CABA

IQBBA

Certified Agile Business Analysis
by A4Q

CERTIFIED AGILE BUSINESS ANALYSIS SYLLABUS

An A4Q Certification

A4Q Alliance for Qualification

Content

Introduction to this Syllabus

Certified Agile Business Analysis

As agile grows in maturity, it is increasingly evident that Business Analysts are fundamental to the delivery of features that yield a satisfactory return on enterprise investment. This is the domain of the business analyst.

While both developers and testers have clear and recognizable roles in the agile development environment, the role of the Business Analyst is not so well defined. This may be because business analysis is misunderstood in general but is probably exacerbated by the lack of any widely accepted definition or framework for agile Business Analysis - until now.

The International Institute of Business Analysis™ (IIBA®) provides business analysis guidelines in A Guide to the Business Analysis Body of Knowledge® (BABOK®Guide) and, in collaboration with the Agile Alliance® has developed the Agile Extension.

The IQBBA Certified Agile Business Analysis course is focused on the work that a Business Analyst practitioner will face in an agile context. The Agile Business Analysis Syllabus and training will support practitioners to become familiarized with the context, the practices and the challenges they will face, and ensure they will be ready to perform.

Course Outcomes

At the end of the course, successful participants will be able to:

- Identify the role of a business analyst in agile software development projects
- Participate in agile teams developing software
- Articulate the BA's responsibilities to both the enterprise and the agile team
- Understand and apply the agile business analysis principles and techniques
- Know BA techniques that enable the delivery of artefacts that add value to the enterprise developing agile projects
- Appreciate the importance of, and how to contribute to, continuous improvement of agile processes

Organization of this Syllabus

This syllabus contains nine major chapters. The top-level heading of each chapter shows the highest level of the learning objectives that is covered within the chapter, and specifies the minimum time to be spent for training in the chapter.

Thank you

We hope this course is enjoyable, engaging and beneficial to you. Any comments or suggestions you may have about the course and examination are welcomed. We look forward to seeing you at future A4Q training and certification program.

1 What is Business Analysis? (K2)

Timing	50 minutes
Terms	Business Analysis, levels of business analysis, Business Analyst competencies

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

1.1 Business Analysis (K1)
LO-1.1.2: Recall a definition of the term business analysis (K1)
1.2 The Business Analyst (K2)
LO-1.2.1: Recognize a common definition of the term business analyst and identify its weakness (K2)
LO-1.2.2: Describe three levels of business analysis and their relevance to the business analyst (K2)
1.3 Competencies of a Business Analyst (K2)
LO-1.3.1: Recognize the six categories of competency for a BA and provide examples (K2)

1.1 Business Analysis (K1)

1.1.1 Introduction

Strictly speaking, there is a difference between business analysis and the work of business analysts. It can be argued that many people in an organization will be performing business analysis which is developing and redesigning business strategies, objectives, operations and procedures to raise quality reduce costs and increase efficiency.

IT has enabled many beneficial enterprise changes and has revolutionized some sectors such as retailing and open government on the internet. However, it has also attracted criticism for not sufficiently considering the organizational perspective.

There has, consequently, been a drive to ensure enterprise strategy and objectives are at the heart of business systems implementation and further, that those needs are unambiguously defined and understood by all stakeholders.

In the past, Systems Analysts were responsible for identifying and recording system requirements, but that role was, by definition, focused on IT systems and IT systems are not the answer to all business problems.

For business analysis to be effective, the analyst must consider a much wider perspective which encompasses the organization (the way the enterprise is organized), the people, the business processes, the external landscape (political, economic, market conditions etc.) and even the culture of the institution as well as the supporting / enabling technologies.

The emergence of the business analyst occurred with the recognition that this wider business context was needed to ensure successful changes that aligned with enterprise goals.

While some have claimed that the BA role is the most difficult to rationalize in an agile setting, the original justifications for creating a specialized business analysis role remain as compelling as they ever were.

1.1.2 Definitions (K1)

LO-1.1.2: Recall a definition of the term business analysis (K1)

Various organizations and writers have defined business analysis.

The definitions below are from Wikipedia, the International Institute of Business Analysis (IIBA®) and the Australian Institute of Business Analysis (AIBA).

- Wikipedia Definition of a Business Analysis: Business analysis is a research discipline of identifying business needs and determining solutions to business problems. http://www.wikipedia.org

- IIBA Definition of Business Analysis: Business analysis is the practice of enabling change in an enterprise by defining needs and recommending solutions that deliver value to stakeholders. Business analysis enables an enterprise to articulate needs and the rationale for change, and to design and describe solutions that can deliver value. IIBA BABOK Guide Version 3.0

- AIBA Definition of Business Analysis: ...effective Business Analysis capability will drive successful project execution and organizational performance improvement. http://businessanalysis.com.au/

A common theme seems to be that business analysis is concerned with identifying and / or responding to opportunities to improve business effectiveness or performance. Business effectiveness might be described as an enterprise's ability to achieve its objectives (or goals).

It's worth noticing that none of the definitions above mention computers although the sources acknowledge that business analysts will commonly engage with the Information Technology (IT) domain. Wikipedia says "Solutions often include a systems development component, but may also consist of process improvement, organizational change or strategic planning and policy development".

- IIBA Definition of Agile Business Analysis: Agile business analysis is the practice of business analysis in an agile context with an agile mindset. Agile business analysis can provide a competitive advantage in fast-paced and complex environments and has moved beyond software development initiatives to any business domain. Agile business analysis focuses on maximizing business value. This constant focus on business value produces better and more timely business outcomes. http://www.iiba.org/BABOK-Guide/Agile-Extension-to-the-BABOK-Guide-IIBA.aspx

1.2 The Business Analyst (K2)

Some writers and organizations consider that anyone engaging in business analysis tasks is a business analyst.

1.2.1 Definition (K2)

LO-1.2.1: Recognize a common definition of the term business analyst and identify its weakness (K2)

- IIBA Definition of a Business Analyst: A business analyst is any person who performs business analysis tasks described, no matter their job title or organizational role. IIBA BABOK Guide Version 3.0

However, it seems pertinent to observe that anyone who takes a photograph is not a photographer. That is to say, they do not have the easy access to tools, techniques and tricks of the trade that come from exercising their professional skills week in and week out. And so, it is with business analysis and business analysts.

1.2.2 The Scope of Business Analysis (K2)

LO-1.2.2: Describe three levels of business analysis and their relevance to the business analyst (K2)

While it's not always the case, typically, BAs begin their careers in the systems requirements area. Because systems are normally developed to support business processes (and automating an inefficient process is wasted investment), the BA later becomes involved in the definition, design and redesign of those processes. And finally, the processes exist to meet the objectives of the enterprise. BAs will find that this is the area that creates the context for their work.

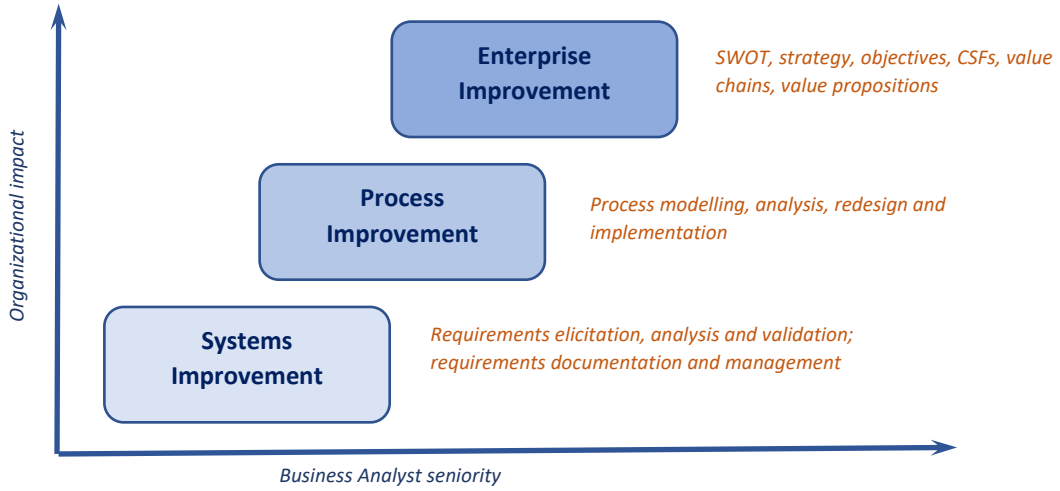


Figure 24 – Development and Beyond

In the *Agile Extension* V2, a new representation of these levels is presented, and referred to as *horizons.*

Depending on organization level, the view of work for a BA is different. Being highly impacted by the planning time frame and the nature of the feedback loops.

In these three levels of business analysis, the techniques presented in the following chapters are all relevant and applicable.

1.3 Competencies of a Business Analyst (K2)

Firstly, it's worth defining what is meant by the term competent. The word competent is commonly used to mean that someone (or some group) possesses ability to an adequate degree. For example, "She is a competent workshop facilitator" means, she can adequately facilitate workshops. Organizations seeking to improve the competencies of their BAs must first decide what abilities they wish a BA to have. Many turn to external organizations for some guidance on these matters.

The IIBA® offers a BA-specific competency assessment tool and the e-CF and SFIA (Skills For the Information Age) model each provide frameworks to define competencies for all information technology related roles, including business analysts.

1.3.1 Underlying Competencies (K2)

LO-1.3.1: Recognize the six categories of competency for a BA and provide examples (K2)

The underlying competencies" for BAs are presented under six categories.

1. Analytical Thinking and Problem Solving
   - Creative Thinking
   - Decision Making
   - Learning
   - Problem Solving
   - Systems Thinking
   - Conceptual Thinking
   - Visual Thinking.
2. Behavioral Characteristics
   - Ethics
   - Personal Accountability
   - Trustworthiness
   - Organization and Time Management
   - Adaptability
3. Business Knowledge
   - Business Acumen
   - Industry Knowledge
   - Organization Knowledge
   - Solution Knowledge
   - Methodology Knowledge
4. Communication Skills
   - Verbal Communication
   - Non-Verbal Communication
   - Written Communication
   - Listening
5. Interaction Skills
   - Facilitation
   - Leadership & Influencing
   - Teamwork
   - Negotiation and Conflict Resolution
   - Teaching
6. Tools and Technology
   - Office Productivity Tools and Technology
   - Business Analysis Tools and Technology
   - Communication Tools and Technology

2 What is Agile? (K1)

Timing	15 minutes
Terms	Agile Manifesto, Agile Principles, traditional SDLC, Iron Triangle

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

2.1 Introduction
No learning objectives
2.2 Agile (K1)
LO-2.2.1: Recall the Agile Manifesto (K1)
LO-2.2.2: Recognize the 12 principles that underpin the Agile Manifesto (K1)
2.3 Agile versus Traditional (K1)
LO-2.3.1: Compare and contrast the traditional SDLC with the agile approach identifying differences, similarities and perceived benefits of each (K1)

2.1 Introduction

Traditional, sequential software SDLCs, including Waterfall and V-Model, are based on an engineering lifecycle of

- specify the product,
- design the product,
- build and
- test the product.

This works well if you're constructing a bridge across a river. From the time that you specify the needs and design the structure to the time that you build, it's unlikely that the river will decide to go somewhere else or that the bedrock will change from sandstone to granite.

Similarly, traditional software development models are based on assumptions that:

- Clients know what they want
- Clients can clearly articulate what they want in such a way that
- everybody will clearly understand their requirements
- The team is able to determine everything it needs to know about the new system upfront therefore allowing accurate estimation of build time and cost
- Having defined what is to be built, nothing will need to change

However, business requirements are, typically, more unpredictable. Moreover, with the power of technology (particularly due to digitalization) and its effect on the speed of business model change, requirement instability has increased.

These business requirement changes inevitably contribute to the overrun that is associated with traditional development lifecycles. Waterfall projects are beset with issues arising from changing business needs and more flexible approaches (such as Rapid Application Development) have long been suggested.

2.2 Agile (K1)

Iterative and incremental development approaches are not new, and neither is the concept of "lightweight" software development methods. The move to less process constrained and documentation-heavy software development has been driven by work in the manufacturing domain that has continually striven to reduce wastage and improve production efficiency without loss of quality in the end product.

