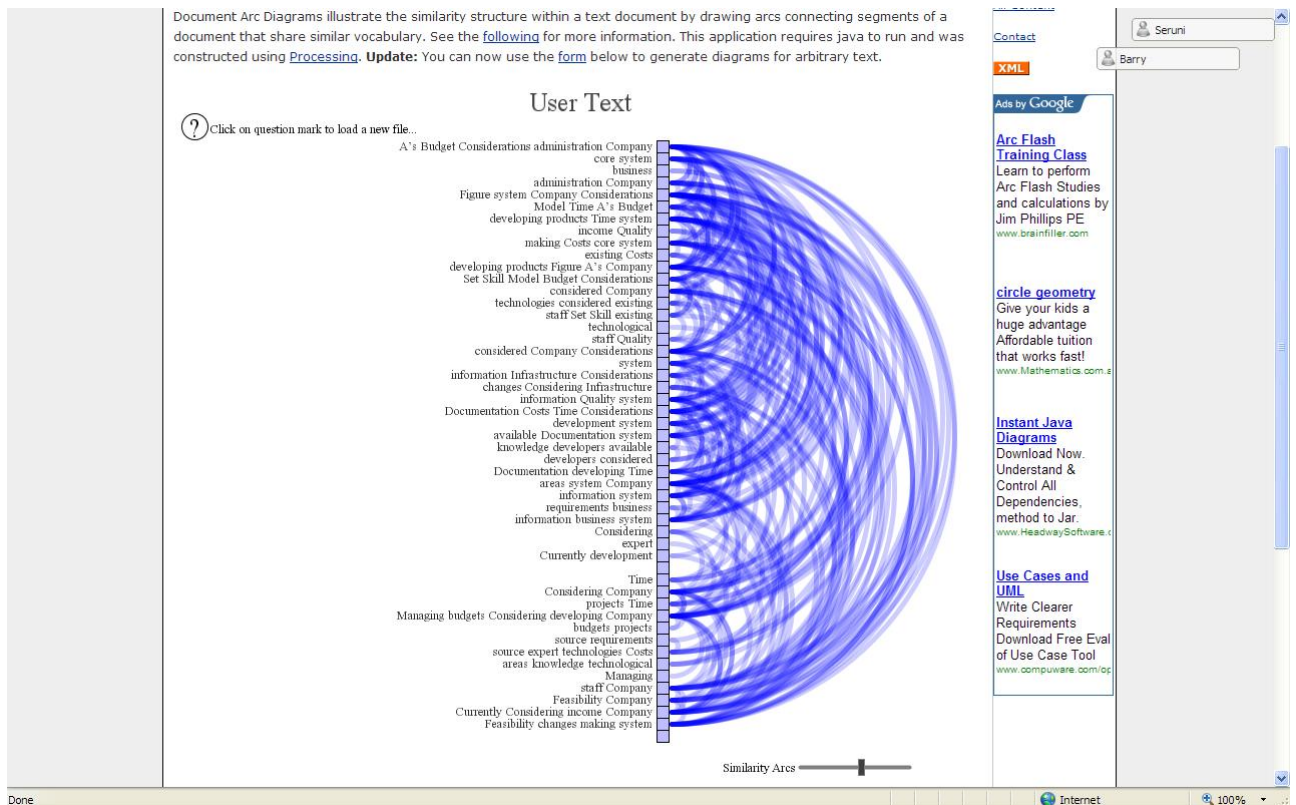
Scott W. Ambler (December 04, 2005). A Manager's introduction to the Rational Unified Process (RUP) - <http://www.ambysoft.com/downloads/managersIntroToRUP.pdf>

APPENDIX B – DOCUMENT ARC DIAGRAM RESULTS

Document Arc Diagramming is a visualisation method to represent complex repetition patterns in string data. This helps determine the common considerations between each project used in this paper. The following screen shots (grouped by common phase) are the results of using the Document Arc Web Application to visually see repetition patterns.

