Office of the Government Chief Information Officer
The Government of the Hong Kong Special Administrative Region

PRACTICE GUIDE
FOR
AGILE SOFTWARE DEVELOPMENT
[G62]

Version: 1.3

May 2019

Distribution	
Copy No.	**Holder**
1	Office of the Government Chief Information Officer

Amendment History				
Change Number	**Revision Description**	**Pages Affected**	**Rev. Number**	**Date**
1	As detailed in 1.01 to 1.11		1.1	December 2016
1.01	Add "List of Figures and Tables"	After "Table of Contents" *(New)*		
1.02	Change "Section 1" to "Part I" , "Section 2" to "Part II", "Section 3" to "Part III" and "Appendics" to "Appendix A"	Preface - Structure of this Guide		
1.03	Rename the section name from "Assessing the Suitability of Agile for the Project" to "Selection of Project to Use Agile"	4		
1.04	Add "Best fit for using Agile" in the section of "Selection of Project to Use Agile" and a new table "List of project criteria"	4.1, Table 2 added		
1.05	Add a new table - "General Tips for User Involvement and Empowerment" in the section of "User Involvement and Empowerment"	5.2 - Table 3 added		
1.06	Add a general tip for Procuring Agile System Development Services in the section of "Procuring Agile System Development Services"	6.1 - Table 5 (b) added		
1.07	Add a new figure - "System Analysis and Design in Agile" in the section of "System Analysis and Design"	7 - Figure 6 added		
1.08	Add a new table – "General Tips for Requirement Defintion" in the section of "Verifying Requirement"	7.1.4 - Table 6 added		

Amendment History				
1.09	Update the content in Section of "Define Functional Specification"	7.2.1		
1.10	Add a general tip for timebox planning	8.2.1 - Table 8 (d) added		
1.11	Add the "Reference" section	Reference *(New)*		
2	As detailed in 2.01 to 2.03		1.2	September 2018
2.01	Update for Agile as the preferred software development approach	2		
2.02	Update to remove criteria related to project size for the "Best fit for using Agile" section	4.1 – Table 2 updated		
2.03	Update to remove criteria related to project criticality for the "Assessing the Suitability of Agile" section	4.21, 4.23 updated		
3	Add the content of Agile approach with design thinking mindset	1.2 Paragraph added	1.3	May 2019

Table of Contents

List of Figures & Tables

CONVENTIONS

Table 1 - List of Acronyms used throughout the Practice Guide For Agile Software Development

Abbreviation	Full Name
Agile	Agile Software Development Methodology
BA	Business Analyst
B/D(s)	Bureaux and Departments
FAF	Funding Application Form
OGCIO	The Office of the Government Chief Information Officer
PAT	Project Assurance Team
PM	Project Manager
PMP	Project Management Plan
PRL	Prioritised Requirements List
PSC	Project Steering Committee
SA	Systems Analyst
SArch	Systems Architect
SDLC	System Development Life Cycle
SPR	Stores and Procurement Regulations
TDD	Test-Driven Development
TSO	Technical System Option
UAT	User Acceptance Testing

PREFACE

(a) Since the 1990s, the Government has progressively adopted a number of system development methodologies for implementation of IT systems.

(b) With the rapid evolvement of technologies, new types of applications such as mobile applications and e-business applications have become popular. These new types of applications require relatively shorter time to delivery and are very often based on requirements that are of high uncertainties or not well defined. New methodologies are thus required for the development and implementation of these types of applications. One of such potential methodologies is the Agile software development methodology (Agile).

(c) The Office of the Government Chief Information Officer (OGCIO) commissioned in 2012 a consultancy study to assess the practicability of adopting Agile in system development projects in the Government. Based on the findings from the consultancy study, it concluded that Agile was practical and adoptable for the Government and on the strength of its recommendation, Agile was applied in some pilot projects to gain and consolidated experiences for reference by bureaux/departments (B/Ds).

(d) This "Practice Guide for Agile Software Development" ("This Guide") aims to illustrate the Agile practices and provide guidance to B/Ds on adopting Agile for implementation of IT systems. It was developed based on common Agile practices in the industry and the experiences gained from the pilot projects of B/Ds. This Guide should be suitably adopted by B/Ds to meet their own project needs.

Scope and Target Readers of This Guide

This Guide is applicable to the implementation of IT systems in the Government. It's target readers include the following:

i) Project Steering Committee (PSC) members who are the ultimate decision makers for the project;

ii) Project Assurance Team (PAT) members who provide advisory support to the PSC and are responsible for overseeing project progress and managing quality assurance activities;

iii) Project managers including both Internal Project Manager (Internal PM) and the contractor's project manager if the project is outsourced, who are responsible for planning and managing the implementation of IT systems;

iv) Business Analysts (BAs) who are responsible for performing the business analysis functions for IT system development projects;

v) Project team members such as Systems Analysts (SA), Programmers and, if applicable, Systems Architects (SArch) who are responsible for the design and implementation of the IT system using either internal or external resources; and

vi) Users who provide requirements and feedback on the IT system during its implementation.

Structure of This Guide

This Guide is organised into three parts and one appendix:

i) **PART I - INTRODUCTION TO AGILE**

Provides an overview and the basic concepts of Agile approach.

ii) **PART II - ADOPTING AGILE IN IT PROJECT**

Illustrates the assessment criteria of adopting Agile to implement an IT system, and the planning and preparation work for adopting Agile in an IT project during project initiation stage.

iii) **PART III - USING AGILE FOR SYSTEM DEVELOPMENT**

Introduces the processes and activities to be carried out as well as the outputs to be produced in using Agile for system development. It also describes the roles and responsibilities of project team members, BA and users involved in various project implementation stages.

iv) **APPENDIX A**

Templates, checklists and sample documents are provided for reference.

PART I - INTRODUCTION TO AGILE

1 WHAT IS AGILE?

(a) Since 1990s, some software programmers decided to break away from the traditional structured approaches on software development and move towards more flexible development styles. Agile approach was then introduced to the market. In 2001, some Agile practitioners introduced four main values for developing software using Agile approach under the "Manifesto for Agile Software Development", which is provided in Appendix A-1.

(b) Agile approach is illustrated as a conceptual framework based on iterative and incremental development approach. It promotes evolutionary development and delivery of IT system using time-boxed (i.e. fixed interval) iterative approach, and encourages rapid and flexible responses to changing requirements. Figure 1 shows a typical example of iterative activities in Agile.

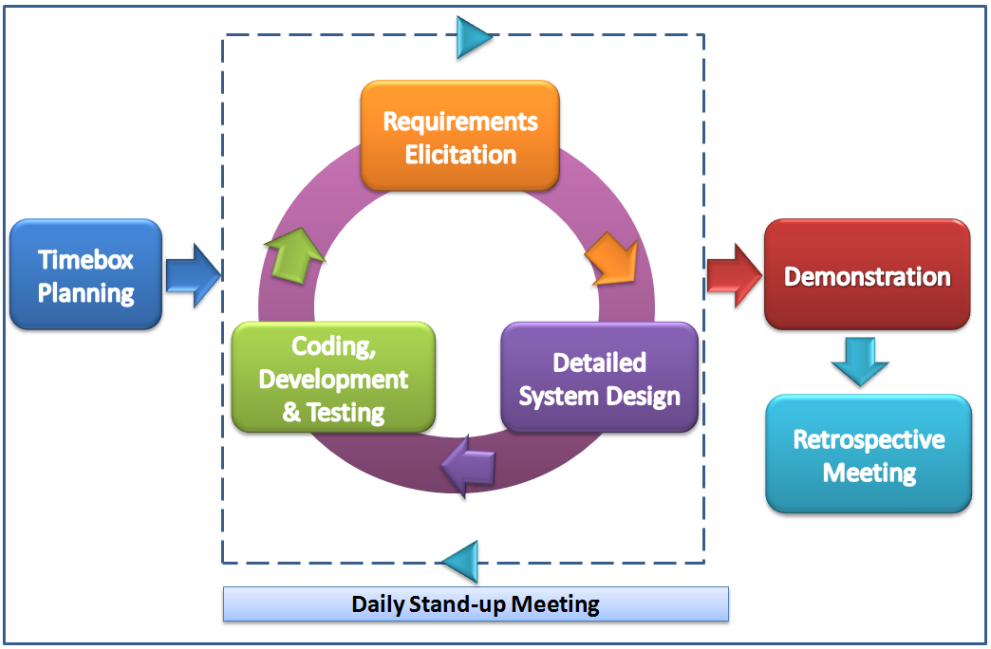

Figure 1 - Iterative Activities in Agile

(c) In general, Agile divides the System Development Life Cycle (SDLC) into multiple iterations of fixed intervals (also called timeboxes) (e.g. 2 to 4 weeks) for development. Each timebox will have pre-defined and pre-agreed target functions of the system to be developed. Under each timebox, a timebox planning meeting will be held followed by a sequence of iterative activities including requirements elicitation, detailed system design and coding and development of each target function. At the end of the timebox, a demonstration and a retrospective meeting will be performed to deliver the target functions.

(d) The initial timebox, which is often called "Timebox 0 (zero)", is reserved for performing high-level system analysis and design to define clearly the project scope, high-level requirements and other preparation work that are essential to start the development. This includes a high-level system and architectural design, high-level user and technical requirements elicitation, and planning and setting up of the basic system architectural model and development platform.

(e) In the subsequent timeboxes, project team members, BA and users will work closely to go through the SDLC activities. At the start of each timebox, project team members, user representative(s), programmers and the Agile coach (if any) will attend the timebox planning meeting. After that, BA will collaborate with users to help elicit detailed requirements of the target functions to be developed. Project team members including SAs and programmers will then do the detailed system design, coding, development and testing of the developed functions iteratively. At the end of each timebox, a demonstration will be conducted for the delivered functions, followed by a retrospective meeting in which the project team will discuss the issues encountered in the timebox and identify any follow-up actions for improvement. The activities in each timebox will be repeated in other timeboxes. Details of the timebox activities are given in Chapter 8. Upon completion of several timeboxes, working software comprising the multiple completed functions may be ready for release. When more and more functions are built, the entire system will gradually be delivered.

(f) In each timebox, project team members will need to report their work status (i.e. the work done and not done) to Internal Project Manager (Internal PM) through a daily short meeting. When there are changes to requirements, BA will help coordinate between user side and IT side. More flexible and rapid responses can be made by either re-scheduling the priorities of functions not yet developed or adding new timeboxes for development/enhancement if necessary.

(g) There are different types of Agile approach in the industry and the major ones are briefly explained at ***Appendix A-2*** for reference.

1.1 POTENTIAL BENEFITS BROUGHT ABOUT BY AGILE

The following are potential benefits brought about by Agile:

i) Better project tracking and monitoring

As SAs and programmers need to report their work status daily, it enables the project managers to review the work status of individual team members and closely monitor the project progress more effectively.

ii) Early discovery of project issues and problems

Daily short meeting also helps project managers to discover any project issues and problems earlier and foresee any project bottlenecks, which in turn, helps to reduce project risks by promptly responding to identified issues and problems.

iii) **Better quality of system**

The quality of system functions and components can be improved with early and active user involvement and feedback. Detailed requirements are elicited, well defined and confirmed by users at the start of each timebox before proceeding to detailed design and coding. Hence user feedback can be quickly collected at the end of each timebox. This improves users' understanding of the system and enhances its quality.

iv) **Shorter time to delivery**

Completed functions delivered from one or multiple timeboxes can be integrated to form working software that can be ready for release. This shortens the delivery time and enables early testing or trying out of the system. This may also enable early realisation of project benefits through partial launch of the completed functions if necessary without waiting for the completion of the entire system.

v) **More rapid and flexible responses to changing requirements**

If requirement changes are related to any planned functions that have not yet been developed, the changes can be made at the beginning of that corresponding timebox. If the changes are related to new requirements that are considered as essential, re-scheduling of the development priorities of functions or addition of a new timebox can be made accordingly.