Various software development experts applied these concepts to software projects and produced a set of methods that were originally categorized as "lightweight". They included:

- Scrum (Ken Schwaber, Jeff Sutherland, Mark Beedle)
- Extreme Programming (XP) (Kent Beck, Eric Gamma and others)
- Dynamic System Development Method (DSDM) (Dane Faulkner and others)
- Agile Unified Process (or Agile RUP) (Scott Ambler)
- Feature Driven Development (Peter Coad and Jeff Deluca)
- Lean Software Development (Mary and Tom Poppendieck)
- Crystal Methods: A family of Methods (Alistair Cockburn)
- Adaptive Software Development (Jim Highsmith)

2.2.1 The Agile Manifesto (K1)

LO-2.2.1: Recall the Agile Manifesto (K1)

In February 2001, the creators of these methods, and some like-minded development professionals, met to learn if they had common cause. The output from the meeting was the "Agile Manifesto" (figure 10 below) and a set of guiding principles (figure 11) for agile software development methods. The Manifesto forms a foundation upon which all agile methods are based.

The Agile Manifesto

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

http://agilemanifesto.org/

2.2.2 Agile's Twelve Principles (K1)

LO-2.2.2: Recognize the 12 principles that underpin the Agile Manifesto (K1)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity- the art of maximizing the amount of work not done – is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly
    http://agilemanifesto.org/

2.3 Agile versus Traditional (K1)

2.3.1 Differences (K1)

LO-2.3.1: Compare and contrast the traditional SDLC with the agile approach identifying differences, similarities and perceived benefits of each (K1)

Martin Fowler[1] identifies two main differences between agile and traditional SDLCs.

1. "Agile methods are adaptive rather than predictive" – Agile processes embrace change while traditional methods typically discourage change
2. "Agile methods are people-oriented rather than process-oriented" – Agile methods support the development team in their work rather than procedure

Most projects start in the same way and inception may be summarized as defining:

- The customer's or business' problem that needs solving – this can be considered as the high-level scope of the project
- The time frame in which the problem must be solved – this is a constraint that will affect the project schedule
- Any upper limit on the cost of the solution – this will constrain the project resources

In traditional Project Management, these issues are often pictured as the "Iron Triangle" (figure below).

1 Fowler, (2005). The New Methodology. [online] martinfowler.com. Available at: http://www.martinfowler.com/articles/newMethodology.html

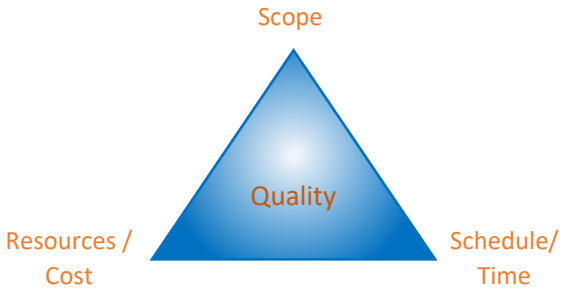


Figure 2 – The Project Management Iron Triangle

One way to think about the triangle is, given enough time and money, you could achieve (pretty much) anything to a high quality. However, if there are cost and/ or time constraints (and when isn't there?) either scope or quality has to give.

Additionally, these factors are estimated, agreed and set at the beginning of the project when, by definition, least is known about them. In consequence, there is a natural resistance to allowing changes to those early estimates because the balance between time, cost, scope and quality is disturbed.

While traditional projects attempt to control (and somewhat discourage) change, agile methods embrace it.

The agile philosophy acknowledges that "you can't know what you don't know" so new information is expected and welcomed throughout the project. There is reevaluation of direction at the beginning of each iteration (or Sprint) by working with the business to constantly reprioritize the backlog of stories, defects and technical tasks.

To effect this change in approach, agile development resource, schedule and quality constraints are fixed (or "boxed") while the scope remains fluid. Thus, the project is able to react to deviations from original expectations and / or any invalid assumptions.

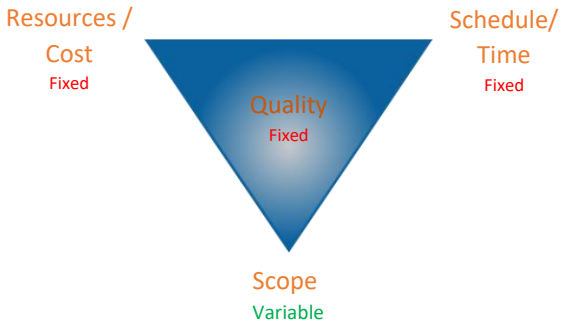


Figure 3 – The Agile Iron Triangle

Agile also focusses on cohesive teams. This is reinforced by moving responsibility, ownership and accountability for delivery to the team.

Ivar Jacobson, computer scientist, major contributor to UML and many other computer science concepts, noted in the "Everyone wants to be agile" article for the TechRepublic that, "No process has ever developed software. It has always been done by people. We have of course always known this, but we have not pushed it as much. The focus on people is really what makes agile unique, and this is why agile originally broke through."

3 Common Agile Approaches (K2)

Timing	100 minutes
Terms	Agile approach, generic model, roles, responsibilities, Scrum, XP, Kanban, Lean, BA skills, BA activities

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

3.1 A Generic Agile Process (K2)
LO-3.1.2: Recognize and describe a generic Agile approach (K2)
3.2 Agile Team Roles and Responsibilities (K2)
LO-3.2.1: Identify generic roles in agile and explain their purpose (K2)
LO-3.2.2: Describe the makeup and responsibilities of an agile team (K2)
3.3 BA Roles in Agile Development (K2)
LO-3.3: Identify agile roles and responsibilities that demand Business Analyst skills and explain why (K2)
3.4 Scrum (K1)
LO-3.4.1: Recognize the Scrum process (K1)
3.5 Extreme Programming (XP) (K1)
LO-3.5.1: Recognize the distinguishing rules and practices of XP (K1)
3.6 Lean and Kanban (K2)
LO-3.6.1: Explain how Kanban and Lean are reflected in agile methods (K2)
3.7 The Value of a Business Analyst on Agile *Projects (K2)*
LO-3.7.1: Recall the activities of a Business Analyst in an Agile Environment and describe the value they bring (K2)

3.1 A Generic Agile Process (K2)

3.1.1 Introduction

Agile is the iterative approach for developing a project.

Agile approaches focus on achieving business value based on business needs.

The approaches for which an overview is made in this chapter are:

- Scrum
- Extreme Programming
- Kanban

3.1.2 The Generic Agile Process (K2)

LO-3.1.2: Recognize and describe a generic Agile approach (K2)

The figure below shows a generic agile process that has commonality across the various approaches. It uses a backlog of items to supply either time-boxed or continuous iterations with requirements for features that make up the end product.

A business representative is responsible for maintaining the backlog and a development team (self-organizing and multi-disciplined) is responsible for delivering the agreed outcome of each iteration.

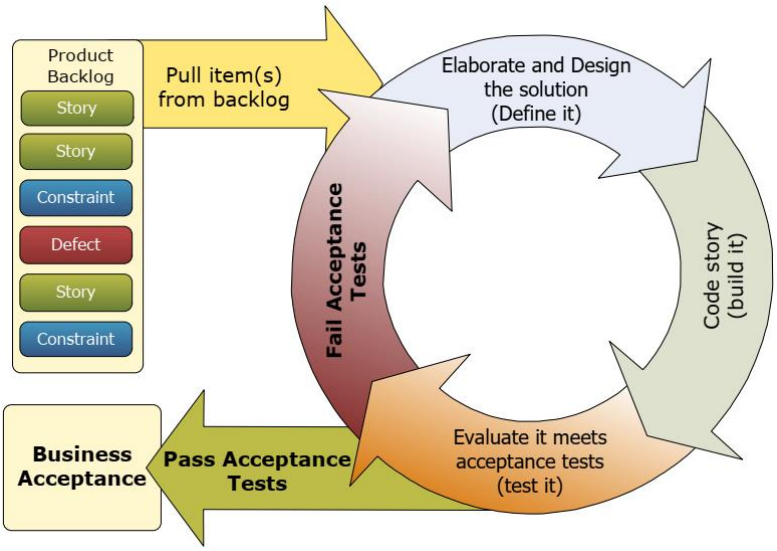


Figure 4 – Generic Agile Process

The generic model is not meant to imply only discrete "sprints" (see the section "Scrum Approach" later in this module) as the lifecycle can be continuous (see the section "Kanban Approach" later in this module below) with items pulled from the backlog as work and resource dictates.

3.2 Agile Team Roles and Responsibilities (K2)

3.2.1 Roles and responsibilities (K2)

LO-3.2.1: Identify generic roles in agile and explain their purpose (K2)
LO-3.2.2: Describe the makeup and responsibilities of an agile team (K2)

Traditional development approaches silo various roles into teams that are active at various stages of the project. So, BAs elicit and document the requirements which are passed to the designers who specify the technical components and artefacts of the system. The design team hands the design documents to a team of developers who write software to integrate into a finished system for the testers to test.

In contrast, agile teams are cross-functional containing analysis, design, coding and testing skills. Moreover, each team member may have more than one skill set.

Scrum and XP (both described later in this module) are the most commonly adopted agile approaches so their terminology is frequently used to describe generic agile roles which, in this manual are:

- Agile Team Leader,
- Business Representative,
- Developer and Tester.

Table 1 below outlines the responsibilities for these roles and also names their Scrum equivalents, in brackets, where applicable.

Role Title	Description

Agile Team Leader (Scrum Master)	The Agile Team Leader/ Scrum Master is responsible for:
	• Enforcing rules – while agile is lightweight and flexible, it has rules and here lies the responsibility for imposing the agile process. Examples include:
	  ○ Attending daily stand-ups on time
	  ○ Keeping others from interfering with the progress of the iteration
	  ○ Performing retrospectives at the end of each sprint
	  ○ Enabling the team to make decisions for itself and not making decisions on its behalf
	• Protecting the team – from any requests by people outside the team that would prevent the team members from meeting their iteration/ sprint goal.
	• Eliminating road blocks – encourage the team to resolve
	• their own problems but, if a block is outside their sphere of control or authority, the Team Leader will take responsibility to address the issue and enable the team to maintain their momentum.
	• Facilitating – this is a key technique for keeping the team
	• focused on the goal – delivery of the agreed functionality/stories at the end of the iteration
	• Helping the organization, stakeholders and employees to adopt and enact Scrum

Role Title	Description
Business Representative (Product Owner)	This role is responsible for working with customers and stakeholders to determine requirements and communicate them to the team. Scrum strongly emphasizes the role of the Product Owner but the role could be performed by a BA. The Business Representative/ Product Owner is responsible for: • Writing user stories • Prioritizing backlog items and selecting items for inclusion • in iterations, ensuring that the most important • requirements are worked on first • Being available to the team for story elaboration; using • their knowledge to drill-down to the details required to • ensure that the story is understood appropriately • Participating in acceptance testing of the stories delivered • during each iteration • Educating the Business stakeholders • Ensuring that the product road map maintains the • alignment between the product's strategy and business • needs.
Developer	Developers write working code to implement their assigned stories or complete their assigned tasks. Developers may work independently or in pairs (as in XP pair-programming). Developers: • Write code • Collaborate with product owners and testers to make sure that the “right code” is written • Write unit tests for all of the code that they produce • Write any testability hooks that will help simplify automated testing of the story or system • Check-in their code to the version control system every day. This is a key responsibility that ensures continuous integration of code into the version control system

Role Title	Description
Tester	Testers begin (static) testing, even before coding starts, by reviewing user stories. Every story that reaches the iteration boundary (the end of an iteration and the start of the next) is reviewed and tested. Key test responsibilities are: • Working with developers and product owners to ensure that the stories are clearly understood • Ensuring that the acceptance tests track the desired functionality of the story • Creating acceptance tests while the code is being written • Executing acceptance tests against completed code • Ensuring that test cases are checked into the version control system every day • Writing and maintaining automated tests that can be executed against the code every day; as part of the continuous integration and testing process

Table 1 – Agile Team Roles and Responsibilities

3.3 BA Role in the Generic Agile Process (K2)

3.3.1 BA Role in Agile Environment (K2)

LO-3.3.1: Identify agile roles and responsibilities that demand BA skills and explain why (K2)

Business Analysts may participate in the agile development process in a number of ways.