- * <http://www.research.ibm.com/visual/papers/arc-diagrams.pdf>
- * <http://www.neoformix.com/Projects/DocumentArcDiagrams/index.html>

Initial Planning

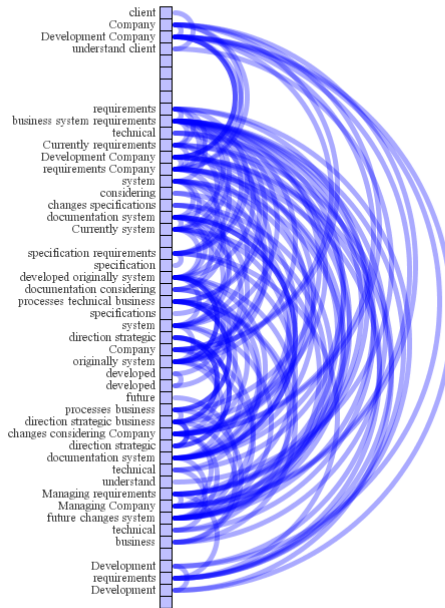


Requirements Analysis

Document Arc Diagrams illustrate the similarity structure within a text document by drawing arcs connecting segments of a document that share similar vocabulary. See the [following](#) for more information. This application requires java to run and was constructed using [Processing](#). **Update:** You can now use the [form](#) below to generate diagrams for arbitrary text.

? Click on question mark to load a new file...

User Text



Similarity Arcs

Contact: Seruni, Barry

XML

Ads by Google

[Arc Flash Training Class](#)
Learn to perform Arc Flash Studies and calculations by Jim Phillips PE
www.brainfiller.com

[circle geometry](#)
Give your kids a huge advantage Affordable tuition that works fast!
www.Mathematics.com

[Instant Java Diagrams](#)
Download Now. Understand & Control All Dependencies, method to Jar.
www.HeadwaySoftware.com

[Use Cases and UML](#)
Write Clearer Requirements Download Free Eval of Use Case Tool
www.compuware.com/uc

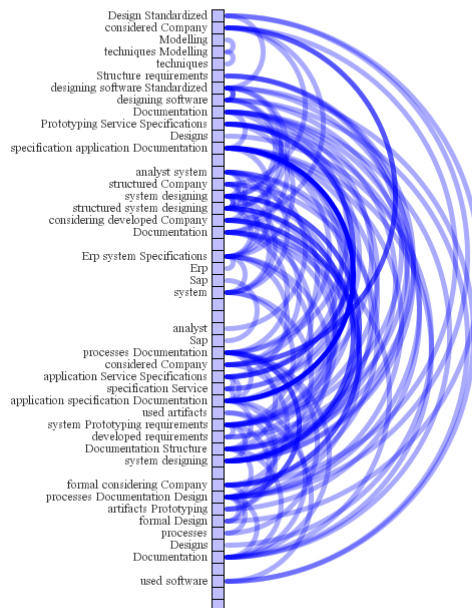
Internet 100%

System Analysis and Design

Document Arc Diagrams illustrate the similarity structure within a text document by drawing arcs connecting segments of a document that share similar vocabulary. See the [following](#) for more information. This application requires java to run and was constructed using [Processing](#). **Update:** You can now use the [form](#) below to generate diagrams for arbitrary text.

? Click on question mark to load a new file...

User Text



Similarity Arcs

All Content

Contact: Seruni, Barry

XML

Ads by Google

[Arc Flash Training Class](#)
Learn to perform Arc Flash Studies and calculations by Jim Phillips PE
www.brainfiller.com

[circle geometry](#)
Give your kids a huge advantage Affordable tuition that works fast!
www.Mathematics.com

[Instant Java Diagrams](#)
Download Now. Understand & Control All Dependencies, method to Jar.
www.HeadwaySoftware.com

[Use Cases and UML](#)
Write Clearer Requirements Download Free Eval of Use Case Tool
www.compuware.com/uc

Internet 100%

System Development

Document Arc Diagrams illustrate the similarity structure within a text document by drawing arcs connecting segments of a document that share similar vocabulary. See the [following](#) for more information. This application requires java to run and was constructed using [Processing](#). **Update:** You can now use the [form](#) below to generate diagrams for arbitrary text.

ⓘ Click on question mark to load a new file..