1.2 AGILE APPROACH WITH DESIGN THINKING MINDSET

While agile is an approach to solution delivery, design thinking is an approach to problem finding. Design thinking emphases high degree of empathy and understanding of end users, iterative process of developing new ideas and redefining problems with the goal of identifying alternative solutions. Five distinct steps in design thinking are shown in ***Appendix A-8.***

The following are concepts of design thinking mindset while adopting Agile approach:

i) **Identify right problems to be solved**

Design thinking helps the project team to find the right problem by understanding of end users and define the problem that will give the necessary direction to proceed towards issues faced by end users.

ii) **Frequent input from end users**

Besides working on developing solutions, writing user stories, creating backlogs and demonstrating working software to the business users, the development team

could continuously seek new insights, suggestions, feedback and ideas from end users that help to better align business goals during the development process.

iii) Mutually reinforcing

Agile approach deliver solutions incrementally with rapid iterations while design thinking focus on strong user centric. Two approaches are better work together to reach optimal solution by ensuring end user needs throughout the entire development process.

PART II - ADOPTING AGILE IN IT PROJECT

2 AGILE AS THE PREFERRED SOFTWARE DEVELOPMENT APPROACH

(a) Traditionally, waterfall software development methodology is commonly used in Government IT projects for the implementation of IT systems. The method divides the system development process into phases, and the phases are performed one by one sequentially. Each phase will be completed with the target deliverables produced before proceeding to the next phase, i.e. flowing steadily downwards (like a waterfall) and will not go backwards. The software will generally be delivered at the end of the SDLC.

(b) The conceptual framework depicting Agile approach is based on iterative and incremental development which consists of multiple timeboxes as shown in Figure 2. Agile generally starts the system development with a high-level SA&D and is then followed by repeated cycles of implementation activities. Agile can facilitate early visibility of the system under development through incremental delivery i.e. working software may be developed or rolled out after completing one or a group of timeboxes.

(c) Depending on the project nature, B/Ds should flexibly adopt Agile practices and consider Agile as the preferred software development approaches. In general, project teams can adopt more than one software development approach and apply multiple practices according to project needs.

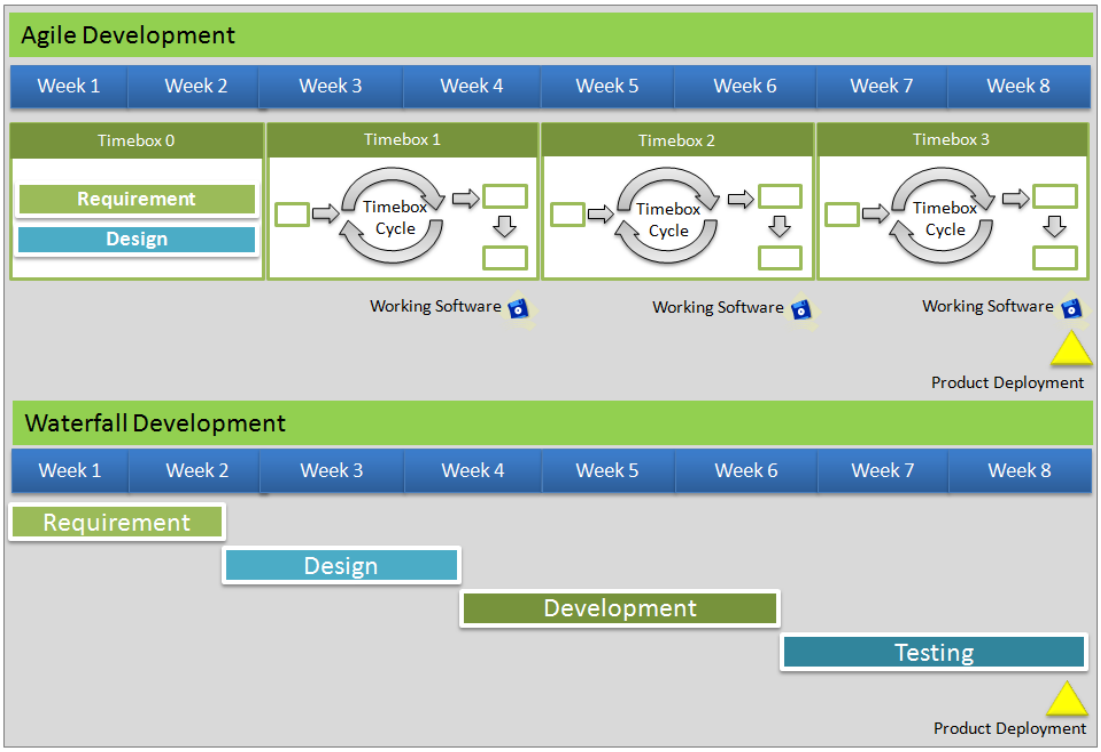

Figure 2 - Comparison between Agile & Waterfall Development Approach

3 AGILE DEVELOPMENT LIFE CYCLE IN IT PROJECT DELIVERY

(a) The life cycle of Agile project is defined according to project nature, duration and target delivery time of functions and the entire system. Typically, the number of timeboxes for each project at the System Implementation phase varies. In Agile, each timebox will complete and deliver a portion of the system. In each timebox, there will be activities including planning, elicitation of requirements, detailed system design, coding, development and testing and demonstration as well as a retrospective meeting in order to produce that portion of the system. This is in contrast to waterfall methodology in which activities are conducted and completed in a sequential manner in the SDLC.

(a) A typical agile development cycle is shown in Figure 3 in which the entire system will be rolled out at the end of the SDLC. Generally, the timebox 0 is used by project team to conduct a high-level system analysis and design and to develop the system architecture and platform of the development environment required for the start of the development work in subsequent timeboxes.

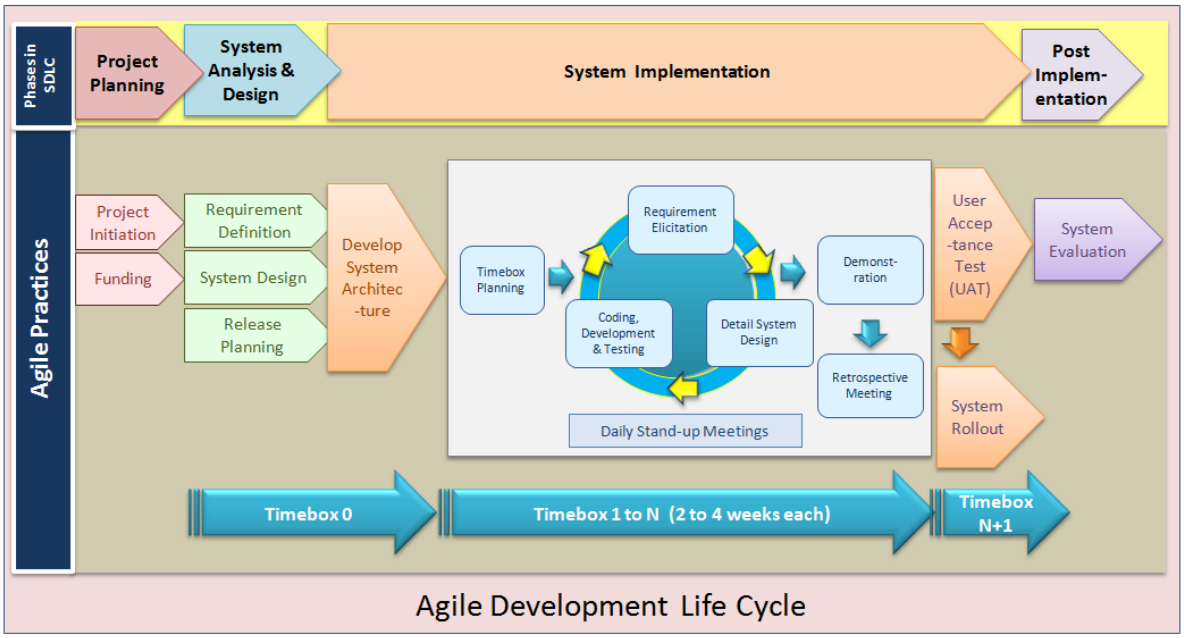

Figure 3 - Typical Agile Development Life Cycle

(b) There may also be changes in sequence of activities in the System Implementation phase if different releases of working software are required to be rolled out to allow early visualisation of the benefits. In Figure 4, it shows three examples of different combination of activities in the System Implementation phase of the cycle. Multiple user acceptance tests can be done, and several system releases can be made after completing the target groups of functions.

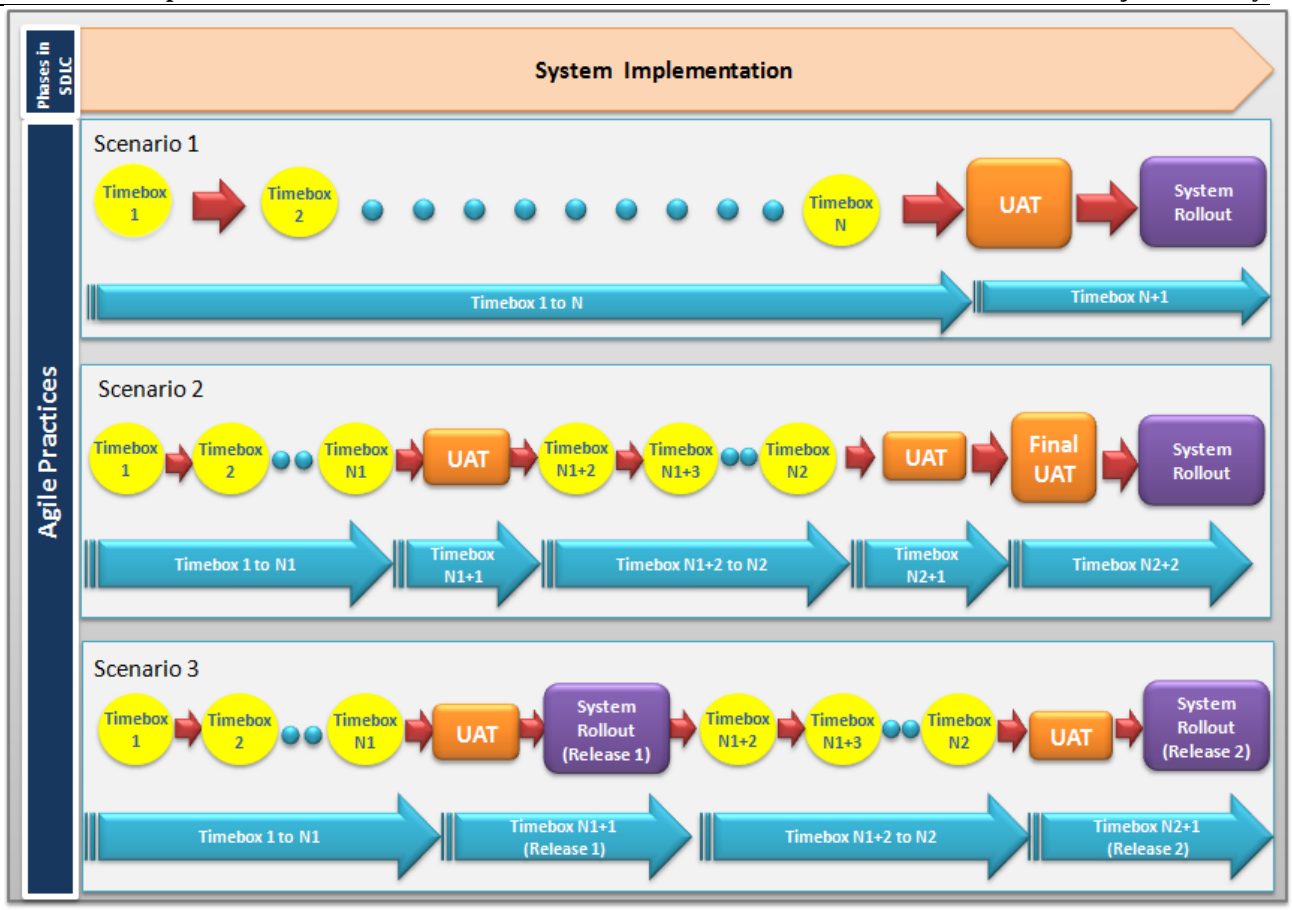

Figure 4 - Examples of different combination of activities in the System Implementation Phase of the cycle

4 SELECTION OF PROJECT TO USE AGILE

It is important to decide whether to use Agile at the outset of an IT system development project. Each project has its own characteristics and features as well as business values. Some projects appear to be more suitable for adopting Agile, while some may be more suitable for the waterfall approach.