Depending on the agile approach of the project, the BA activities are handled by different roles. Going through the following agile approaches, one can identify the activities of the business analyst role.

In a generic way, the activities handled by a BA in an agile environment would be:

- Prioritize backlog items for the next iterations
- Taking up the role of a business representative, a BA creates user stories in collaboration with business stakeholders.
- Elaborate user stories (acceptance criteria) in collaboration with the team
- Provide testing support
- Accept on behalf of the business or help business to evaluate for acceptance

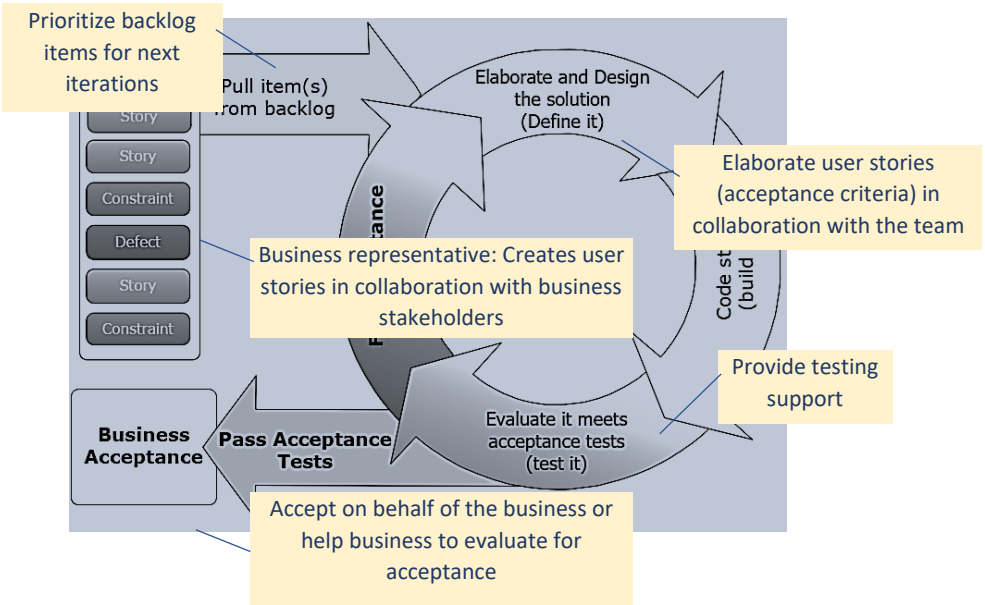


Figure 5 – The BA in the Generic Agile Process

3.4 Scrum (K1)

3.4.1 The Scrum Approach (K1)

LO-3.4.1: Recognize the Scrum process (K1)

Scrum is an iterative and incremental method developed by Ken Schwaber & Jeff Sutherland[2]. Its purpose is to provide a framework for the organization and management of product/ application development projects. Scrum welcomes self-organizing teams, encouraging co-location of, and verbal communication between, team members.

3.4.2 The Scrum Team (K1)

Within the formal rules of Scrum, imposed by the Scrum Master, Scrum teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by a manager or supervisor.

The Scrum Guide TM suggests no fewer than three and no more than nine team members.

Scrum Master

The Scrum Master is responsible for ensuring that Scrum is understood by everyone on the team and that Scrum practices and rules are adhered to.
Scrum Masters will also provide assistance in clearing obstacles (or "roadblocks") that are impeding team progress.

Product Owner

The Product Owner is responsible for representing the interests of the customer or stakeholders. Product Owners prioritize and reprioritize the items in the product backlog before each sprint.

The Development Team

This is a team of people brought together to produce a useable product (typically, working software). The team is not a group of individuals working in discrete roles and managed by a superior (Project Manager) with a plan and responsibility for completing that plan. Rather, the team members are responsible for creating the artefacts that will be delivered at the end of each Sprint. The development team comprises developers, testers, QA, BA and any other people/roles required to complete the Sprint. It is self-organizing, self-managing and cross functional so, while discrete roles can and do exist in the development team, individuals may take more than one role and/ or help other roles with their work.

3.4.3 The Scrum Process (K1)

Release Planning

Like all agile methods, Scrum uses the release planning meeting to create a plan which documents what the project intends to have released at its completion. The Release Plan defines the project at a high level and will be used to plan periods of activity (sprints) that will produce ever increasing increments of the final release.
Each sprint follows the Scrum Process (figure 6 below).

2 Schwaber and Sutherland, (2017). The Scrum Guide™. Available at: http://www.scrumguides.org/

Release planning may also be used to make other decisions that apply to the project (release) as a whole. These include items such as

- sprint duration,
- sprint team characteristics and
- Definition of Done.

On a sizable project, where these issues are more complex or intricate, an iteration (sprint) zero may be used to give more time and attention to these matters.

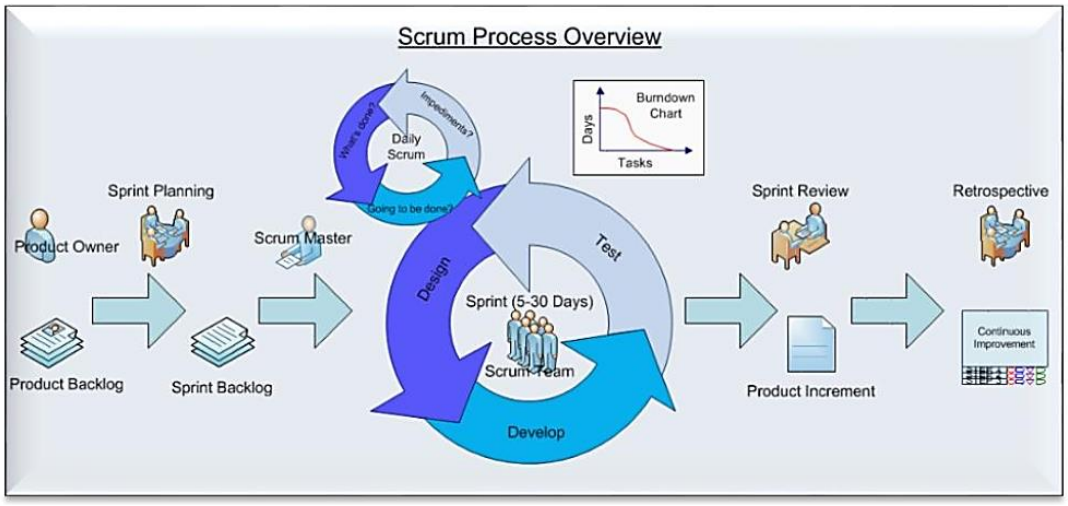


Figure 6 – The Scrum Process

Product Backlog
The Product Backlog is a ordered (by priority) list of features (requirements) that might be needed in the product and is the repository for any changes that must be made to product increments that have already been delivered. The Product Owner is responsible for creating and maintaining the Product Backlog.

Sprint Backlog
The Sprint Backlog comprises a set of selected Product Backlog items and a plan for delivering the product Increment and achieving the Sprint Goal.
The Sprint Backlog identifies the functionality (requirements) that will be addressed in the next Increment along with the work needed to deliver that increment.

Sprint Planning Meeting
The Scrum Team creates the Sprint Backlog in a Sprint Planning Meeting during which, some of the highest priority items are pulled from the product backlog.

3.4.4 The Sprint (K1)

A Sprint is a time-boxed period (a month or less) during which a product increment (functionality or artefact) is produced. Sprint duration is consistent throughout the development. When one Sprint has completed, the next Sprint begins.

Daily Scrum
At the same time of every Sprint-day, there is a team Scrum. The Scrum is a 15-minute (time-boxed) meeting for team members to confirm and align their work.

Progress (the last 24 hours) is shared and a plan for the next 24 hours is agreed.

The Sprint Review
At the end of the Sprint, the Sprint Review is held to inspect the artefacts that have been "Done". That is, the increment that has been built to the standards or criteria set out as the Definition of Done for this Sprint. This is an informal meeting with the Scrum team and relevant stakeholders (managers, users, customers etc.) as identified by the Product Owner.
The Development Team demonstrates the product increment and answers any questions arising. From this discussion, ideas may emerge regarding the contents of the next Sprint and the Product Backlog may change.

A typical Sprint Review lasts for around four hours, but it will vary according to the length of the Sprint.

Definition of Done (DoD)
Individual requirements (stories) will have their own acceptance criteria that are used to test the artefact for correctness.

The Definition of Done is determined for the project as a whole (see Release Planning) and imposes quality constraints at feature, iteration and release levels.

These include items such as: The required documentation has been completed, all code is completed by pair-programming, all high-risk defects are resolved etc.

While the DoD is likely to be different for every Scrum team, the criteria should always include that the increment is shippable. This is to say, there would be no problem if the Product Owner wished to release it immediately.

Sprint Retrospective
At the end of each sprint the Scrum team conducts a Sprint Retrospective. The purpose is to discuss

- what worked well and should be carried forward
- as well as which areas could/ should be improved.

The retrospective will focus predominantly on process, relationships among people, and tools.

It is recommended that the maximum duration of retrospective meetings should be an hour for each week of the sprint.

3.5 Extreme Programming (XP) (K1)

3.5.1 XP Approach (K1)

LO-3.5.1: Recognize the distinguishing rules and practices of XP (K1)

Extreme Programming (XP) emerged from work on a payroll project (the Chrysler Comprehensive Compensation System or C3) using a lightweight methodology to deliver the software. Kent Beck refined the approach and wrote *Extreme Programming Explained* which was published in 1999. The name originated from the term "extreme sports" which was in vogue at the time.

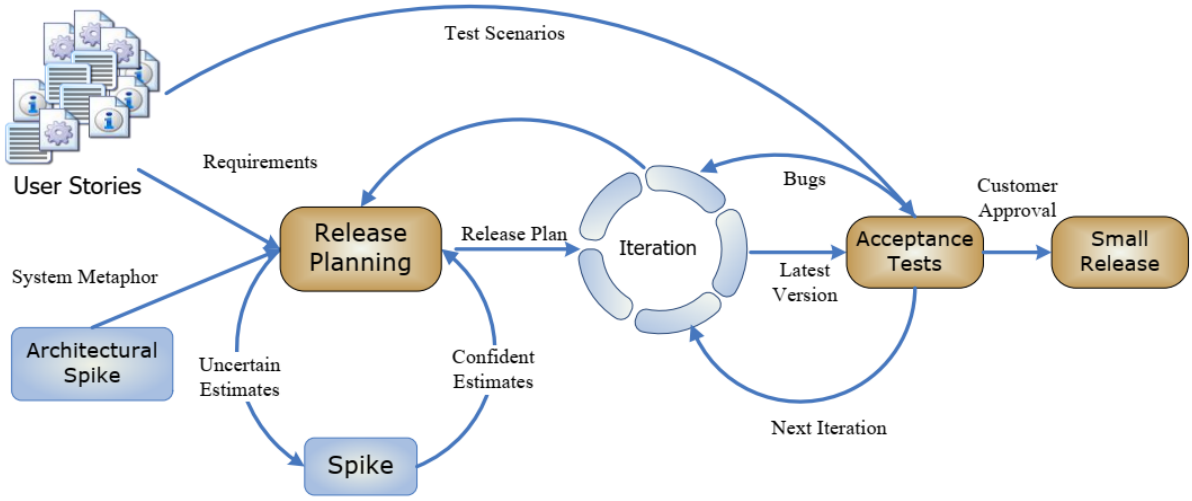


Figure 7 – The XP Process

3.5.2 Core Values (K1)

XP has five core values:

1. Communication - Daily face-to-face communication to ensure that all developers share the same view of the system and that their view matches that of system users.
2. Simplicity - Take small steps, starting with the simplest solution, producing what's been asked for and no more. Additional features/ functionality can be added later.
3. Feedback - Demonstrate functionality early and frequently. Listen to feedback and change as required.
4. Courage - Not afraid to adapt to change. Estimate truthfully and tell the truth about progress.
5. Respect - Every team member deserves respect. Developers respect the knowledge of the business and the business respects the ability of the team to self-manage, be responsible for progress and deliver.