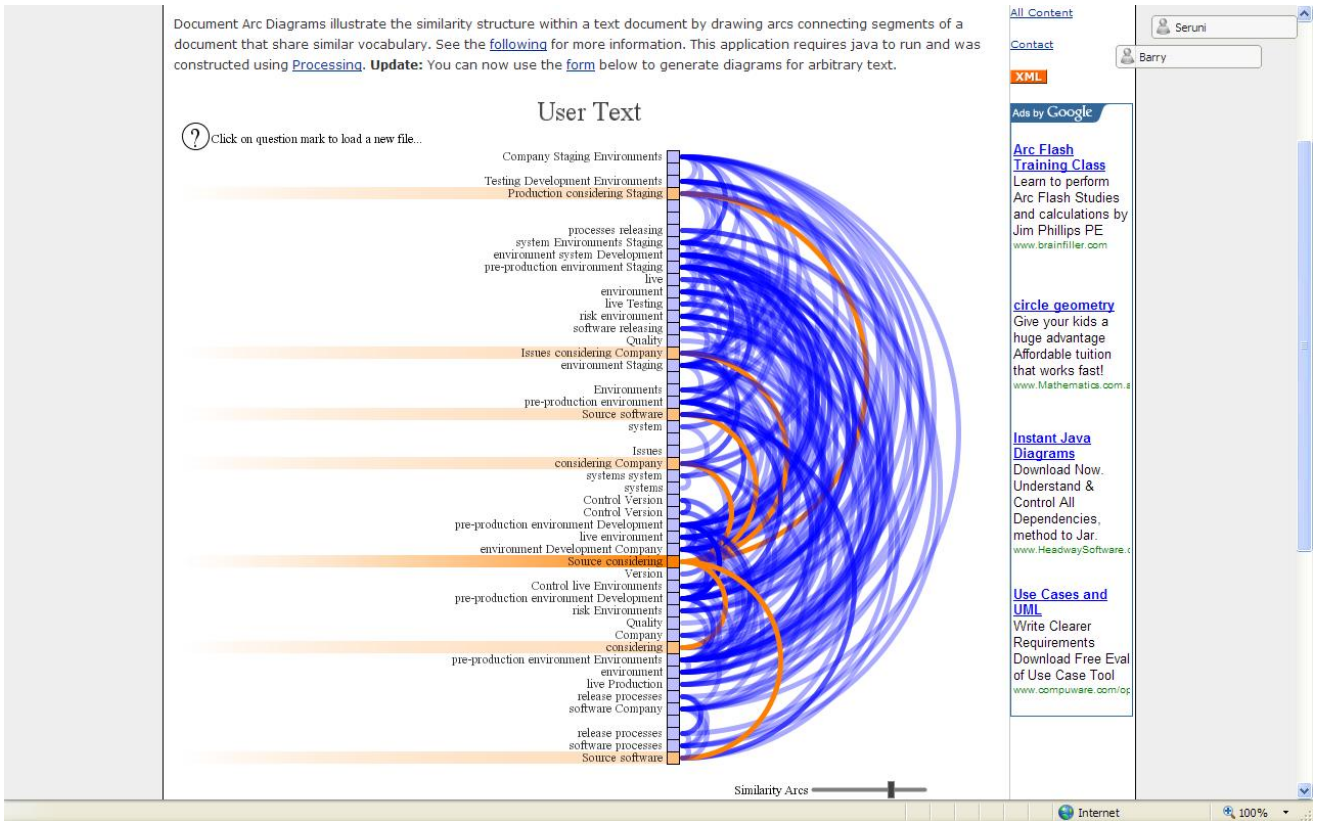
User Text

Company Testing
considered Testing
system
unit Testing
Technologies Company
defined considering architecture
Technologies
existing architecture Testing
used Testing
unit
Testing
developed Testing
procedures
involvement considering Company
involvement Testing
system
Company
usability issues
software development system
issues Company
usability system
defined Testing
development procedures Testing
using unit Testing
documentation developed Testing
documentation
Testing
existing Testing
considering system Company
involvement Testing
system Company
using considered
technology development
technology considered Company
technology system
development
software used
system
Skill
set Skill developed