4.1 BEST FIT FOR USING AGILE

(a) The most appropriate projects for Agile are those of / have volatile requirements, more novelty and innovation, shorter time for delivery, high extensibility, high user elements and many user interfaces.

(b) Project team should use Agile if the project meets the following criteria:

Table 2 - List of project criteria

Project Type	Basically all application system development projects and especially best fit for Government e-Service, mobile application development, and website development

4.2 ASSESSING THE SUITABILITY OF AGILE FOR OTHER PROJECTS

Besides the above listed project criteria, project team could also assess the suitability of adopting Agile according to the following two sets of criteria for consideration, the feasibility criteria and the benefits criteria.

4.2.1 FEASIBILITY CRITERIA

The feasibility criteria are used to examine whether the success factors and the project characteristics required for adopting Agile are met. Success factors refer to three aspects: project, people and technical. The project aspect examines the team size, team co-location and external dependency. The people aspect examines staff knowledge and skills, and user involvement and interaction. The technical aspect examines the availability of required Agile tools.

4.2.2 BENEFIT CRITERIA

(a) Apart from the feasibility criteria, the project characteristics should also be examined to determine whether a project can benefit most if Agile is adopted for the development of the IT system.

(b) Examples of IT systems that can benefit most from adopting Agile are systems with volatile requirements and requiring shorter time to delivery such as mobile applications and website development.

4.2.3 SUITABLE TYPES OF PROJECTS FOR AGILE

(a) Figure 5 shows the feasibility-benefit matrix which visualises different suitability levels of Agile approach for system development projects. However, the existence of negative responses to the criteria may not necessarily indicate non-feasibility. ***Appendix A-3*** shows a suitability checklist to illustrate the idea in more detail.

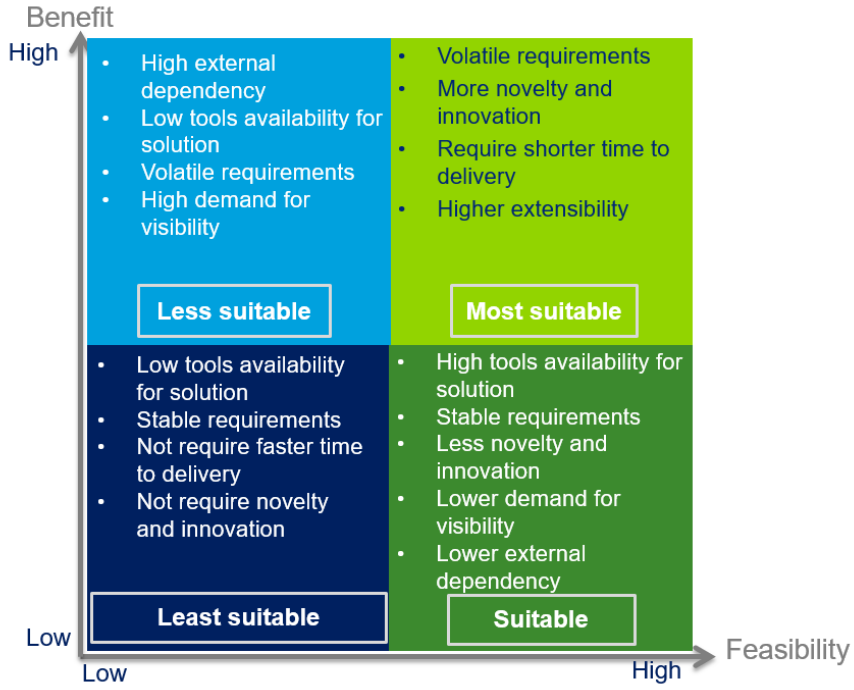

Figure 5 - Feasibility-benefit matrix

(b) As referred to the above figure, Agile is considered as "most suitable" for projects with high feasibility and high benefit. These types of projects are typically with relatively more volatile requirements, novelty/innovation and extensibility, or require shorter time to delivery. Projects for some mobile applications or website development that meet the above criteria usually belong to this group.

5 PLANNING FOR AGILE

5.1 FUNDING APPLICATION

(a) When preparing the Funding Application Form (FAF), the Internal PM should identify any additional resources required for adopting Agile, such as:-

i) user resources for constant involvement in attending meetings for each timebox during development, providing detailed requirements, conducting preliminary tests and acceptance of completed functions, and providing feedback during development in the System Implementation phase;

ii) cost for Agile training courses, Agile tools or Agile coaching services for project team members; and

iii) office accommodation for contractor's project team members to provide on-site development services, hold face-to-face daily stand-up meetings and facilitate on-line video communication.

(b) The project schedule and tasks included in the FAF should appropriately reflect the characteristics of an Agile project. Typically, a shorter period of time is allocated for performing SA&D while a longer period of time for system implementation in Agile comparing with the waterfall approach. It is because Agile will generally perform a high-level SA&D in the SA&D phase and detailed SA&D during iterative development in the System Implementation phase.

(c) If a final user acceptance test (UAT) is conducted for the whole system at the end of the System Implementation phase, the total time required for conducting the final UAT can be shortened because preliminary testings and reviews have already been done by user representatives during the demonstrations on the completed system functions of the user stories at the end of each timebox. Interim UATs may be conducted for the working software after completing a pre-defined group of timeboxes.

5.2 USER INVOLVEMENT AND EMPOWERMENT

(a) In contrast to the waterfall methodology in which users are mainly involved in the SA&D phase and UAT, the involvement of user representatives in Agile will be constant and evenly distributed throughout the system development period. In each timebox, the user representatives are required to provide detailed requirements, conduct testing and acceptance, and provide comments on the working software delivered during the demonstration at the end of each timebox or during interim UATs if any.

(b) To streamline the implementation process, user representatives need to be empowered to make decisions on matters which do not have much impact on the project scope, quality, cost and schedule, for example, acceptance of the functions and working software and any necessary changes to the requirements and acceptance criteria.

(c) Basically, user representatives who are accustomed to waterfall methodology need time to adapt to the new working practices in adopting Agile although there may not be any increase in the overall workload of user representatives in the IT project.

(d) Resources of user representatives should be estimated and staff turnover should be avoided to provide necessary user support continuously to the project and reduce the risk of project failure.

Hints & Tips 

Table 3 - General Tips for User Involvement and Empowerment

It is very important and beneficial to the project that dedicated BAs can be assigned to represent the business users and make corresponding decisions during the requirement collection workshop and timebox planning meeting.

5.3 PROJECT ORGANISATION

(a) If the project team is not familiar with the Agile practices, the project team may consider to acquire Agile coaching services for providing coaching and mentoring to project team members, user representatives and project managers for adopting the Agile practices. Otherwise, the role of the Agile coach can be taken up by the Internal PM or one of the internal project team members who is familiar with the Agile practices.

(b) BA is responsible for performing the business analysis functions for IT system development projects throughout the SDLC such as analysing business needs and facilitating the elicitation of user requirements from business perspective and effective communication between business user side and IT side. Therefore, BA should participate in each timebox. Since Agile adopts an iterative and incremental approach, the user requirement documents in the SA&D report may only cover high-level rather than detailed requirements. Please refer to the document "*Best Practices for Business Analyst (BPBA)* [1]" for more information about BA practices and user requirements document.

(c)

5.4 PROJECT MANAGEMENT PLAN

(a) Following the normal project management practices, the Internal PM should prepare the project schedule, milestones and scope of procurement in the Project Management Plan (PMP). The project schedule and milestones should base on the information in the FAF in which adoption of Agile should have been taken into account.

(b) In addition, the Internal PM should state clearly in the PMP that Agile practices would be adopted. In this phase, the project schedule needs not be broken down into

detailed development timeboxes which will generally be determined at the start of the System Implementation phase in the SDLC. If required, the PMP may then be revised to include the detailed schedule for timeboxes and releases.

(c) If any training on Agile is recommended for the internal project team members and user representatives or any Agile coaching services are to be procured, the Internal PM may include such information into the project schedule and scope of procurement of the PMP as necessary.

(d) An example of a project schedule with a list of deliverables at different phases is shown in ***Appendix A-4***.

Hints & Tips 

Table 4 - General Tips for Project Management Plan in Agile Projects

Unlike the waterfall approach, more frequent project milestones are preferred in Agile projects to enable more frequent delivery of completed functions to be ready for release, while meeting user's expectations.

5.5 CHANGE MANAGEMENT

(a) As usual, the Internal PM should define the change request management process and ensure all change requests are processed according to the established change request management procedures.

(b) Due to the iterative nature of Agile, user representatives should be empowered to make decision during the development timeboxes on changes of low-level requirements that do not affect the project scope, quality, cost or schedule. The PSC should delegate the necessary decision power to the user representatives. To assure the integrity and quality of the whole system, the low-level requirement change requests approved by user representatives should be recorded properly and reviewed by the PAT periodically.

(c) Changes to high-level requirements should still need to seek endorsement from higher level of authority such as the PAT or PSC according to the PMP. In case the user representatives deemed that a low-level requirement change will affect the project scope, quality, cost, schedule or other areas that are outside their scope of work, the change request should also be escalated to the PAT and PSC for approval.

5.6 QUALITY MANAGEMENT

(a) As usual, the Internal PM should define the quality standard, quality control and assurance activities and the acceptance criteria for major deliverables in the Quality Management Plan. Specific major deliverables for Agile such as user stories and prioritised requirements list (PRL) may also be included as necessary.

(b) Due to the iterative nature of Agile, there will be quality control and assurance activities on the completed system functions delivered during each timebox and each release such as demonstration and acceptance tests.

5.7 COMMUNICATIONS MANAGEMENT

(a) As there is more user involvement in Agile projects, the Internal PM should arrange effective communication means between user representatives and the project team, such as short meetings, release planning meetings, timebox planning meetings, timebox review meetings, etc.