3.5.3 XP Rules and Practices (K1)

XP practices are deliberately uncomplicated. Various writers have grouped them in different ways. The lists below are taken from the Extreme Programming site (www.extremeprogramming.org). www.extremeprogramming.org/rules.html

Planning

1. User stories are written
2. Release planning creates the schedule
3. Make frequent small releases
4. The project velocity is measured
5. The project is divided into iterations
6. Iteration planning starts each iteration
7. Move people around
8. A stand-up meeting starts each day
9. Fix XP when it breaks

Coding

1. The customer is always available
2. Code must be written to agreed standards
3. Code the Unit test first
4. All production code is pair programmed
5. Only one pair integrates code at a time
6. Integrate often
7. Use collective code ownership
8. Leave optimization till last
9. No overtime

Designing

1. Simplicity
2. Choose a system metaphor
3. Use CRC_1 cards for design sessions
4. Create $spike_2$ solutions to reduce risk
5. No functionality is added early
6. $Refactor_3$ whenever and wherever possible

Testing

1. All code must have unit tests
2. All code must pass all unit tests before it can be released
3. When a bug is found tests are created
4. Acceptance tests are run often and the score is published

3.6 Lean and Kanban (K2)

3.6.1 Lean and Kanban Approaches (K2)

LO-3.6.1: Explain how Kanban and Lean are reflected in agile methods (K2)

Lean is a manufacturing practice which originates primarily in the development of Toyota's Toyota Production System (TPS). It comprises a set of tools that help to identify and eradicate waste (or "muda" in Japanese). As waste is eliminated, quality improves and production time and costs are reduced.

3.6.2 Lean Manufacturing Principles (K2)

Lean is based on the principle that the use of resources for anything other than producing value for the end customer is wasteful.
Value is defined as any action or process that the customer is willing to pay for.

The Lean approach is based on five principal steps:

1. Specify product value from the end customer's perspective
2. Identify all steps in the product value stream and, where possible, eliminate steps that do not create value
3. Have the value-creating steps in tight sequence so production flows smoothly toward the customer
4. Let customers pull value from the value stream – that is, deliver value just in time based on demand

5. Begin the steps again (1 – 4) and continue until perfect value is created without any waste

Many of the principles of Lean, such as

- the focus on value,
- respect for people,
- teamwork,
- value pull (just in time)
- and the continuous search for improvement

are reflected in agile methods.

1 CRC = Class, Responsibilities and Collaboration cards
2 Spike - Is a technique to investigate a design or feasibility of a technology or approach
3 Refactor – Is the redesign of the code to be more efficient, clear and maintainable

Similarly, various agile approaches have adopted Lean tools such as Value Stream Mapping (see Module 5) and Kanban.

3.6.3 Kanban (K2)

Pull versus Push

Kanban originated in a study of supermarkets conducted by Toyota in the late 1940's. The company saw a parallel between shelf-stocking processes and materials supply on the factory floor. Supermarkets stocked only what they expected to sell over a given period of time and customers pulled only what they wanted from the shelves. In effect, the supermarket would restock in response to the *pull* of customer demand. It was reasoned that, in a manufacturing context, consumption should drive the demand for more production in contrast to methods that forecast demand and *push* products to market.

In production environments, Kanban cards represent demand for more materials. For example, a card placed in a bin may mean that the bin needs re-stocking. The Kanban Board is another way to register tasks that need to be done and to what stage they have been progressed. They are similar to Task Boards.

Task Boards

Many agile software development projects use task boards to identify the tasks that must be completed in a time-boxed increment. The task board makes progress highly visible to all interested parties. Typically, item (task) Post-it® notes are placed in the To-do column on the left and moved across the board as they progress towards completion.

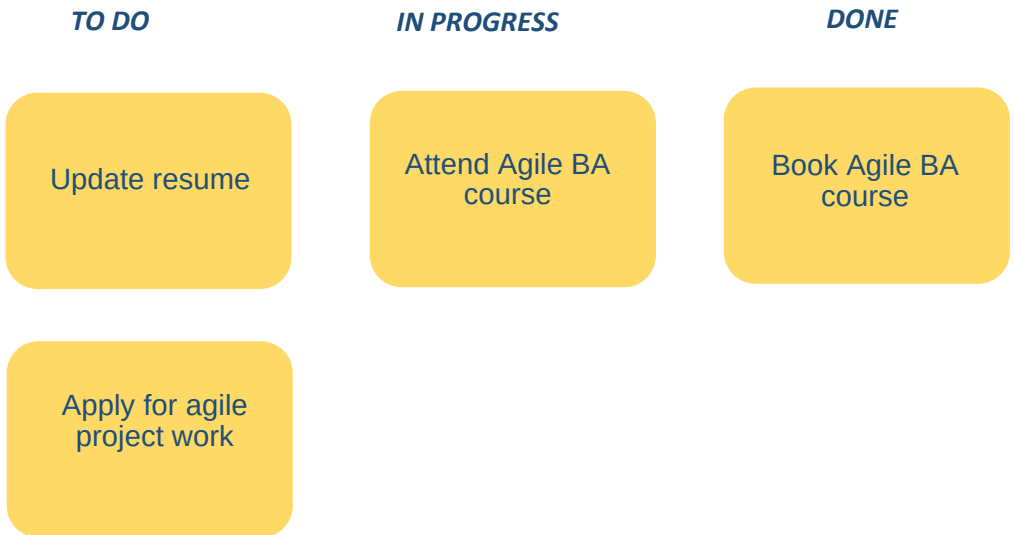


Figure 8 – A Simple Task Board

In a time-boxed iteration (or Sprint) the final column ("DONE" in figure above) should contain all of the items by the end of the iteration (figure below).

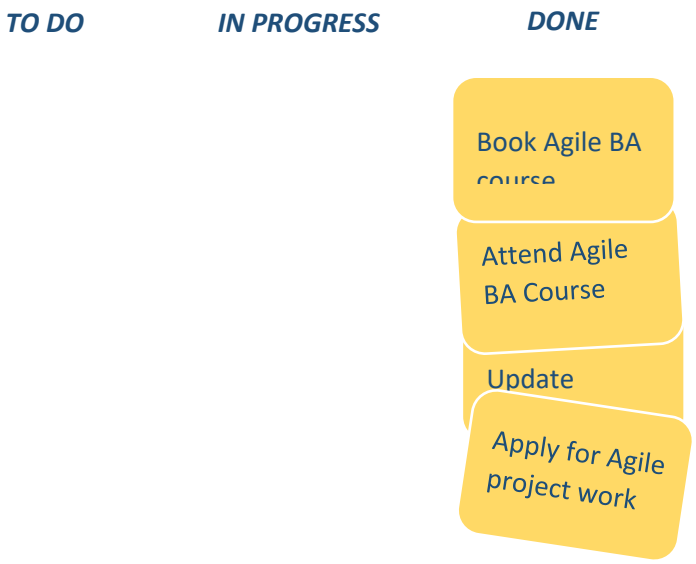


Figure 9 – Task Board at Iteration End

Kanban Boards

Kanban boards are designed to support the continuous Lean pull approach rather than the time-boxed iteration in agile. Restrictions can be placed at the head of appropriate columns to ensure that team members are not overwhelmed by a buildup of tasks. The aim is to ensure that effort flows smoothly through the work stages.

Kanban boards make roadblocks quickly visible to all interested parties. Such roadblocks happen when, on a column with a restriction placed, there are a greater number of items than the established restriction.

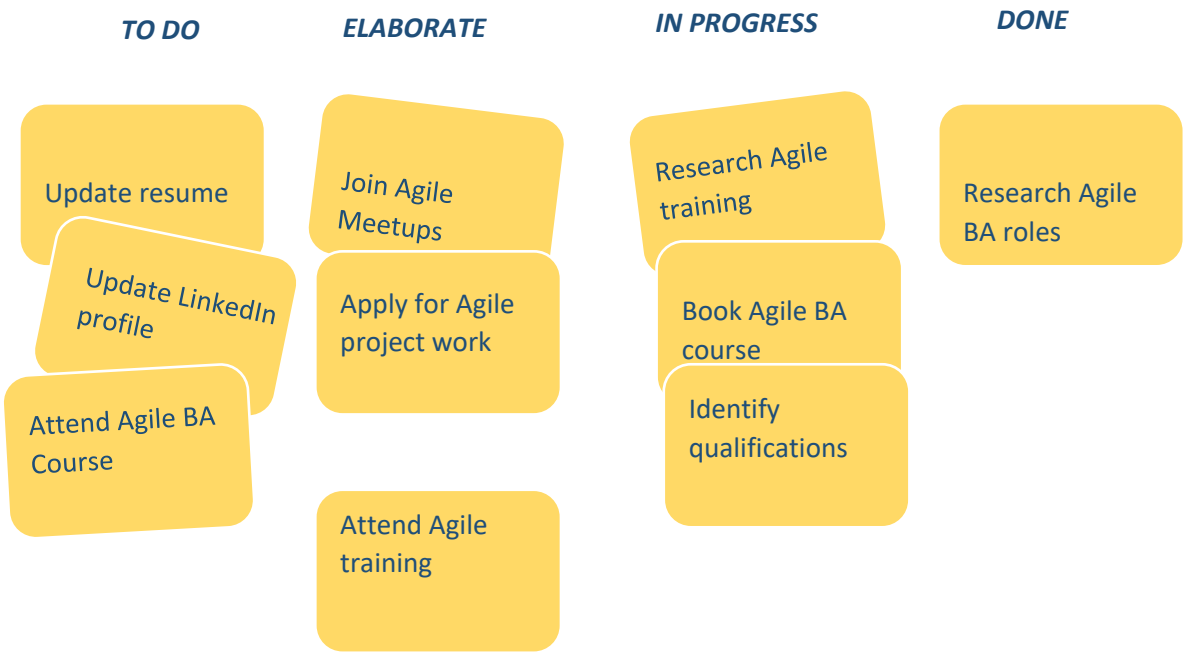


Figure 10 - Kanban board with restrictions

3.7 The Value of a Business Analyst in Agile Projects (K2)

3.7.1 The Business Analyst Activities in an Agile Environment (K2)

LO-3.7.1: Recall the activities of a Business Analyst in an Agile Environment and describe the value they bring (K2)

Whatever flavor of agile is in use, business analysis skills are as vital to project success as programming and testing. The main difference for the BA in an agile environment is that s/he will employ the techniques at different stages and to different degrees.

While this course focusses predominantly on the iterations of an agile project, it should be common for business analysis to be performed in order to identify the problem or opportunity that is of interest to the enterprise. The approach to this work may, depending on the context, be performed by a team in an agile manner or may be an assignment for a single BA working to a more traditional plan.

If the problems or opportunities are not clear, or the various stakeholders have conflicting interests and/ or business requirements, appropriate investigation and analysis work must be completed before launching any solution development, agile or otherwise. While agile teams welcome change, without a well-formed vision (often in a Product Vision Statement) or objective to work towards the project is very likely to fail.

At whatever stage in the project, analysis artefacts are commonly produced just in time (JIT) and will be lighter because dense textual descriptions (such as requirements catalogue entries) are considered wasteful of time and effort (muda).

Documentation (particularly modelling) promotes rigorous analysis and can be useful to provide context and communicate understanding but it should trigger and support, not replace, collaboration and conversation.

Because of agile's focus on people rather than process, the behavioral and interpersonal BA skillset is likely to be in higher demand.

Agile BAs can:

- provide the link between the organization's strategy and the initiatives resourced to meet the goals of the strategy,
- discover, interpret, and communicate information in order to increase understanding and clarity on where value can be created,
- clarify for whom value is created, who can contribute to the creation of value, and who else might be impacted, and
- help stakeholders make decisions about approaches, priorities, and tradeoffs to stay focused on continuous value creation in the face of constraints, differing opinions, risks, and complexity

While technicians will tend to focus on detail (their work demands it of them) BA's must attend to numerous levels of detail (from the enterprise big picture down) to ensure that the team doesn't effectively and efficiently build the wrong product.