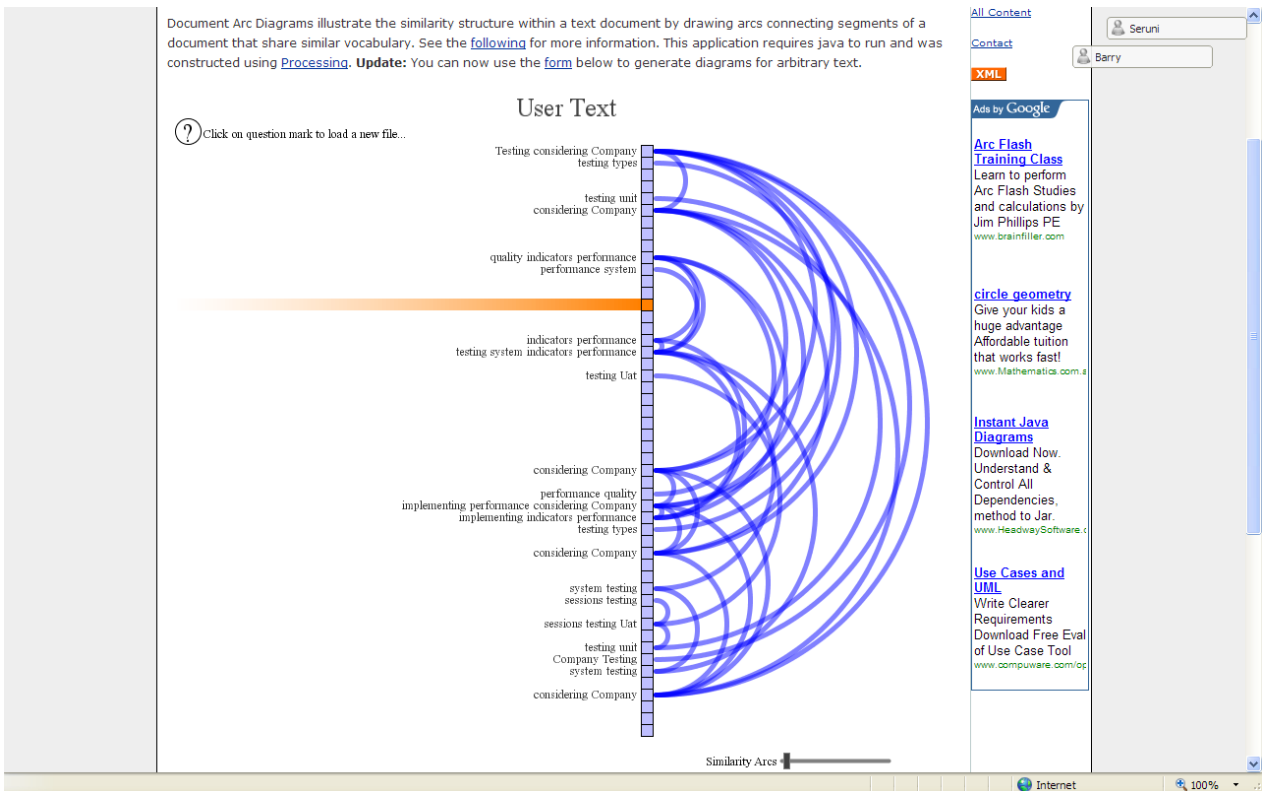
Similarity Arcs

Internet 100%

Integration/Deployment



Quality, Evaluation and Testing



Maintenance/Support

Document Arc Diagrams illustrate the similarity structure within a text document by drawing arcs connecting segments of a document that share similar vocabulary. See the [following](#) for more information. This application requires java to run and was constructed using [Processing](#). **Update:** You can now use the [form](#) below to generate diagrams for arbitrary text.

? Click on question mark to load a new file...

User Text

System Support
considering Company
manage Support
agreements clients
manage
Company
considering
manage
clients
systems
considering System
Support
Level Service
Level Service
available agreements
users
developers
place agreements
developers
considering Company
Service
Level
Level agreements
Support
place
available systems
Company
manage considering
systems Support
agreements Support
System
Support
Company
considering Company
System
users Support

Similarity Arcs

All Content
Contact
XML
Ads by Google
[Arc Flash Training Class](#)
Learn to perform Arc Flash Studies and calculations by Jim Phillips PE
[www.brainfiller.com](#)
[circle geometry](#)
Give your kids a huge advantage
Affordable tuition that works fast!
[www.Mathematics.com.s](#)
[Instant Java Diagrams](#)
Download Now.
Understand & Control All Dependencies, method to Jar.
[www.HeadwaySoftware.c](#)
[Use Cases and UML](#)
Write Clearer Requirements
Download Free Eval of Use Case Tool
[www.compuware.com/op](#)

Internet 100%