(b) Face-to-face communications in a co-locating working environment is most desirable in Agile to facilitate communications between user representatives and project team members in particular those SAs and programmers who are heavily involved during the System Implementation phase.

(c) Where face to face meeting is not feasible due to practical constraints, alternative means such as video conferencing or even telephone conferencing can be considered in place of the daily stand-up meetings.

5.8 RISK MANAGEMENT

(a) Risk of rework or disputes due to discovery of poor quality of programs, misunderstanding of requirements or changes in requirements etc. at latter part of the System Implementation phase can be mitigated by nature of the Agile approach through regular demonstrations and review meetings conducted at the end of each timebox. The key point is that project team members or user representatives need to take remedial actions as early as the problems are discovered.

(b) Users may have a perception that Agile allows them to change requirements freely which may finally lead to scope creeping, i.e. uncontrolled changes in project scope. Proper training and coaching to users and programmers can minimise such misunderstanding and provide them with proper Agile concepts and practices in handling requirement changes.

(c) The Internal PM should conduct a risk assessment at project planning stage regarding the adoption of Agile, and include them into the risk response plan as necessary. The risk register should also be updated regularly.

6 PROCURING THE SERVICES

6.1 PROCURING AGILE SYSTEM DEVELOPMENT SERVICES

(a) Government IT contracts usually are of fixed-price, fixed-scope and with a fixed project completion date. B/Ds shall comply with the Stores and Procurement Regulations (SPR) in Government in acquiring IT implementation services and should follow the practice of arranging fixed-price and fixed-scope contracts.

(b) In preparing the service brief or project specifications for procuring the implementation services, B/Ds are recommended to state in the brief or specifications that the contractor is required to use Agile approach for development. Besides, the required experience and certifications/qualifications of the potential supplier and its project team members as well as the required core Agile practices such as incremental and iterative delivery of working software using timeboxes, demonstration of completed functions at the timebox review meetings, etc. should be included in the brief or specifications as appropriate.

(c) Project and payment milestones can be defined by the completion of different stages similar to that for the waterfall approach. It is suggested to consider having more than one project milestone or payment milestone if a lengthy program development stage is anticipated.

(d) In Agile, project and payment milestones should be fitted to one or multiple release(s) in which the working software will be progressively developed until the entire system is delivered. Additional interim UATs may be added before each release to verify whether the requirements are completed and acceptable.

Hints & Tips

Table 5 - General Tips for Procuring Agile System Development Services

(a) When specifying the required experience and certifications/qualifications, the Internal PM should consider the availability of such required resources from vendors and IT professionals in the market.
(b) If project team decides to use any tools in the project, they are suggested to i) include the cost, if any, of the tool into the project budget, and ii) explicitly specify the requirements of using the tools, and to what extent the tools should be used, in the tender specification.

6.2 PROCURING AGILE COACHING SERVICES (OPTIONAL)

(a) If either the Internal PM or one of the internal project team members is familiar with Agile practices, he or she may take up the role of Agile coach. Otherwise, Agile coaching services are required to facilitate the project team, user representatives and programmers in adopting the Agile approach for an IT project.

(b) It is essential that the Agile coach has experience in both Agile system development as well as coaching on Agile adoption in order to facilitate the collaboration between project team members and user representatives in adopting the Agile practices. A list of sample professional requirements for the Agile Coach is provided at ***Appendix A-5***.

PART III - USING AGILE FOR SOFTWARE DEVELOPMENT

7 SYSTEM ANALYSIS AND DESIGN

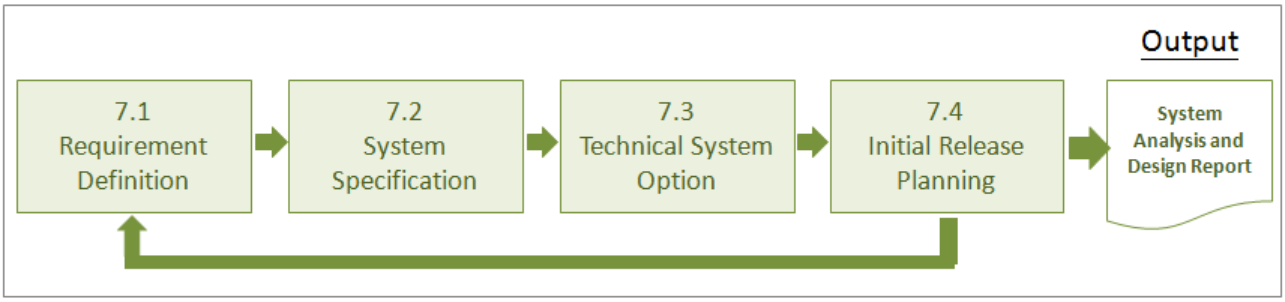

Figure 6 - System Analysis and Design in Agile

(a) Agile practices focus on system implementation and do not come with specific practices for the SA&D stage. The SA&D practices stated in the "*Effective Systems Analysis & Design Guide (ESADG)* [2]" and the BA practices stated in the *BPBA* [1] can be suitably applied to Agile projects.

(b) Deliverables of the SA&D phase are expected to be at high-level covering the core functions and critical system components to the extent as is sufficient to start system development. Details will be worked out and refined iteratively in the System Implementation phase.

7.1 REQUIREMENT DEFINITION

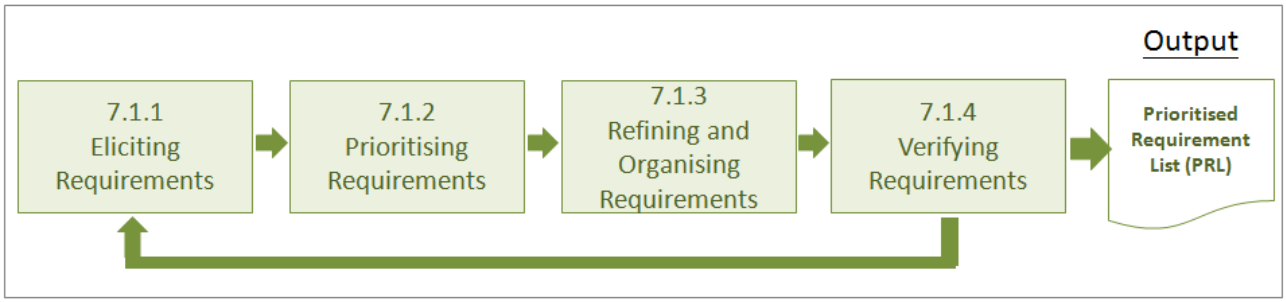

Figure 7 - Requirement Definition

7.1.1 ELICITING REQUIREMENTS

(a) In Agile project, high-level requirements are documented in the form of "user stories" which are written by users in layman's terms. BA will collaborate with users and assist them in preparing the user stories as necessary. Last but not least, any technical requirements will be collected by SAs and, if applicable, SArch as usual.

(b) In some circumstances, low-level requirements of some functional areas may also be provided by users during the SA&D phase. If the information is sufficient, detailed

analysis and design may also be conducted for those functional areas as well. Otherwise, such information can be recorded and retrieved for further analysis during the System Implementation phase.

(c) The content of a user story should include a unique user story ID, user story description, acceptance criteria for the requirements stated in the user story, an assigned business value, and assumptions, if any. Some examples of user stories are shown at ***Appendix A-6***.

7.1.2 PRIORITISING REQUIREMENTS

(a) User should assign a priority, i.e. ranking of relative importance, of development to each functional and non-functional requirement.

(b) MoSCoW (**M**ust, **S**hould, **C**ould and **W**on't) is a technique commonly used in Agile project that helps users prioritise each requirement based on its relative importance. Requirements which have not yet been confirmed to be implemented should not be assigned with must-have or should-have priorities, but may be considered for could-have priorities.

(c) Besides the MoSCoW prioritisation technique, user stories may also be prioritised by assigning a number say 1 to 5 (from the lowest to the highest) to indicate the relative business value of the user stories. A user story with the lowest number will be assigned with the lowest priority of development, or even be withdrawn from the list of requirements. This helps ensure that the proposed system can be completed and delivered on time and within budget based on current available resources.

7.1.3 REFINING AND ORGANISING REQUIREMENTS

BA should assist users in refining and organising the user stories by using appropriate modelling techniques such as process modelling for future business processes, functional decomposition, scenarios and use cases, flow diagram, activity diagram, sequence diagram, structured walkthrough, etc. The goal is to ensure that the user stories delivered are well-organised and grouped, and are clearly written and understood.

7.1.4 VERIFYING REQUIREMENTS

(a) Before proceeding to the system design phase, BA should help verify and finalise the prioritised user stories and seek user confirmation and acceptance. A prioritised requirement list consisting of user stories and other non-functional requirements will be delivered. If necessary, BA may go through several iterations of elicitation, prioritisation, refinement and reorganisation and verification of requirements. The verified requirements will then be submitted to seek users' confirmation and acceptance.

(b) A list of technical requirements will also be produced by the SAs after verified with the technical staff.

Hints & Tips

Table 6 - General Tips for Requirement Definition

(a) User requirement is easier to collect if project team and business users can work closely together.
(b) Close collaboration between project team and business users could also help the project team better understand the user expectation.

7.2 SYSTEM SPECIFICATION

(a) Following normal SA&D practices, SA should work together with the SArch, if applicable, to develop the System Specification which includes Functional Specification, Architecture Design and System Design. But the system specification process can be flexibly adopted and simplified according to the project needs and the key consideration factors stipulated in the *ESADG* [2].

(b) The following paragraphs highlight a few points to be noted when performing the system specification for Agile projects. Details about the process can be found in the corresponding sections of the *ESADG* [2].

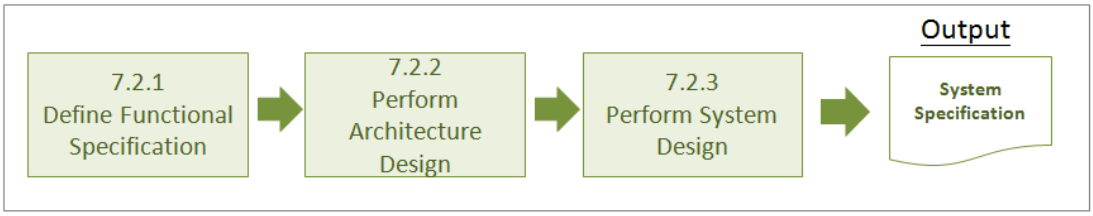

Figure 8 - System Specification

7.2.1 DEFINE FUNCTIONAL SPECIFICATION

PRL could be used as Functional Specification directly. If considered appropriate to facilitate system implementation, Required System Overview should be defined to provide a functional overview of the targeted system and Function Definition should be included to detail the expected behaviour of the system for the user stories.

7.2.2 PERFORM ARCHITECTURE DESIGN

The SA should follow normal SA&D practices in doing the Architecture Design. But the various architectures such as Application Architecture, Security Architecture, Integration Architecture and Data Store Architecture need only be defined at higher level, with the focus on core functions and components.

7.2.3 PERFORM SYSTEM DESIGN

(a) System design will be conducted to produce a high-level Logical Data Model and User Experience based on the business rules and user stories defined.

(b) The Logical Data Model needs only be "just enough", or just a subset of the high-level diagram to show only the core data entities that have key business meanings and the relationships between those core data entities. It is not necessary to design every single detail in the data models, or to completely describe all details of the system. The data models are expected to gradually evolve during the System Implementation phase.