4 Techniques in Agile Projects (K2)

Timing	50 minutes
Terms	Principles, agile techniques, business analysis techniques, planning levels

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

4.1 Introduction

No learning objectives

4.2 Principles and Techniques (K2)

LO-4.2.1: Explain the use of the principles in an agile environment (K2)

LO-4.2.2: Recognize example BA techniques that are applicable to the agile business analysis principles (K1)

LO-4.2.3: Understand, and provide examples of, how the techniques identified may contribute to business analysis on agile projects (K2)

4.3 Planning Levels (K2)

LO-4.3.1: Recognize and explain the levels of planning in the agile approach (K2)

4.1 Introduction

Having a new environment where the BA role has to be defined, the *Agile Extension to the BABOK Guide* offers analysts a set of techniques and tools to assist them in the challenges of the agile world. These will be detailed in this chapter.

4.2 Principles and Techniques (K2)

LO-4.2.1: Explain the use of the principles in an agile environment (K2)

LO-4.2.2: Recognize example BA techniques that are applicable to the agile business analysis principles (K1)

There are seven agile business analysis guiding principles. Some useful analysis techniques are associated with each principle.

- See the whole
- Think as a customer
- Analyze to determine what is valuable
- Get real using examples
- Understand what is doable
- Stimulate collaboration and continuous improvement
- Avoid waste

The principles stated here are covering two areas:

- The who, what and why: this is analogous to requirements elicitation, analysis and documentation in the traditional Requirements Engineering process. However, unlike Requirements Engineering, there is no expectation that requirements will be complete or stable in this context.
- The how and when: how the product features will behave and when the team will deliver them.

The techniques associated to each principle are:

1. See the Whole
    - Business capability analysis,
    - Personas
    - Value Stream Mapping
2. Think as a Customer
    - User story,
    - Story Decomposition,
        i. Story Elaboration,
    - Story Mapping
    - Story Boarding
3. Analyze to determine what is valuable
    - Backlog management,
    - Business value definition,
    - Kano analysis,
    - Purpose Alignment Modelling
4. Get Real Using Examples
    - Behavior driven development
5. Understand what is Doable
    - Relative estimation,
    - Planning workshops
    - Real Options
6. Stimulate Collaboration and Continuous Improvement
    - Retrospectives
    - Collaborative games
7. Avoid Waste
    - Lightweight documentation

The principles are a useful way to think about the BA's approach to agile work. However, they are not intended to imply a development process or sequence. The principles and techniques overlap and interrelate. Moreover, all of the techniques for business analysis are candidates for inclusion on agile assignments.

4.2.1 Business Analysis Techniques (K2)

LO-4.2.3: Understand, and provide examples of, how the techniques may contribute to business analysis on agile projects (K2)

On the defined seven business analysis areas of knowledge there are associated activities, tasks and techniques. The techniques described are relevant to business analysis work in any setting using any approach or method depending on the context.

Considering the agile business analysis principles and the business analysis techniques, a matrix can be created to see the relationship between them.

Business Analysis Technique	See the Whole	Think as a Customer	Analyze to Determine What is Valuable	Get Real using Examples	Understand What is Doable	Stimulate Collaboration and Improvement	Avoid Waste
Acceptance & Evaluation criteria definition		✓		✓			✓
Base lining		✓	✓				✓
Benchmarking	✓						✓
Brainstorming						✓	
Business Rule Analysis	✓	✓		✓			✓
Checklists				✓			✓
Coverage Matrix	✓						✓
Data Dictionary and Glossary	✓			✓			✓
Data Flow Diagrams	✓			✓			✓
Data Modeling	✓			✓			✓
Decision Analysis				✓	✓		✓
Document Analysis	✓			✓			✓
Estimation			✓		✓		✓
Feasibility Analysis			✓		✓		✓
Focus Groups		✓	✓	✓			
Force Field Analysis	✓					✓	
Functional Decomposition		✓		✓	✓		✓
Interface Analysis		✓		✓	✓		✓
Interviews		✓				✓	
Lessons Learned Process	✓					✓	
Metrics and Key Performance Indicators	✓		✓			✓	✓
Non functional Requirements Analysis		✓		✓			✓
Observation	✓			✓		✓	
Organization Modeling	✓		✓				✓
Problem or Vision Statement	✓	✓	✓				✓
Problem Tracking	✓	✓	✓	✓			✓
Process Modeling	✓			✓			✓
Prototyping		✓		✓	✓		✓
RACI Matrix	✓			✓			

Requirements Documentation							✓
Requirements for Vendor Selection							✓
Requirements Workshops	✓	✓		✓	✓		
Risk Analysis	✓		✓	✓	✓		✓
Root Cause Analysis	✓			✓		✓	✓
Scenarios and Use Cases	✓	✓		✓			
Scope Modeling	✓			✓	✓		✓
Sequence Diagrams	✓			✓			✓
Signoff							✓
Stakeholder Map	✓			✓			
State Diagrams	✓			✓			✓
Structured Walkthrough	✓	✓	✓	✓		✓	
Survey/ Questionnaire	✓		✓				✓
SWOT Analysis	✓		✓				
Timeboxing/ Budgeting			✓		✓		
User Stories	✓			✓			✓
Variance Analysis	✓					✓	✓
Vendor Assessment				✓			✓
Voting			✓		✓	✓	

Figure 11 – BA Techniques Mapped to Agile Guidelines

A number of techniques specifically selected for their suitability to the principles are further detailed in this syllabus.

4.3 Planning Levels (K2)

4.3.1 Planning in the Agile Approach (K2)

LO-4.3.1: Recognize and explain the levels of planning in the agile approach (K2)

On an agile project, business analysts will work to produce only the required amount of information and documentation needed for the given stage of development. This just-in-time and just-enough philosophy is a matter for agreement and planning at different stages of the work.

Various sources define agile planning stages (or levels) in different ways and with different names. Figure 12 shows the common theme.

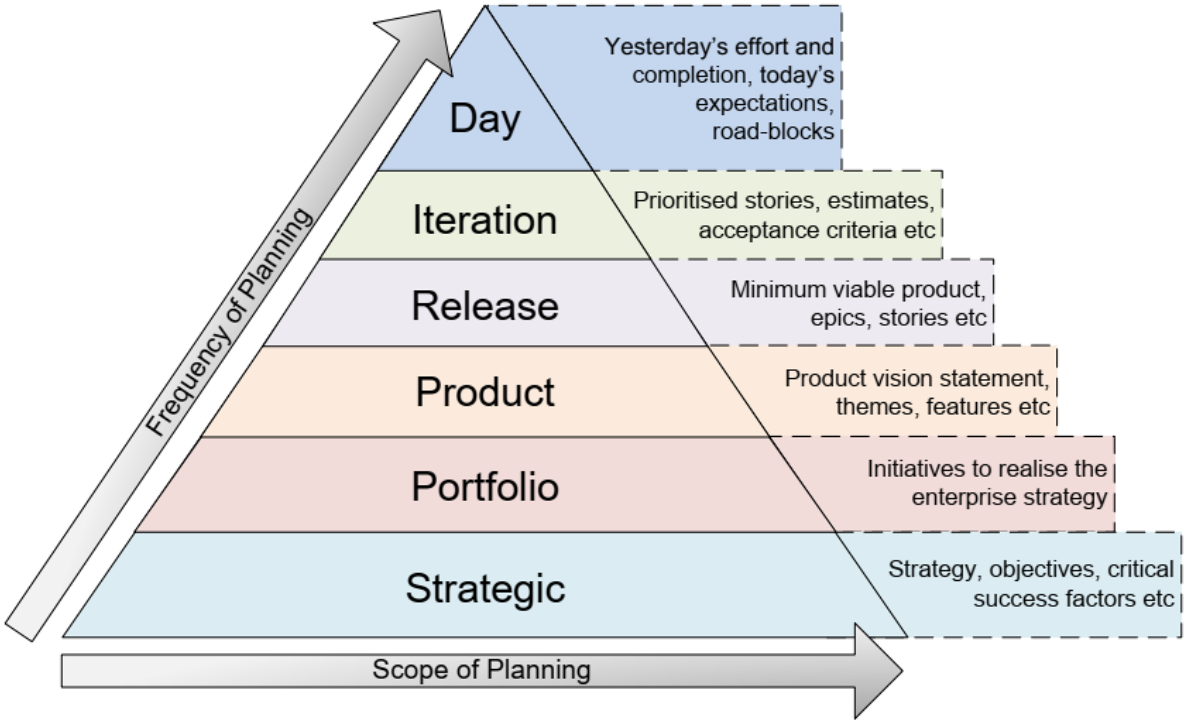


Figure 12 – Agile Planning Stages

The agile development team is primarily focused on release, iteration and daily planning. The business analyst is also likely to be involved in product planning and may contribute towards portfolio and even strategic planning. In any event, the BA must be aware of pre-release planning outputs to ensure that the product development remains aligned with enterprise objectives.

5 See the Whole (K3)

Timing	70 minutes
Terms	Business Capability, Business Process, Personas, Value Stream Mapping, model

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

5.1 Introduction

No learning objectives

5.2 Business Capability versus Business Process (K2)

LO-5.2.1: Recall the difference between business capability and business process and explain the relevance of capability analysis to an agile development project (K2)

5.3 Tool and techniques (K3)

LO-5.3.1: Create an example persona and explain the use of personas in understanding the value proposition offered by software solutions (K3)

LO-5.3.2: Summarize the value stream mapping approach contrasting it with business process mapping and highlighting potential advantages (K2)

LO-5.3.3: Create a simple current-state value stream model and articulate its business implications (K3)

5.1 Introduction

Business analysis is about returning value for enterprise investment. The value of a solution (software or otherwise) can be found in the difference between how much it costs to buy the solution and how much it costs not to buy it. That is, the value of the solution equals the cost of problem minus the cost of solution.

Agile projects are typically about developing software solutions and that requires attention to detail. Developers become immersed in code and technical challenges so there is a tendency to lose sight of the business big picture where the reason for (value of) this work exists.

The principle of See the Whole relates to the ability to attend to the development detail while maintaining awareness of relevant higher-level issues such as the market place, competition, enterprise objectives, strategy and processes.

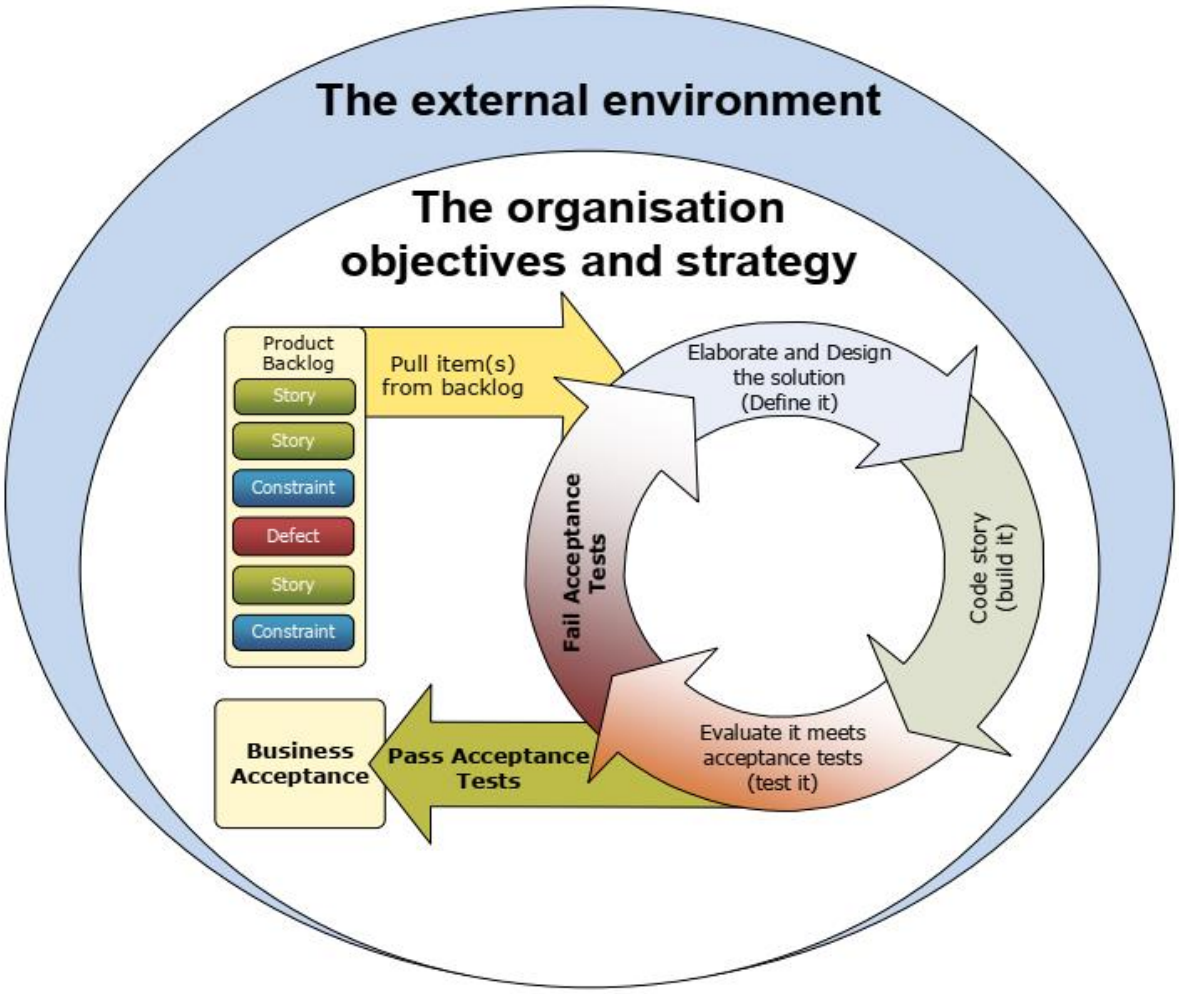


Figure 13 – The Whole

This is a key skill for business analysts in any lifecycle. For example, while defining a requirement in the traditional SDLC, a BA should immediately become alert if there seems no direct relationship between the requirement under scrutiny and the business outcomes or objectives that are driving this piece of development effort.