(c) User Experience will be defined using prototyping techniques mainly for core or major functions.

(d) The Physical Data Model covering detailed design is not required to be defined in the SA&D phase.

7.3 TECHNICAL SYSTEM OPTION

(a) Technical System Option (TSO) is generally independent of the software development methodology. The project team should perform the process as usual.

(b) Implementation of the technical architecture is normally commenced at timebox 0, i.e. before the start of the first development timebox. In Agile, the planning and scheduling of individual development timebox will be conducted in next Release Planning stage.

7.4 INITIAL RELEASE PLANNING

(a) An initial release planning is to identify the development tasks required for implementing each user story, and then split and prioritise the user stories into one or more release(s) of working software for rollout.

(b) A release plan represents a schedule for delivering one or more release(s) to users, i.e. the target functions of working software to be delivered to users by different deadlines. The functions will be delivered upon completion of a planned number of timeboxes associated with the selected user stories.

(c) The project team and users should also refer to the high-level Functional Definition in the SA&D report when grouping of user stories and planning for the functions to be released.

(d) In normal practice, release planning refers to planning for design, coding and testing of the application software based on user stories. Non-functional and technical requirements are often related to technical solutions and normally incorporated into the implementation of the technical system architecture, and hence will not be addressed individually in the release planning.

(e) Agile has no specific practices for guiding the implementation of the technical system architecture. Nevertheless, the project team should reserve the initial timebox, i.e. timebox 0, to plan for and develop the basic architectural models and development platform.

(f) The release planning process is shown in Figure 9 below.

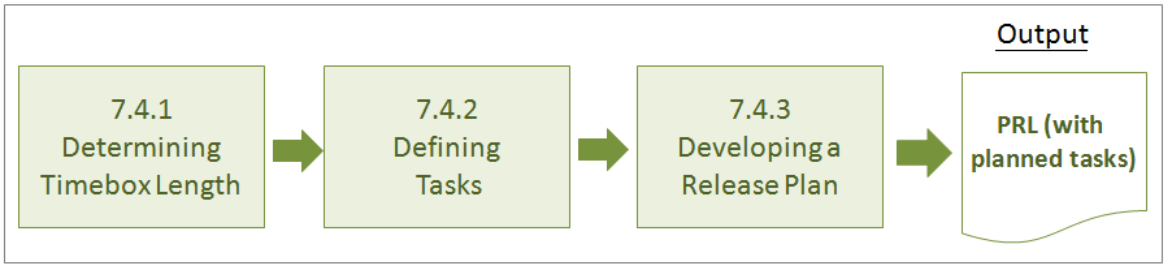

Figure 9 - Release Planning

7.4.1 DETERMINING TIMEBOX LENGTH

(a) The project team should first determine a fixed time interval for iteration i.e. the length of each development timebox. The time interval normally ranges from 1 week to 4 weeks, but it is common to use a 2-week time interval.

(b) An appropriate timebox length should be set to ensure that sufficient time will be allocated to each timebox for completing the activities including planning, design, coding, testing, demonstration and retrospective meeting. Typically, the shorter the length of a timebox, the faster the programmers will receive feedback from users.

(c) In addition, the total number of user stories to be implemented, the availability of staff resources as well as the project budget and schedule should also be considered when determining the timebox length.

7.4.2 DEFINING TASKS

(a) SAs and programmers are required to roughly define the development tasks for implementing each user story. For example, if a user wants to online enquire his/her own accounts' balances and transactions, the development tasks may include a task for screen layout design, a task for user account searching, a task for extracting all transactions from the selected user account, and a task for coding the main program. The task list will be enhanced with more details at the beginning of each individual timebox in the System Implementation phase.

(b) At this stage, the PRL for user stories should contain the parent user story ID, tasks descriptions, status, creation date and last update date.

7.4.3 DEVELOPING A RELEASE PLAN

7.4.3.1 WHAT IS A RELEASE PLAN?

(a) In Agile, there is often no separate document named as a release plan. Rather, the existing user stories in the PRL are updated with release information and the updated PRL is functionally equivalent to a release plan. At this stage, the PRL for user stories should contain the parent user story ID, tasks descriptions, estimated man-days/story points required for each task, the timebox assigned to each user story, the release number of each timebox, status, creation date and last update date.

(b) The information produced from the release planning will serve as the basis for subsequent timebox planning. At the start of each timebox, programmers have to conduct timebox planning to come up with a more accurate estimation of the resources required, and to confirm with the users on the user stories and detailed tasks (together with their acceptance criteria) to be implemented in that timebox.

(c) A sample model showing major elements in a release plan is shown below.

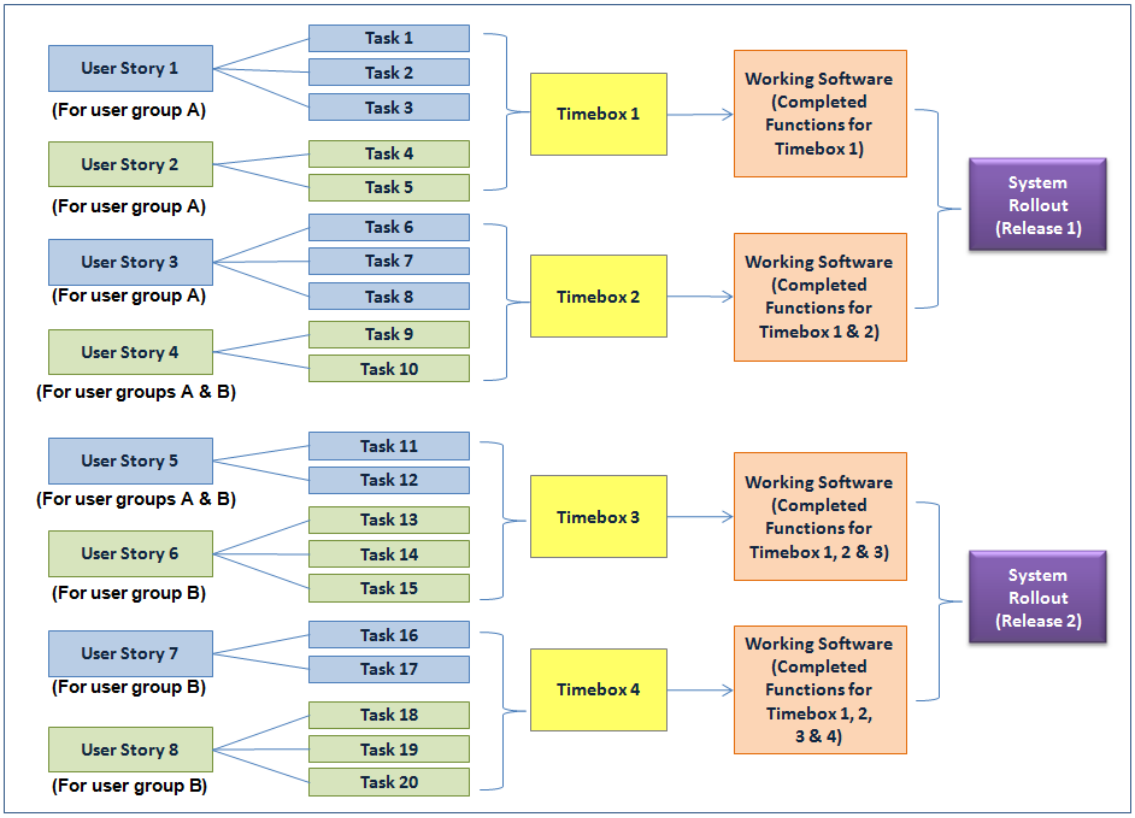

Figure 10 - A Sample Model of a Release Plan

7.4.3.2 INITIAL MEETING

An initial meeting will be held between the project team members and the user representatives to start the release planning. The goal is to estimate roughly what features/functions will be delivered by a target release date.

7.4.3.3 RELEASE PLANNING ACTIVITIES

(a) The release planning process will generally consist of the following activities:

i) to walkthrough the user stories to ensure project team members especially programmers understand what the user stories mean;

ii) to roughly estimate the relative staff effort required for completing each task;

iii) to determine which user stories are to be implemented in each timebox; and

iv) to determine which timeboxes are grouped for a release.

(b) Before the release planning meeting, users are required to review the user stories and update the validation criteria such as validation rules, detailed functions, steps for checking, etc. During the walkthrough of user stories, users should provide supplementary information about the user stories that are essential for the SAs or programmers to plan the development tasks.

(c) The acceptance criteria of user stories sometimes can only be completely defined at the beginning of each individual timebox planning. It is because there may be changes in user stories, or more time is required by users to think thoroughly about the acceptance criteria. One of the benefits of using Agile is to allow program development to start earlier without waiting for all the detailed requirements to be well defined.

(d) SAs and programmers should roughly estimate the relative staff effort required for each task. This can be estimated using man-days or story points. Story points are relative numbers normally ranging from 10 to 100. Initially, the project team should assign a story point value as the estimated effort for completing one timebox. The project team will review the value at the end of the first timebox and adjust it if necessary for the coming timeboxes. For example, the project team may estimate that 40 story points will be required for completing a user story for enquiry of an account balance while relatively 20 story points will be required for completing a user story for printing of an account statement.

(e) During release planning, user stories will be grouped into different timeboxes for implementation. A complex user story can be split and broken down into simpler and smaller user stories as necessary. Usually, user stories will not be assigned to the last timebox and/or the second last timebox for implementation. Such timebox(es) is/are reserved as time buffer for the project for wrapping-up, or for completing any outstanding requirements or issues, or additional development tasks.

(f) When assigning user stories to the timeboxes, the project team should consider the following main characteristics of user stories:

i) business values or priorities;

ii) required staff effort for implementation;

iii) dependencies on other user stories; and

iv) availability of related stakeholders for attending the timebox meetings.

7.4.3.4 GROUPING OF TIMEBOXES INTO RELEASE

After assigning user stories into the timeboxes, the project team starts grouping timeboxes into one or more release(s) according to the priorities and dependencies of user stories and the relevance of the delivered functions.

7.4.3.5 DETERMINING THE TIME PERIOD FOR EACH RELEASE

(a) If rollout of the system by phases or releases is feasible and preferable, the project team should determine the time required for each release. Each release will contain a number of completed timeboxes. The target functions delivered in a release should be a working software integrated with the functions delivered in the previous release.

(b) When development is in progress, there will probably be changes in details of the user stories or other external environment, for example, interfaces with other systems or policy/regulations, that may cause necessary adjustments in the priority of user stories in a release.

7.4.3.6 IMPLEMENTATION PLAN FOR TECHNICAL SYSTEM ARCHITECTURE

A separate, detailed implementation plan for building up the technical system architecture is required. Such implementation plan should be aligned with the release schedule to ensure that the necessary system technical architecture is developed to support the development and operation of the application software accordingly.

7.4.3.7 TOOLS

(a) There are many online tools in the market to facilitate the project manager to monitor the progress and maintain the PRL of the Agile project. Most of the online tools support the functions of release planning, timebox planning, reporting, test management, project progress tracking, etc

(b) Besides using online tools, the PRL can also be created by using a spreadsheet consisting of multiple worksheets. Each worksheet will have some pre-defined calculation rules for generating diagrams and charts, and for keeping track of the project status. A sample spreadsheet for Agile project management is provided at ***Appendix A-7***.

Hints & Tips 

Table 7 - General Tips for Release Planning

(a) Sometimes, user stories with higher priorities may not be done first during implementation due to other factors such as additional estimated effort, inter-dependencies of the user stories, external dependencies and availability of stakeholders for the timebox meetings.