Common techniques for seeing the whole include:

- Business capability analysis
- Personas
- Value stream mapping

5.2 Business Capability versus Business Process (K2)

5.2.1 Differences (K2)

LO-5.2.1: Recall the difference between business capability and business process and explain the relevance of capability analysis to an agile development project (K2)

To meet their desired outcomes, organizations must be able to successfully perform certain tasks. Their ability to do so is often viewed from a process perspective.

However, a process is not a capability; it is how the organization performs a capability. So, a capability is **what** and a process is **how**.

This is an important distinction because processes are not as stable or persistent as capabilities. For example, an organization may have (or require) a sales capability but ways to perform that capability are likely to have changed radically over the past two decades with the advent of internet technologies. It's also true that there may be a number of processes that support the sales capability and all of them will be subject to change (often instigated by BAs in process improvement initiatives).

A capability model is, therefore, at a higher level of abstraction (a bigger picture) than a process model. The lower the level of detail, the more changeable a model is likely to become. A map of your home town is likely to be more persistent than a map that shows where the furniture is in your living room. Similarly, a business capability map will be useful over a longer period than a business process map.

5.3 Tools and Techniques (K3)

5.3.1 Personas (K3)

LO-5.3.1: Create an example persona and explain the use of personas in understanding the value proposition offered by software solutions (K3)

A persona is an invented character who represents a group of customers or stakeholders and provides the development team (and other interested stakeholders) with a person to focus on as a client for the product. It is normally more effective and efficient to define what might benefit a specific person rather than to have an abstract group or market segment in mind. As a part of See the Whole, personas may seem, at first glance, rather detailed. However, their use focusses the team on the client experience and needs which are drivers for the development work.

A brief examination of computerized systems designed by technologists frequently reveals a disassociation from the people who will have to use the product. Microwave ovens that have buttons to do everything except the thing you want to do; which is reheat a cup of coffee. The satnav that insists on knowing the street number (which you don't know) before it will let you enter the street name (which you'll probably know if it provides a dropdown list after the first couple of letters). It seems that nobody thought to imagine you using the product when they were designing and building it.

Personas are about more than user experience (UX). They address:

- value propositions,
- customer expectations and
- user satisfaction.

Personas can help to ensure that the development team considers the reason for the product rather than just the technical challenges of building it. Typical personas will have details like

- name,
- gender,
- age,
- family,
- income,
- home town,
- workplace,
- education,
- qualifications,

- attitudes,
- behaviors,
- personal preferences,
- expectations
- and more (they will have a picture of the persona)

Personas aren't just fantasized. Dependent on context, personas should be based on facts from

- professional surveys,
- subject matter or domain experts,
- industry statistics
- and/ or your own research.

5.3.2 Value Stream Mapping (K3)

LO-5.3.2: Summarize the value stream mapping approach contrasting it with business process mapping and highlighting potential advantages (K2)

LO-5.3.3: Create a simple current-state value stream model and articulate its business implications (K3)

Value Stream Mapping (VSM) is a technique which is used to help design processes that provide optimum value to the customer with minimum waste. A value stream map is similar to a flow chart or business process model but uses specific notation to depict and improve the flow of inventory and information.

In the graphical representation of a VSM, it is depicted the flow of value from the raw material to the final product for the customer. Value in this case means material and information.

Basically, the VSM comes as a bigger picture over the processes of the business, that drives the production of the finite product.

6 Think as a Customer (K3)

Timing	110 minutes
Terms	User story, INVEST, Story decomposition, MVP, MMF, themes, epics, story map, storyboarding

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

6.1 Introduction
No learning objectives.
6.2 User stories (K3)
LO-6.2.1: Write a user story to a defined standard (K3)
LO-6.2.2: Recall the meaning of INVEST (K1)
LO-6.2.3: Identify a common user story elaboration approach (K1)
6.3 Story decomposition (K2)
LO-6.3.1: Understand and explain story decomposition (K2)
LO-6.3.2: Describe the purpose of and difference between, MVPs, MMFs, themes, epics and user stories (K2)
6.4 Story mapping (K3)
LO-6.4.1: Explain the story mapping technique and its uses (K2)
LO-6.4.2: Create a simple story map (K3)
6.5 Storyboarding (K2)
LO-6.5.1: Describe the use and benefits of storyboarding in an agile development environment (K2)

6.1 Introduction

The “Think as a Customer” principle is a key component of the agile business analysis. This analysis starts with the customer goals, at a high level view, and decomposes these down to the detailed view of the specific needs that the product must meet.

The techniques that can be used are presented in this chapter.

6.2 User Stories (K3)

Many agile methods use user stories to:

- Record and prioritize requirements
- Underpin investigation and analysis
- Define user acceptance criteria
- Track and report project and delivery progress
- Report on value produced
- Estimate and plan

6.2.1 User Story Writing Standard (K3)

LO-6.2.1: Write a user story to a defined standard (K3)

User stories normally come in the form:

As a [person/ role], I want to [do something] so that I can [achieve something].

For example:

"As someone booked on a course I want to login so I can access my booking details"

User stories are the responsibility of, and are created by, the product owner in collaboration with business stakeholders. Stories are added to the product backlog and will be selected later (primarily on priority) for development in an iteration.

Mike Cohn (Mountain Goat Software) says user stories are often written on index cards, stored in a shoe box and laid out on a pin-board or table for planning or discussion. In this way, the user story encourages face to face communication rather than documentation.

6.2.2 User Story Quality – INVEST (K1)

LO-6.2.2: Recall the meaning of INVEST (K1)

According to Bill Wake (INVEST in Good Stories, and SMART Tasks – www.XP123.com) a good user story will be:

- Independent – which means cohesive or self-contained. That is, it has no reliance on another user story. This is common best practice in software design
- Negotiable – always subject to change up to the point of inclusion in an iteration
- Valuable – of value to the client or user
- Estimable – the time required to deliver the story as working software can be estimated
- Small – small enough to plan as part of a release and/ or an iteration
- Testable – has acceptance criteria that are unambiguous enough to create tests for

6.2.3 Elaboration (K1)

LO-6.2.3: Identify a common user story elaboration approach (K1)

In eXtreme Programing, the process of creating and elaborating user stories is commonly referred to as "the three Cs".

- Card – the user story is written on an index card
- Conversation – with the business representative to understand and confirm the details and meaning of the user story
- Confirmation – tests to determine that the user story delivers the required value to the clients

Card

User stories begin as brief statements of need often written on an index card, sticky note or electronic equivalent.

Conversation

When the story is selected for development in an iteration, a conversation between the developers and the business representative will identify more details. Ways to document such details include:

- Models (such as use case or class diagrams) for clarity
- Screen or report mock-ups (typically low-fi)
- Data tables (input and output)
- Tasks required to produce the software function
- Scenarios to illustrate use of the software
- Acceptance criteria against which the function will be tested
- Anything else that's useful

Confirmation

Acceptance criteria are typically added to the back of the card.

Note that, in addition to the acceptance criteria, the definition of done must be met for any deliverable to be accepted.

6.3 Story Decomposition (K2)

6.3.1 A Guide for Story Decomposition (K2)

LO-6.3.1: Understand and explain story decomposition (K2)
LO-6.3.2: Describe the purpose of and difference between, MVPs, MMFs, themes, epics and user stories (K2)

Stories may be created at different levels, ranging from business goals and general themes down to individual functions and their acceptance conditions.

Various sources call the different levels of stories by different names and describe them differently. However, while there is no exact taxonomy, there are clear general agreements. As a guide:

- Goal - high level business driver
- Themes – high level features (such as Stock Management or Production Planning) that will have many epics and user stories within them (Add Stock, Amend Stock, Allocate Stock etc.)
- MVP - Minimal Viable Product (MVP - defined in the *Agile Extension* Version 2 page 69) is an aggregation of MMFs (see below). The MVP represents the least amount of features that the product requires to be of value
- MMF - Minimal Marketable Feature (MMF is defined in the *Agile Extension* Version 2 (page 97) as features or components of valuable functionality to be considered for inclusion in the product
- Epic – a user story that's too large to complete in a single iteration. It will be decomposed into a number of appropriately levelled user stories at the appropriate time
- User story – describes a discrete software function from the software user's perspective
- Acceptance criteria – provide user story details as conditions or tests that the completed functionality must meet to be acceptable

Story Decomposition is one way to ensure that requirements are linked to a business objective. Because of the JIT nature of agile development, Story Decomposition is usually conducted over time. Early analysis (at strategic, portfolio and product planning phases) will identify goals, themes, epics, MMFs, and MVPs. User stories will, typically, emerge at release planning level and acceptance criteria at iteration planning time.

6.4 Story Mapping (K3)

LO-6.4.1: Explain the story mapping technique and its uses (K2)

LO-6.4.2: Create a simple story map (K3)

Story mapping is a technique that may be used to discuss and agree sequence and / or groupings. Story mapping is employed for a number of uses and context will dictate the way the map is constructed.

Commonly, the top of the model has themes, high level features or epics sequenced in order of client activity (left to right). Placed below each of these are the stories and / or epics that relate to that theme.

During product planning, story mapping may be used to discuss, agree and document the features that constitute the Minimum Viable Product (MVP) to be built in the first release and which features are candidates for subsequent releases.

Story mapping may also be used to create a roadmap for an agile project with themes prioritized from left to right and stories (within themes) prioritized top to bottom.

6.5 Storyboarding (K2)

LO-6.5.1: Describe the use and benefits of storyboarding in an agile development environment (K2)

Storyboarding is a visual scenario modelling technique used to explore user interactions with the software product. The storyboard may be based on a persona and typically spans multiple user stories.

Storyboards provide a low-fi alternative to prototyping. Storyboarding is normally conducted in workshops with the appropriate stakeholders contributing to the discussion and the story using a white board and sticky notes. The aim is to elicit, Analyze, agree and validate requirements.

The storyboard will help interface design and elaborate the stories by placing them in a real-world context.

7 Determine: What is of Value? (K3)

Timing	65 minutes
Terms	Backlog, Backlog management, Purpose Alignment Model, Kano analysis

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

7.1 Introduction
No learning objectives
7.2 Backlog Prioritization
LO-7.2.1: Understand and describe the elements of product backlog (K2)
7.3 Business Value Definition
LO-7.3.1: Explain the need to determine and communicate business value (K2)
7.4 Tools and techniques
LO-7.4.1: Describe the Purpose Alignment Model and its value using illustrative application/product examples (K2)
LO-7.4.2: Explain the purpose and process of Kano prioritization and apply Kano analysis to a simple example (K3)

7.1 Introduction

All work initiated in an organization should increase or protect the value of that organization. If it doesn't, it is a waste of time, money and effort.

It is not uncommon for organizations to start initiatives without properly identifying the needs they have or the benefits they expect. It is uncommon for such projects to be successful. Numerous examples can be found by talking to colleagues and associates about their own experience and industry surveys support the stories you'll hear.

An independent study in the United States and United Kingdom (conducted by Opinion Matters for 1E in 2011) identified US$15 billion of unused software ("shelfware") purchased for PCs alone. In a 2013 survey (by Flexera Software) 56% of companies surveyed reported at least 10% of their application spend is on tools that are under-utilized.

7.2 Backlog Prioritization (K2)

LO-7.2.1: Understand and describe the elements of product backlog (K2)

7.2.1 The Product Backlog (K2)

The product backlog will be initiated at the start of an agile project. Effectively, it's a sophisticated to-do list containing items that are candidates for inclusion in the project deliverable (product). Items will remain on the product backlog until they are either selected for inclusion in an iteration or no longer relevant to this product.

A backlog can contain anything that is relevant to the product's development including:

- user stories,
- epics,
- features,

- functional and non-functional requirements,
- outstanding defects,
- requested changes.

The items are ordered by business value with the highest value items at the top of the backlog.

7.2.2 Responsibility for the Backlog (K2)

The business representative (product owner) is responsible for backlog management (also known as "grooming") which means, keeping it in good order. This effectively means that new items will be added and delivered items removed. Additionally, the priority and business value of product backlog items are subject to constant review and revision.

As the project progresses, items will be pulled from the backlog for development in iterations (sprints) and, because the most important items are likely to be next in line for development, they should contain enough detail for the team to identify the tasks and estimate the effort required to complete them. These details (to ensure that items are measurable and testable) will emerge from conversation and are documented (just in time) at meetings with the development team and business stakeholders.

Because the items of greatest business importance are at the top of the list, backlog management should ensure that the development team effort is always focused on the greatest business value.

7.3 Business Value Definition (K2)

7.3.1 Define what is of Value (K2)

LO-7.3.1: Explain the need to determine and communicate business value (K2)

When business stakeholders ask for items to be included in a product, they do so from their own perspective. Issues that seem important to them may not be high priority to the organization they work for or the customers they serve.

Eliciting and articulating business value is a task that rests well in the BA's role description and competency profile. It requires investigation (elicitation) skills and drives valuable analysis. Regardless of other benefits, business value definition may identify opportunities to prevent unjustifiable expenditure of time and money.

A business analyst's job is to continually seek opportunities to ensure that enterprise initiatives deliver the outcomes required to meet strategic goals and the products produced by projects and BAU (Business As Usual) deliver business value.

Principally, this means identifying ways to:

• Reduce or avoid costs

- increase process efficiency,
- reducing unnecessary staff,
- retaining key staff,
- avoid regulatory transgressions (with associated fines and compensation payment)
- etc.

• Increase revenue

- enter new markets,
- create new markets,

- increase current market share,
- retain customers etc.

In the context of software development, if the value of a product is not socialized among the product definers, designers, developers and testers then they are deprived of the information they need to make good decisions while the product is being envisaged, designed, built, tested and accepted.

A business value definition is likely to include some product description and metrics (costs, benefits, constraints etc.). A likely source of value definition information is the business case and/ or initiation document for this initiative. For example:

- Losing staff costs our company $1.2M per annum
- Exit interviews identify "lack of information and support" as a significant factor in staff attrition
- The new Staff Portal is part of our Employer of Choice initiative. It will provide staff with all of the information and support they need to thrive within our organization
- The first release must meet the 1 June deadline and cost a maximum of $200,000

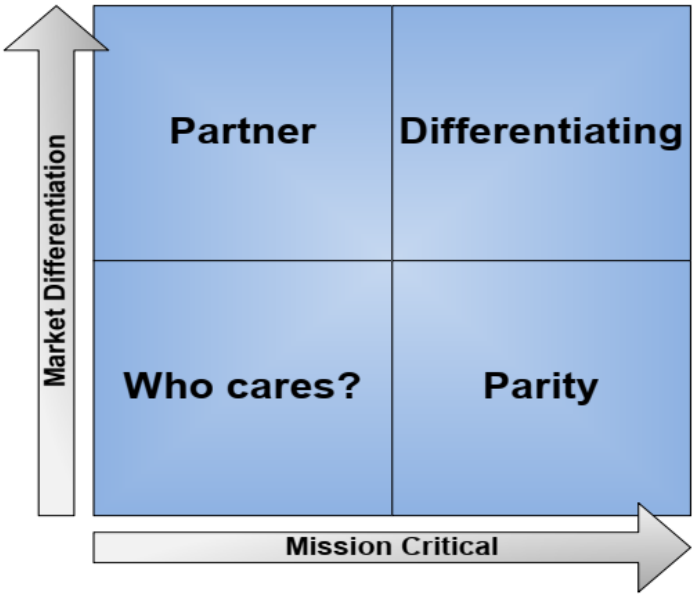
The definition is voiced from the business's (or customer's) perspective. It says why and when the product is needed and may be used to help identify the MVP/ MMF during release planning as well as providing a clear focus for valuable discussion and further decisions during development iterations.

7.4 Tools and techniques (K3)

7.4.1 Purpose Alignment Model (K2)

LO-7.4.1: Describe the Purpose Alignment Model and its value using illustrative application/product examples (K2)

The Purpose Alignment Model (or Nickolaisen Model, after its creator) is a valuable tool for identifying and analyzing the business priority of candidate product features.



The model comprises four segments across two dimensions (see figure below). The vertical dimension shows market differentiation and the horizontal, mission criticality.Figure 16 – The Purpose Alignment Model

Each segment implies a particular focus of attention:

- Differentiating – Mission-critical and high market differentiation
  Processes, activities, capabilities and requirements in this quartile are about attracting customers and gaining market share. They distinguish our enterprise from others in the same domain and we must seek to continually improve and innovate in this area.
- Parity – Mission-critical and low market differentiation
  Most of the processes (and requirements) in an organization are mission critical but not differentiating. While it is vital that the enterprise performs parity processes well, it need be no better than the rest of the industry. A restaurant must be hygienic but no more so than similar establishments; it isn't necessary to be as clean as an operating theatre.

  Treating parity processes as if they were differentiators is very costly. Thus, parity should be maintained with the rest of the business domain but not beyond. The popularity of enterprise resource planning (ERP) software, which provides industry standard business process support, results from the Parity quadrant on the Purpose Alignment Model.
- Partner – High market differentiation and low criticality
  Where differentiation is important but not critical, the enterprise might partner with an organization which treats this area as differentiating and mission-critical.
- Who cares? – Low market differentiation and low criticality
  As the name suggests, this quadrant is worth little or no investment.

7.4.2 Kano Analysis (K3)

LO-7.4.2: Explain the purpose and process of Kano prioritization and apply Kano analysis to a simple example (K3)

Requirements may be Analyzed for importance in a number of ways, one of which originates from the work of Dr Noriaki Kano (Tokyo University – 1978).

Kano was influenced by the work of Frederick Hertzberg, a psychologist, who, in 1968, developed his two-factor theory of motivation. Briefly, Hertzberg found that some things (satisfiers) were motivational. These included achievement, recognition and responsibility. Other, so called *hygiene* factors didn't motivate people if present, they merely de-motivated if absent. For example, working conditions do not motivate people very much because they expect the working environment to be adequate (or even good). However, poor conditions will have a negative impact on satisfaction.

Noriaki Kano felt that product quality could be considered in much the same manner.

Kano Classifications
Kano defines three attribute categories as shown in the figure below.

Figure 17 – Kano Importance/Satisfaction Grid

- Basic factors - must exist for the product to achieve success. Although customers may remain neutral in attitude even if these attributes are improved, they will not tolerate their absence. Sometimes known as Threshold factors, these often represent the cost of market entry and may be associated with MVP
- Performance factors - directly correlate to customer satisfaction (also known as Linear factors). Increased functionality or quality of execution will result in increased customer satisfaction. Unsurprisingly, decreased functionality will result in greater dissatisfaction
- Excitement factors – engender surprise and greater satisfaction (Delighters and Exciters) and may be market differentiators that customers are willing to pay more for. Satisfaction will not decrease below neutral if the product lacks these features because, what customers never had, they will not miss. Kano notes that, over time, features that delighted and excited clients become commonplace. Thus, exciters become linear attributes; the client is will be dissatisfied if these attributes are not present and satisfied if they are. Similarly, linear attributes that, at one time, gave satisfaction become basic must haves. The camera on a mobile phone is a good example of a once exciting addition which became a nice to have function and is now expected.

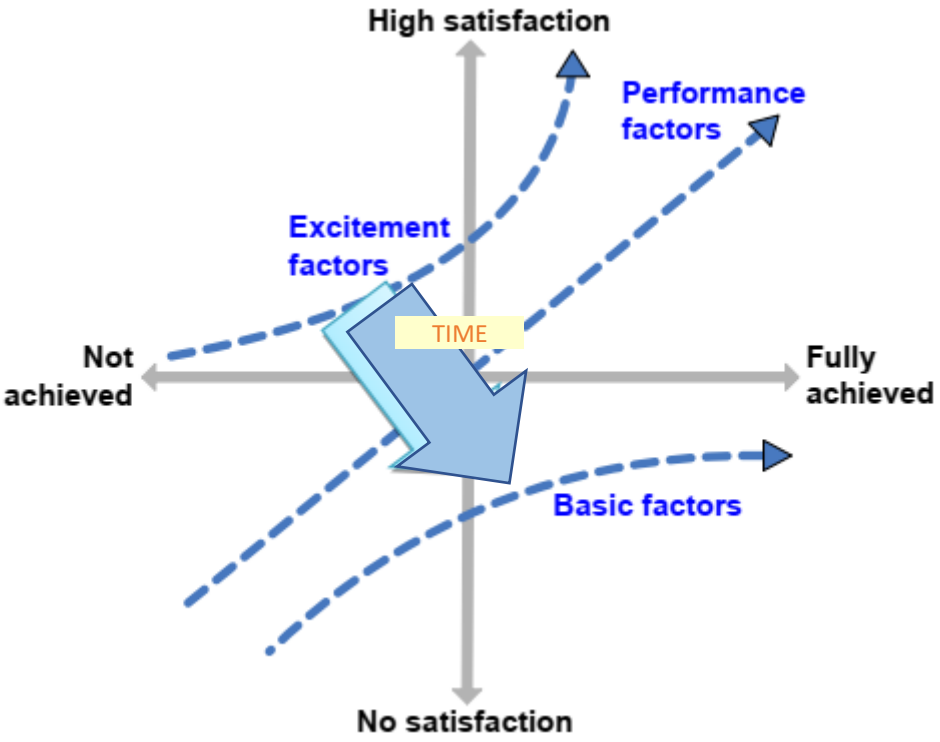


Figure 18 – The Effect of Time on the Kano Grid

Kano Questionnaires

Kano analysis uses questionnaires to identify the requirements that fall into the three categories. There are two questions for every feature or group of features. The first question is set in the positive - "How do you feel if this feature is present?" and the second, negative - "How do you feel if this feature is NOT present?" The examples in this text offer five possible answers but three (like, neutral, dislike) are also commonly used.

1. I like it that way
2. I expect it to be that way
3. I am neutral
4. I can live with it that way
5. I dislike it that way

Asking both positive and negative questions increases understanding beyond that of simple prioritization. For example, if the stakeholder expects a feature to be present, but can live without it, it is not a mandatory requirement.

The figure below shows the results of a positive/negative requirement question pair with

- E(xciter),
- L(inear),
- M(ust have),
- I(ndifferent)
- R(eject) results.