(b) It is more effective if user stories related to the same group of stakeholders can be merged together under the same group of timeboxes for development. This facilitates scheduling of meetings and collaboration with users.

(c) Effort estimation should be precise, and should cover the detail as far as practicable, instead of applying the same amount of relative effort to most user stories.

(d) Story point should be relative rather than absolute value. The story point is mainly used for scheduling of tasks, estimating the man-effort and measuring the progress. It should not be affected by the delivery schedule.

(e) User stories should be merged into one user story if they are really interpreted as same function.

(f) Agile emphasises "just-enough" documentation. Project teams should decide the level of system documentation based on project needs, bearing in mind the idea of "just-enough". Project teams can also make reference to the *ESADG* [2] to flexibly prepare the essential SA&D documents.

(g) Although the working software has been accepted by users during the previous timebox, the entire software still needs to pass a UAT before rollout. The project schedule may also be updated to incorporate the details of the release planning as necessary.

8 SYSTEM IMPLEMENTATION

This section introduces the processes and activities as well as the output to be produced in the System Implementation phase. The processes and activities are generally presented according to the sequence of flow shown in Figure 3. The activities shown from Section 8.2 Timebox Planning to Section 8.7 Retrospective Meeting are iterative in nature, and should be repeated and performed in every timebox.

8.1 DEVELOP SYSTEM ARCHITECTURE

Before starting the System Implementation phase, the project team should set up the initial architecture models and the development environment as generally applicable to SDLC. This should be done at timebox 0. The project team should install and configure the related servers, databases, network connections, etc. according to the SA&D report developed in the SA&D phase.

8.2 TIMEBOX PLANNING

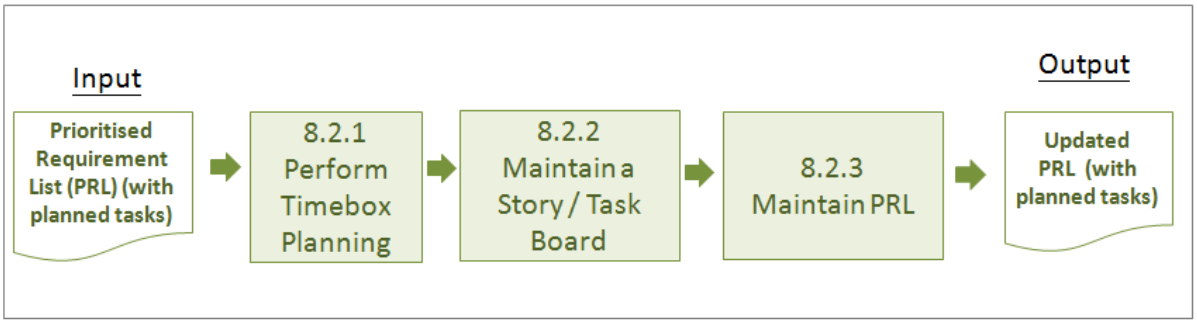

Figure 11 - Timebox Planning

8.2.1 PERFORM TIMEBOX PLANNING

(a) At the start of each development timebox, a 2-4 hour meeting will be conducted to plan for implementation of user stories in each timebox. Project team members, user representative(s), programmers and the Agile coach (if any) will attend the timebox planning meeting.

(b) In the meeting, programmers and other project team members should be motivated and encouraged to select user stories and the associated tasks that they have the capabilities and resources to perform, according to the prioritised user stories in the current timebox and the release schedule. They can discuss how to share the workload amongst themselves. If required, SAs or the Internal PM may also assign tasks to programmers especially for mission-critical functions/features where more experienced programmers are required, or for new programmers who may not know the estimated effort required.

(c) After tasks have been assigned, the programmers should clarify the acceptance criteria of user stories with the user representatives or other involved project team

members. The user stories and tasks can be further revised, re-grouped or broken down into smaller stories and smaller tasks wherever necessary.

(d) The resources required for implementing each task is re-estimated using man-days or story points as mentioned in section 7.4.3. If the total estimated resources required are less than the total available staff resources in the current timebox, planned tasks in the next timebox can be advanced to the current timebox.

(e) The project team should complete those assigned user stories within the period of time defined for the timebox, and produce the working software for user review at the end of the timebox.

(f) In each timebox, some of the available resources, e.g. 20%, should be reserved as time buffer to cater for tasks/events such as administrative tasks, sick leave or vacation leave of programmers/project team members, and important outstanding issues or problems brought forward from the preceding timebox as necessary.

(g) Poker planning technique can be used for estimating the effort for each user story. A brief description of the technique is given below:

i) Each programmer and/or SA is given a set of playing cards bearing numerical values, e.g. 1, 5, 10, 15, 20, 50, 100, etc., as appropriate for points estimation of effort for a user story;

ii) Individual user stories are then presented and briefly explained by users;

iii) After discussion, each programmer chooses a numbered card that represents his/her estimate of how much work is involved in the story;

iv) The numbered cards are kept private until each participant has chosen a card, all estimates are revealed;

v) The two (or more) card players who give the highest and lowest estimates are required to justify their reasoning; and

vi) After brief discussion, the team may seek convergence toward a consensus of estimate by playing one or more further rounds as necessary.

Hints & Tips 

Table 8 - General Tips for Timebox Planning

(a) The detailed requirements should be clarified at the timebox planning meeting instead of the demonstration session at the end of each timebox, which is too late.
(b) Any new user story (e.g. because of new user requirement) arising during the System Implementation phase can be added to the PRL during timebox planning of the next timebox. Other outstanding user stories in the PRL can be re-prioritised and tasks can be re-scheduled as appropriate.

(c) However, if the estimated effort for implementing the new user story is so significant that the time and effort cannot be absorbed by the available buffer (i.e. the last timebox or the second last timebox), then some of the user stories or tasks with lower priorities may need to be swapped out and replaced by the new user story.

(d) If the planned tasks are too large to complete in one timebox, project team should consider splitting the planned tasks into 2 or more timeboxes.

8.2.2 MAINTAIN A STORY/TASK BOARD

(a) The status of the user stories and tasks are recorded using story boards and task boards respectively and updated continuously during development. Each board is divided into multiple columns with each column representing one of the following status:

i) To do

ii) In progress

iii) Tested

iv) Done

v) Removed

(b) For the project team which adopts a manual story/task boarding process, programmers will write down the information related to each task such as task ID, task description, estimated effort, timebox ID, etc. on the post-it sticker and then stick it into the column of the task board representing the latest status and progress of assigned tasks. A similar approach for recording the progress of user stories is used for the story board.

(c) The Internal PM or project team members will review the project progress through this manual process.

(d) Other than physical story/task board, an electronic version board created by software tools or manual spreadsheets can also serve the same purpose.

8.2.3 MAINTAIN PRIORITISED REQUIREMENT LIST

(a) The PRL should be updated during the timebox period for any changes to user stories, sequence of tasks, etc. If necessary, the project manager should review the project progress, tasks dependency and performance of programmers and then re-prioritise the user stories and the sequence accordingly.

(b) In case there are any changes to the user stories and/or acceptance criteria, the PRL and the release plan should be updated as appropriate. The Change Management Plan of the project should be followed for seeking of approval. It is essential that the change control process allows minor changes to be handled efficiently for Agile projects.

(c) Those low priority requirements should be assigned with “Could-have” priority and the corresponding tasks should be placed after those “Must-have” and “Should-have” tasks. If the revised schedule finally does not have enough time to accommodate all

"could-have" tasks, then some of those tasks may need to be withdrawn from the requirement list.

Hints & Tips

Table 9 - General Tips for Maintaining Prioritised Requirement List

(a) Although Agile has the strength of entertaining requirement changes, it is still beneficial to the project as a whole if the project team and user representatives can keep the changes to a minimum to avoid scope creeping.
(b) Some developers may have concern about the reasonableness and acceptable numbers of requirement changes raised by users. Under such situation, the BA should take up the role to liaise between the users and the programmers in relation to the requirement changes. Reservation of one or few more spare timeboxes at the release planning stage can provide flexibility in entertaining requirement changes.

8.3 REQUIREMENT ELICITATION

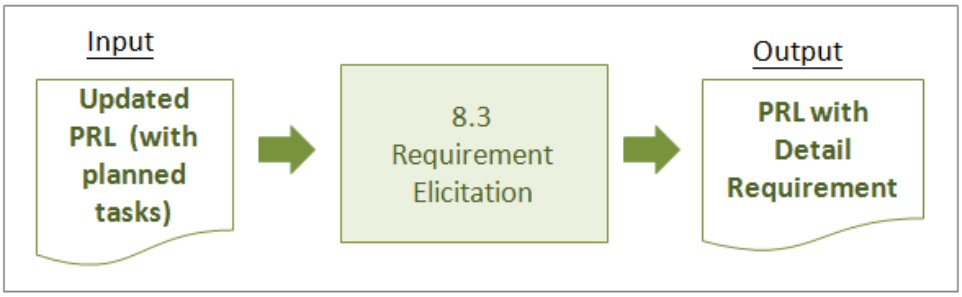

Figure 12 - Requirement Elicitation

(a) To collect the in-depth user requirements after timebox planning, the BA and the project team should further discuss with users on the detailed requirements as evolving and updated in the acceptance criteria of the user stories.

(b) The project team can develop prototypes to show the sample screen and report layouts to seek confirmation from users and finalise the detailed requirements before starting the coding.

(c) During the iterative process in the System Implementation phase, some of the detailed requirements may change over time. BA should periodically review the user stories to ensure that the requirements are aligned with the latest business needs.

8.4 DETAILED SYSTEM DESIGN

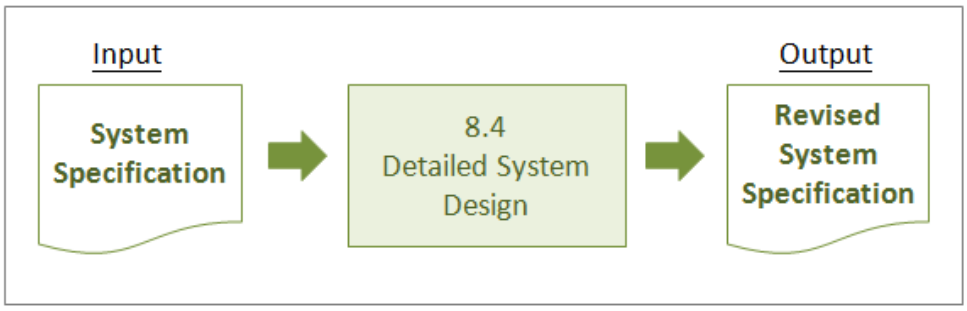

Figure 13 - Detailed System Design

(a) After collecting & confirming the detailed requirements from users, the project team will start to perform the detailed system design including the data and application design. The Logical Data Model, Physical Data Model, Application Design and User Experience Design specified in the SA&D report will be worked out or enhanced progressively during implementation.

(b) Since the detailed system design may be refined during each timebox period, either the corresponding sections of the whole SA&D document need to be updated or changes be recorded as part of the system documentation to be delivered at the end of the implementation.

8.5 CODING, DEVELOPMENT & TESTING

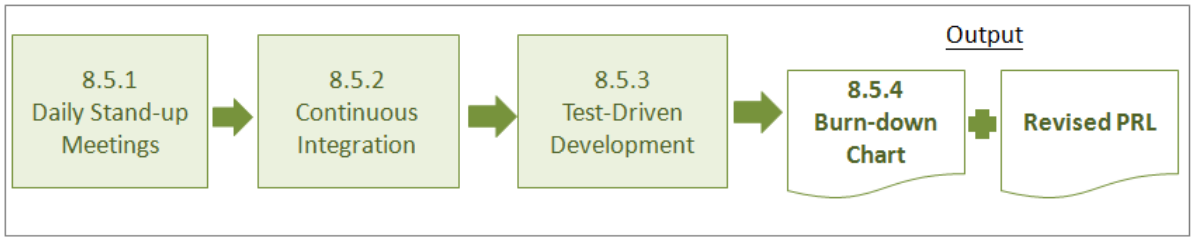

Figure 14 - Coding, Development & Testing

8.5.1 DAILY STAND-UP MEETINGS

(a) A short daily stand-up meeting of not more than 15 minutes will be held among project team members and programmers for a quick status update to team members in particular the project manager. All team members are encouraged to attend the stand-up meeting. The meeting should not be cancelled even if someone cannot attend.

(b) The purpose of standing up at the meeting is to keep the meeting short and concise.

(c) The daily stand-up meeting focuses on the following 3 questions:

i) What user stories or tasks were finished yesterday?

ii) What user stories or tasks are you working on today?

iii) Is anything blocking you from making progress?

(d) Project team members should report the progress of the tasks which were assigned at the PRL and planned to be done at the previous day's daily stand-up meeting.

(e) Daily stand-up meeting is an effective communication means for the project team and users to understand the project progress, and any technical issues such that remedial actions can be taken timely. It facilitates project management with frequent updates on the project progress. The programmers can discuss with users on those reported blocking tasks immediately after the meeting.

(f) As an alternative to the face-to-face daily stand-up meeting, telephone conference and video conference can also be conducted which are also effective for daily update among project team members.

(g) Daily stand-up meeting can speed up the system development work by giving pressure to the programmers to complete the scheduled tasks every day.

(h) In case there are multiple incomplete tasks in one or more timeboxes, the Internal PM should analyse the situation and consider escalating it to the PAT or PSC to alert them of any potential delay in project milestones or identified project issues as soon as possible. Some suggested possible solutions for such case is:

- Re-prioritising requirements and dropping other "could-have" requirements;
- Re-prioritising some requirements to the last timebox or the second last timebox, i.e. using the buffered resources;
- Requiring the development team to work overtime to complete the outstanding tasks; or
- Adding new timebox(es).

8.5.2 CONTINUOUS INTEGRATION

(a) The practice of continuous integration in Agile projects allows the project team and users to perform unit and/or integrated testing frequently during each timebox, and thus enables timely feedback to users. This helps identify program defects earlier, and allows the project team to fix them promptly.

(b) Up-front system architecture design and set up should be performed at Timebox 0 (i.e. before coding & development). Project team should enhance the system architecture design and progressively develop the system architecture during the System Implementation phase. The system architecture will be gradually evolved and built to provide an integrated platform supporting the delivery of target functions at each timebox. Detailed tasks for developing the system architecture should be performed during the System Implementation phase.

(c) To conduct continuous integration, the first step is to set up an automated environment to build the software and perform the validation process automatically.

The automated environment enables the entire system to be automatically re-built when there are any changes in the programs. It can identify areas which are getting wrong, and expose the failure by automated testing.

(d) Besides, programmers are suggested to integrate the completed codes frequently and regularly under the build server environment through the automated build program. This allows the project team to detect and locate any correlated problems early and easily and reduce the effort to redo.

(e) As compared with the traditional big-bang integration approach, integrating and building the software frequently helps reduce the impact caused by detected flaws, and thus minimise the project risk.

8.5.3 TEST-DRIVEN DEVELOPMENT

(a) In Agile, program development is usually done using a test-driven development (TDD) approach in which the programmers write the test cases before writing the programs, and automated testing tools are often used to perform the testing.

(b) TDD is a common approach used in Agile projects where the development of code is driven by first writing an automated test case, followed by writing the code to fulfil the test, and then refactoring (re-writing code without changing the functionality).

(c) TDD ensures that conditions covered by the test cases are tested, and makes it easy for re-testing after any program changes. However, programmers need some training and practice to master the techniques of TDD, and considerable effort is required for writing test cases. Resources estimation for TDD should be included during timebox planning.

(d) TDD forces the project team to understand the requirements clearly and consider the software interface carefully before starting the coding process.

(e) While TDD provides the above benefits, a reasonable balance should be struck between the coverage of test cases and the effort required to write the test cases particularly for small projects where the requirements are relatively simple. Regardless of whether TDD is used, it is the project team's responsibility to thoroughly test the programs to make ready the working software for user acceptance.

8.5.4 BURN DOWN CHART

(a) Based on the amount of work completed and the time and resources used during previous timeboxes, the velocity of the project team can be determined. Velocity refers to the total number of tasks completed in a period of time (i.e. one timebox). It allows the project team to estimate the time and resources required for the remaining user stories and tasks. The work plan may need to be adjusted as necessary to include additional resources or drop "Should-have" or "Could-have" requirements to allow the system to be delivered as scheduled.

(b) A burn down chart is a commonly used tool in Agile projects to graphically show the amount of work left. It is useful for displaying the progress towards completion and indicating when all the planned work will be completed. It shows the number of estimated story points and the actual story points done at every timebox. Hence the project team can easily see from the chart the tasks left and the performance of the project team.

(c) The X-axis of the burn down chart usually shows the timeline (i.e. timebox) whereas the Y-axis shows the story points to be completed for the project. Together, the time estimates for the work remaining will be presented. Two sample burn down charts at Timebox 0 and Timebox 3 are shown below respectively.

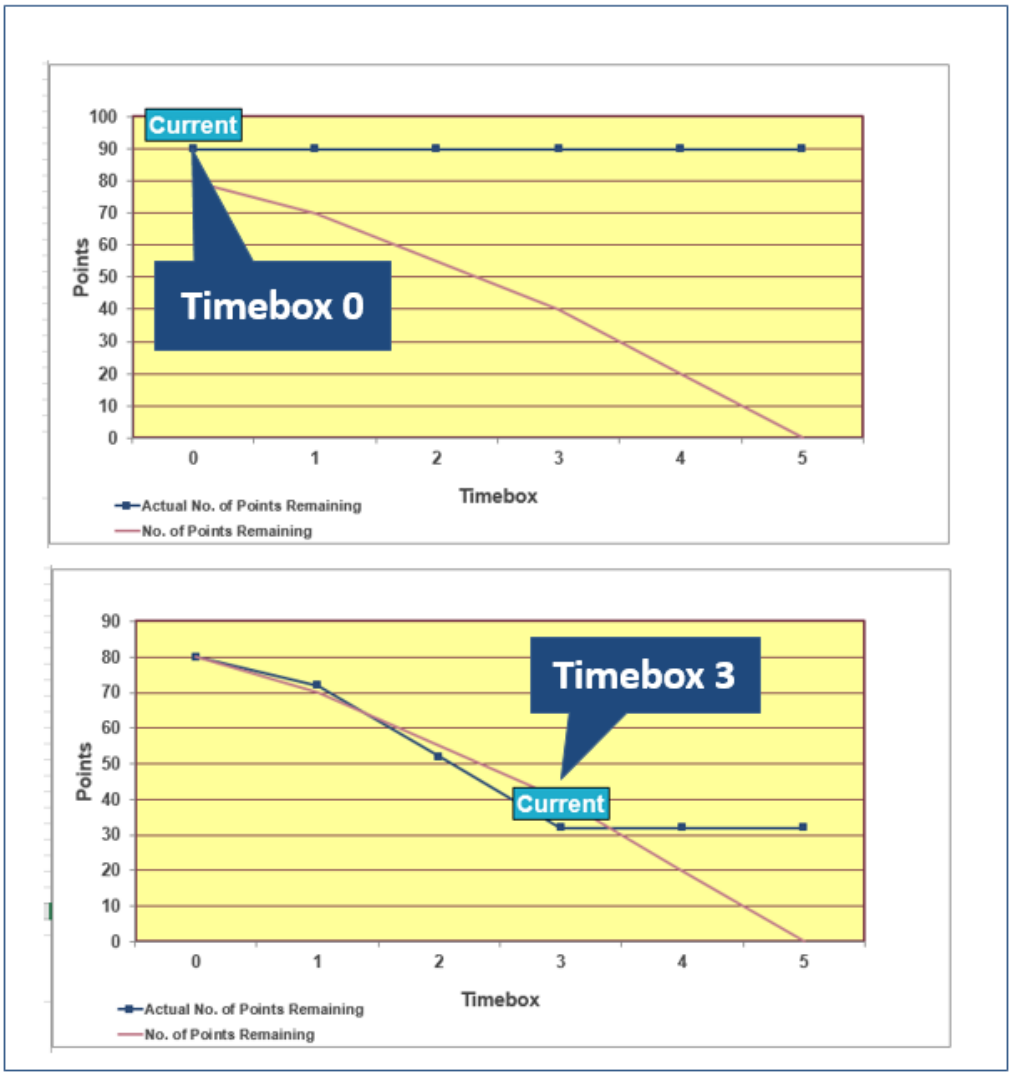

Figure 15 - Sample Burn Down Charts at Timebox 0 and Timebox 3

8.6 DEMONSTRATION

(a) At the end of each timebox, a demonstration meeting will be held between the programmers and the users. In the meeting, the programmers will demonstrate to users the working software as completed. Users will then review the working

software to determine whether all of the user stories are implemented as planned, and all the acceptance criteria are met. The working software will be accepted based on the user stories, acceptance criteria, and definition of "Done" agreed at the beginning of a timebox.

(b) The project team should ensure that all planned tasks are completed and a well-prepared demonstration is conducted with essential testing scenarios for user review.

(c) Demonstration helps uncover program defects as well as necessary improvements or changes to requirements. The programmers will collect feedback from user representatives and make any necessary changes to the working software during the next or subsequent timeboxes.

(d) After the demonstration, users can review the progress, programmers' performance and other dependencies to determine if any re-prioritisation of the assigned tasks and planned user stories is needed in the next timebox.

Hints & Tips

Table 10 - General Tips for Demonstration

(a) Demonstrating the prototype / screen layout before going through the acceptance criteria greatly help end users to have a preview and clearer picture of the workflow, and can avoid misunderstanding of the actual needs of users.
(b) Use of simple English instead of technical terms in the workflow demonstration can avoid misunderstanding between programmers and end users.
(c) The project team should well prepare the acceptance criteria for testing during the demonstration session.
(d) The project team should use as many testing scenarios as necessary for demonstration.

8.7 RETROSPECTIVE MEETING

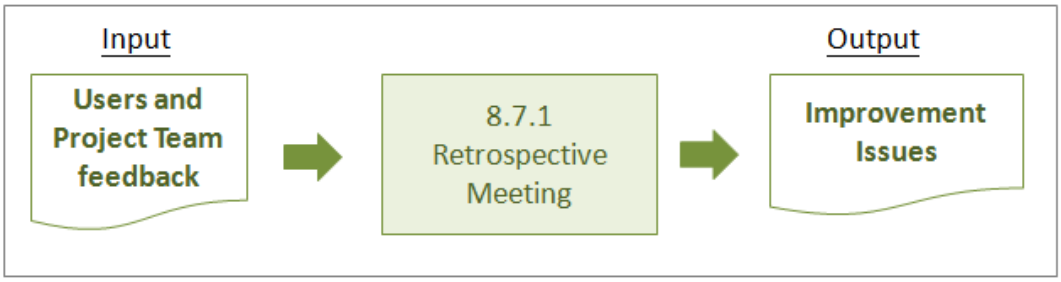

Figure 16 - Retrospective Meeting

(a) The retrospective meeting is normally conducted at the end of each timebox. The objective of the meeting is for users and the project team to provide feedback and to review the process of each timebox so as to enhance productivity and efficiency for the following timeboxes.

(b) The retrospective meeting helps identify any possible improvement areas for the programmers to work in a more effective way. At the retrospective session, feedback on the following questions will be collected from the project team and discussed among team members in order to improve the quality and performance of the project team.

i) What did we do well?

ii) What didn't go so well?

iii) What should we do differently next time?

8.8 USER ACCEPTANCE TEST (UAT)

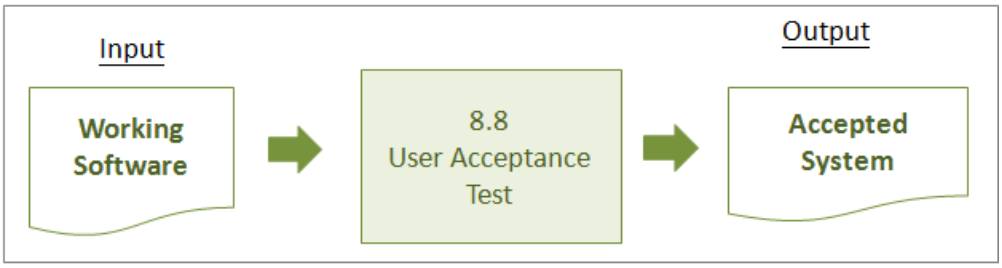

Figure 17 - User Acceptance Test (UAT)

(a) To allow user testing to be conducted frequently and more effectively to ensure the accuracy and quality of work, the project team is recommended to conduct an interim or a partial UAT for selected timeboxes (e.g. 4-6 timeboxes) before the final UAT. It allows the user to test the system right after the development and provide feedback on small pieces of work. It also enables the project team to fix the reported issues immediately instead of leaving them to the final UAT.

(b) During the interim/partial UAT, BA should assist users in preparing the test plan with test cases and test data while the project team should set up the test scripts and the testing environment for users to test the program. BA should ask users to validate the test cases as early as possible to avoid misunderstanding of user requirements that may affect the schedule of demonstration.

(c) After the interim/partial UAT, users should provide comments on the implemented user stories and the project team can then fix the issues reported by users and develop the testing summary report.

(d) After all timeboxes are completed and all user stories are implemented, the final working software (i.e. the entire system) will be made ready for users to perform the final UAT.

(e) Through frequent and repeated testing at each timebox, a better quality of software can be expected prior to the final UAT as compared to non-Agile projects. It also gives confidence to the project team and users that the working software should be capable of meeting actual user needs. The final UAT, which will be conducted at the

end of all timeboxes, will likely require less time than that using the traditional waterfall approach.

Hints & Tips

Table 11 - General Tips for User Acceptance Test

To facilitate the UAT for timeboxes, the project team should reserve sufficient time for preparing the test scripts and the test environment and also fixing the reported issues after each UAT. The project team may thus consider reserving a break period (normally 2-3 weeks) before starting the next timebox to fix all UAT issues which may affect the schedule of the subsequent development timeboxes.

8.9 SYSTEM ROLLOUT

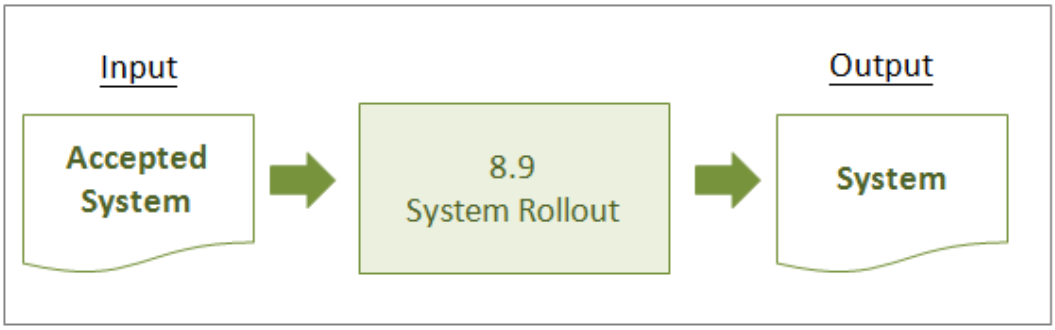

Figure 18 - System Rollout

(a) The project team is recommended to prepare the application operation manual and the application user manual as in the usual case of waterfall approach. On the other hand, other implementation phase documentations such as the system manual, program manual, data manual and computer operating procedures manual may not be necessary in Agile. Nevertheless, the project team may prepare any of these documents in consideration of the needs for on-going maintenance support, etc. as appropriate.

(b) Prior to the system or release rollout, the project team should ensure all the user requirements are completed and meet the business needs and users expectation. Upon acceptance of the system or release by users, the project team can prepare for the rollout of the system or current release as planned.

8.10 POST-IMPLEMENTATION REVIEW

Similar to traditional development approach, project teams should work together with BA and other project stakeholders to conduct the post-implementation review after system rollout for a certain period of time. Feedback collected during the retrospective reviews at the end of every timebox can also be consolidated into the post-implementation report for reference by other similar projects in future.

REFERENCE

1. “Best Practices for Business Analyst” can be found at http://www.ogcio.gov.hk/en/infrastructure/methodology/system_development/best_practices_for_ba.htm.

2. “Effective Systems Analysis and Design Guide” can be found at http://www.ogcio.gov.hk/en/infrastructure/methodology/system_development/effective_guide.htm.

GLOSSARY

Table 12 - Glossary to facilitate the consistency of terms

Term	Definition
Agile	Agile is an iterative and incremental software development method, where requirements and solutions evolve through close collaboration between users and project teams. It promotes evolutionary development and delivery using iterative approach, and also encourages rapid and flexible responses to changing requirements.
Agile Coach	A person who provides coaching/mentoring services to assist the project team members, user representatives and project managers in adopting the Agile practices.
Business Analyst (BA)	BA refers to any person(s) who is(are) responsible for performing the business analysis functions for IT system development projects.
Burn down Chart	It is a graph that shows the quantity of work remaining in terms of effort required for every timebox during the project period. The estimated effort planned and the actual effort used will be shown on the graph for analysis.
Change Management	A practice to control any additions, deletions, and/or modifications to the scope and project plan.
Change Request	A formal request to change the scope, design, method, or other planned aspects of a project, usually including estimates of the effect to the project cost and schedule. It may arise through changes in the business or issues in the project. The document should be logged, assessed, and approved before a change to the project is made.
Communications Management Plan	The document that describes the communications needs and expectations for the project; how and in what format information will be communicated; when and where communication will be made; and who is responsible for providing each type of communication. The communication management plan is contained in, or is a subsidiary plan of the project management plan.
Contractor Team Members	People who are developers, systems analysts or project managers of an outsourced project team.

Term	Definition
Daily Stand-up Meeting	A short daily stand-up meeting lasting for not more than 15 minutes will be held at the same time every morning among project team members and programmers to provide a quick update of progress of individual tasks finished yesterday, work going to do today and any issues blocking the progress.
Done (Definition of Done)	A list of criteria which must be met before a requirement is considered "done". Failure to meet the criteria at the end of a timebox usually implies that the work is incomplete and should not be counted into the timebox's velocity.
Internal Project Manager (Internal PM)	A lead person (Government resource) accountable for project planning and delivery. As the person is responsible for meeting the project objective and ensuring project success, the project manager is expected to provide oversight and input to project management aspects such as project workplan, budget, quality, risks, and issues, as well as management of contractor resource performance through working closely with the contractor project manager in an outsourced project.
Low-level requirements	In this document, low-level requirements refers to those requirements which are within the project scope but not fixed in the contract (in the case of outsourcing) or in the system design stage. Such low-level requirements are to be defined or refined during the timeboxes of the system development phase.
MoSCoW	MoSCoW (Must, Should, Could and Won't) Analysis is a technique that helps users to prioritise each requirement based on its importance.
Prioritised Requirements List (PRL)	A list of requirements for the project, which have been prioritised using the MoSCoW technique. Agile's concept on prioritisation of requirements suggests users and project team to use one single prioritised requirement list to improve transparency, clarity and control.
Programmers	People who design and implement the system. They can be project team members (for in-house projects) or contractor team members (for outsourced projects)

Term	Definition
Project Assurance Team (PAT)	The PAT looks after the quality assurance work on behalf of the PSC from the business, user and technical perspectives. They are individuals who have knowledge or expertise in the specific subject matter area that is part of the project scope. The PAT consists of Business Assurance Coordinator (BAC), User Assurance Coordinator (UAC) and Technical Assurance Coordinator (TAC), that is a balanced representation of the Business User and technical interests.
Project Management Plan (PMP)	A formal, approved document that defines how the project is executed, monitored and controlled. It is used as a live document during the course of the project and is composed of subsidiary management plans from other project dimensions and other planning documents.
Project Steering Committee (PSC)	The PSC is accountable to the project owner for the progress and performance of the project. They decide on all actions needed in order to complete the project.
Project Team	Includes people who design and implement the system. They can be project team members (for in-house projects) or contractor team members (for outsourced projects).
Quality Management Plan	The Quality Management Plan describes the activities required to ensure that project deliverables are meeting specifications, meeting quality standards and functioning correctly, etc. (the quality control activities).
Release	A release refers to the target functions/features of working software to be delivered to users after the implementation of a planned number of timeboxes is completed.
Release Plan	A release plan represents a schedule for delivering one or more releases to users, i.e. the target functions/features of working software to be delivered to users by different deadlines.
Story board	It is a notice board that shows the completion status of each user story in every timebox during development. It tells the project team how many user stories are completed or still outstanding.

Term	Definition
Story Points	It is a unit used to measure the relative effort required to complete a user story. It is normally expressed in numbers e.g. 10, 20, 50, 100, etc.
Systems Architect	Systems Architect defines the structure, behaviour, and more views of a system, and is also responsible for the overall design of the systems. For small projects, this role is usually taken up by the Systems Analyst.
Task Board	It is a notice board that shows the progress of each task in every timebox during development. The progress status is normally defined as "To Do", "In Progress" and "Done".
Test-Driven Development (TDD)	Test-Driven Development refers to a style of programming in which three activities are tightly interwoven: coding, testing (in the form of writing unit tests) and design.
Timebox	The development time for the entire project is divided into multiple iterations of fixed time interval. Each interval is called a timebox. Each user story with its associated tasks is assigned to one timebox for implementation.
Timebox Plan	A timebox plan details the deliverables of a specific timebox, along with the activities to produce those deliverables and the resources to do the work.
User Representative(s)	People who provide user requirements for the IT system under development.
User Story	User stories act as requirements for the system, and are written by users in layman's terms that facilitate verbal communication between the end users/Internal PM and project team. The format of a user story is: [As a <Role>, I want to <goal> so that I can <reason>]
Velocity	The speed at which the project team are working, measured in terms of the total estimated effort associated with the user stories that were completed at the end of a timebox. Metrics related to this value across many timeboxes can help a team to establish a sustainable pace of working.