		Feature Absent				
		Like	Expect	Neutral	Live with	Dislike
Feature Present	Like	**?**	**E**	**E**	**E**	**L**
	Expect	**R**	**I**	**I**	**I**	**M**
	Neutral	**R**	**I**	**I**	**I**	**M**
	Live with	**R**	**I**	**I**	**I**	**M**
	Dislike	**R**	**R**	**R**	**R**	**?**

M(ust) **L**(inear) **E**(xciter) **I**(ndifferent) **R**(eject)

Figure 19 – Kano Grid

8 Get Real with Examples (K3)

Timing	70 minutes
Terms	BDD, test automation

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

8.1 Behavior Driven Development (BDD) (K3)
LO-8.1.1: Describe the link between user stories and BDD (K2)
LO-8.1.2: Explain the benefits of a scenario based approach to requirement definition (K2)
LO-8.1.3: Recall the structure of a BDD scenario description (K1)
LO-8.1.4: Create BDD scenarios from an example user story (K3)
8.2 BDD and Test Automation (K2)
LO-8.2.1: Summarize the process and benefits of using BDD for test automation (K2)

8.1 Behaviour Driven Development (BDD) (K3)

8.1.1 Introduction

This principle suggests using real customer examples in order to express and validate the product needs. Behavior Driver Development proposes a formal way to express the product needs as concrete examples, thus giving the stakeholders and the project team members the possibility to communicate directly on these.

8.1.2 BDD User Stories (K2)

LO-8.1.1: Describe the link between user stories and BDD (K2)

BDD user stories may be in the form:

In order to [achieve an outcome] **as a** [role] **I want** [a capability].

It is arguable that, if the first part (why) of the BDD user story cannot be completed, the following parts (who and what) are irrelevant.

"*In order to decide if a book is worth reading, as a library member, I want to see reviews by previous readers*"

8.1.3 BDD Scenarios (K3)

LO-8.1.2: Explain the benefits of a scenario based approach to requirement definition (K2)
LO-8.1.3: Recall the structure of a BDD scenario description (K1)
LO-8.1.4: Create BDD scenarios from an example user story (K3)

BDD provides a formal syntax for writing scenarios. It takes the form:

Given [a condition or context] **when** [an event occurs] **then** [an outcome or result arises].

Ands and **Or**s may be used for compound conditions (given), events (when) and outcomes (then).

In this example, one realistic scenario is that there are reviews available. Thus, the BDD structured text might read:

Scenario: Book reviews exist
Given that I wish to read a review of a book
And reviews exist
When I ask to see reviews
Then I can select a review from the list of reviews
And it will be displayed for me to read

Clearly, there is an alternate scenario wherein no reviews exist. This requires consideration by whoever specifies system behavior. In this case a decision is made to ask the potential reader to submit a review if s/he does read the book.

Scenario: Book reviews do not exist
Given that there are no reviews
When I ask to see reviews
Then I will be asked if I will submit a review when I've read the book

Creating realistic scenarios is a powerful approach that stimulates the conversation and analysis required to elaborate stories to the level of detail needed for coding and testing software.

8.2 BDD and Test Automation (K2)

8.2.1 Test Automation Process using BDD (K2)

LO-8.2.1: Summarize the process and benefits of using BDD for test automation (K2)

BDD's origins are in Test Driven Development (TDD). TDD is an approach to (or a philosophy about) software development that favors creating tests first and then writing the application code to meet the tests. Broadly, the process is:

- Create the tests
- Run tests to fail the tests
- Write code to pass the tests
- Re run the tests to hopefully pass
- Refactor the code to improve the design/ quality if needed

Dan North (originator of BDD) noticed that testing reveals system behavior and, from a customer's viewpoint, it is easier to think about (and define) desired system behavior than system tests. So, BDD provides a (fairly) natural language which can be understood and written by non-technical people (business stakeholders) to produce precise scenarios that can be used to create tests. Thus, in theory, trained and experienced clients can create and elaborate their own stories.

With this coding and testing heritage, BDD is designed to provide input to automated testing. That is, the BDD's structured scenarios can be parsed by test generating software to create code to test the application.

Automated testing is a very powerful (and, some would say, necessary) addition to the agile development approach. Agile development iterations produce ever larger increments of the end product (release). As each new software feature is added, it must be verified that the previous parts of the system still work as they should. This is called regression testing and, without automation, it becomes increasingly cumbersome as the product grows and changes.

BDD software tools include Cucumber (http://cukes.info/) and Jbehave (http://jbehave.org/).

9 Understand what is Doable (K3)

Timing	95 minutes
Terms	Last responsible moment, commitment, Feature Injection, planning workshop, estimation, Story Time pre-planning, relative estimation, velocity

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

9.1 Real Options (K1)

LO-9.1.1: Define the concept of last responsible moment (K1)
LO-9.1.2: Distinguish between options and commitments (K1)
LO-9.1.3: Describe the Feature Injection approach (K1)
LO-9.1.4: Recall the three steps in the Feature Injection process (K1)

9.2 Planning Workshops (K2)

LO-9.2.1: Know the levels of planning that are relevant to software development estimation (K1)
LO-9.2.2: Identify the elements required to conduct a planning workshop and explain why they are necessary (K2)
LO-9.2.3: Recall the purpose and positioning of Story Time pre-planning analysis (K1)

9.3 Relative Estimation (K3)

LO-9.3.1: Explain the reasoning behind the use of relative estimating and know where it is used in the agile process (K2)
LO-9.3.2: Describe how estimating is improved in agile development and identify the points at which estimating accuracy will be addressed (K3)
LO-9.3.3: Explain the concept and use of velocity on agile projects (K2)

9.1 Real Options (K1)

9.1.1 Introduction

Real Options Analysis (ROA) is a financial investment tool that encourages the consideration of alternatives or choices that will become available as details of the investment emerge. So, business cases are typically created by identifying costs and benefits and calculating the return on investment (ROI) using discounted cash flow (DCF) and internal rate of return (IRR) mechanisms to decide whether to commit money to the initiative as a whole.

ROA identifies points in the project where options to proceed in various ways (or not at all) will be available. This allows smaller financial commitments to be made up to the points at which those decisions must be made.

The definition of a Real Option is that it has value and will expire at some time. The principal is, don't commit to any course of action until you must or until you have enough information to make the decision with clarity. While this may seem self-evident, many people believe that "any decision is better than no decision". So, driven by a false belief that a decision should be made, they may commit before they have sufficient data or really need to. This will lead to waste when, later, emerging information demands a different course.

9.1.2 Last Responsible Moment (K1)

LO-9.1.1: Define the concept of last responsible moment (K1)
LO-9.1.2: Distinguish between options and commitments (K1)

ROA provides agile teams with a principle to support their planning activities. That is, do not commit to a plan (course of action) until enough information is available or circumstances dictate that the option will close. Once a decision is made, the option has been lost so keep options open for as long as possible.

One way to avoid waste is to wait until the Last Responsible Moment (LRM) from Lean Software Development. ROA says “Never commit early unless you know why”.

Commitments taken too early are more prone to revision which creates waste. So, the real options approach fits well with the agile/ lean philosophies of working just-in-time to eradicate waste.

9.1.3 Feature Injection (K1)

LO-9.1.3: Describe the Feature Injection approach (K1)
LO-9.1.4: Recall the three steps in the Feature Injection process (K1)

If analysis is to be performed just-in-time, it will need to be efficient as well as timely.

Feature injection is an approach which begins (as all business analysis should) with business value.

Surprisingly, many projects are given requirements that are actually solutions (“we need a database of off-shore suppliers”) and assume that there is a value (“the business wouldn’t have asked for it if it didn’t have value”) and start looking for inputs to enable the solution (“what supplier data will we need to put into the system?”). This has not been considered from the business value perspective.

Kent McDonald has described Feature Injection as a three-step process.

1. Identify the value (to the business)
2. Inject the features (that will provide the value identified in step 1)
3. Seek examples (scenarios that illustrate and contradict)

Feature Injection has three steps:

1. Identify the business value

Stakeholders frequently identify functions (or non-functions such as speed) that they want but are often unable to say why that requirement is valuable to the business. This is beyond the simple 3rd clause in the user story which often says why that particular user wants the function, not why the business would want it. Feature Injection focusses the search for business requirements with verifiable value that will be returned if we deliver a given requirement.

Earlier on this course, we looked at the third principle - Analyze to determine what is valuable. It contained

- Business Value Definition,
- Kano Analysis, and
- the Purpose Alignment model which are all candidate tools in this context.

Additional useful tools include

- the Boston Business Maturity Model (Boston Box),
- ROI models,
- SMART,
- Entity Relationship or Object Modelling,
- Statistical modelling,
- Enterprise Capability Modelling and
- Value Stream Mapping (these last two in Chapter 5- See the Whole) and many more.

2. Inject the features

Having made a compelling case for a requirement, we begin to Analyze the features that must exist in order to deliver that requirement. This is a good team task and fits the agile approach to identifying more and more detail (elaboration) as we approach the build effort.

3. Look for examples

The features will ensure that the required outcome will be delivered. For those of you who've created Use Case descriptions, this is known as the Main (or sometimes, Happy Day) Scenario. Step 3 seeks exceptions, or variants, to this path. Perhaps the outcome is to deliver a product to the customer's home as quickly as possible. The Main Path does that. But, what if that means the product arrives at three in the morning? If this may not be a beneficial outcome, we must consider how the features will handle this exception. This is scenario analysis for which we might use the BDD scenarios described in Chapter 8 – Get Real with Examples.

9.2 Planning Workshops (K2)

LO-9.2.1: Know the levels of planning that are relevant to software development estimation (K1)
LO-9.2.2: Identify the elements required to conduct a planning workshop and explain why they are necessary (K2)
LO-9.2.3: Recall the purpose and positioning of Story Time pre-planning analysis (K1)

Agile teams plan frequently because they expect, and react to, continual change as the project progresses.

Planning workshops happen when the team needs to produce a reliable (as possible) commitment to the delivery of a product or product increment. Most commonly, this happens at release (product) and iteration (increment) levels.

9.2.1 Release Planning (K1)

Release planning will happen when details of the requirements are not available or at a high level but there is a need to produce a reasonable expectation of what the product will contain when it is delivered at the end of this release. It's worth noting that delivery means meeting the definition of done - "almost there" doesn't count as delivered.

Discussion will be based on estimates (typically, story points) and issues such as priorities and MVP which are driven by business goals.

9.2.2 Iteration Planning (K1)

In iteration planning, the business representative/ product owner and development team will agree which deliverables will be completed during this iteration. This means that enough must be known

about the backlog items to understand what they are, accurately estimate the time to produce them and break down the work into tasks for team members to complete.

To complete the iteration planning within an acceptable timeframe (probably two hours) the backlog will need

- to have been groomed in advance so that the candidate stories can be accomplished in an iteration,
- have acceptance criteria defined and
- are prioritized.

This may be done in one or more pre-planning sessions (sometimes called Story Times) where the stories are discussed and elaborated.

9.3 Relative Estimation (K3)

LO-9.3.1: Explain the reasoning behind the use of relative estimating and know where it is used in the agile process (K2)
LO-9.3.2: Describe how estimating is improved in agile development and identify the points at which estimating accuracy will be addressed (K3)
LO-9.3.3: Explain the concept and use of velocity on agile projects (K2)

Estimating is fundamental to planning. Without an assessment of the time and effort required to accomplish a task it is difficult to commit to product delivery with any certainty.

Traditional projects estimate during the project planning phase, before project work begins. The start of a project is a challenging time to estimate because, at that time, least is known about the project deliverable (product) size, scope and complexity.

Thus, these early estimates are known as “order of magnitude” estimates and, despite indications that variance is likely to be, at the very least, plus or minus 50% (usually plus), commitments are made and the estimate is treated as a deadline.

Early estimating is also unreliable in agile projects but that unreliability is acknowledged and accepted because further estimates will be made regularly throughout the process and feedback on accuracy will be quick, highly visible and frequent.

People find it easier to compare size than to quantify size. It’s hard to judge how heavy a mouse is or how heavy an elephant is but it’s easy to judge which is heavier.

So, in agile approaches, a common feature of estimating is relative size. Relative estimates may be made at any stage in the agile development lifecycle.

This is particularly useful in early planning because detailed requirement definitions aren’t available; the stories typically describe larger chunks of functionality. It’s also likely that little is known about the team’s productivity. It will be hard to estimate how much time it will take to build something when we don’t know much about what we’re building and we don’t know how quickly we can build things.

Understandably, developers are reluctant to estimate time without having detailed knowledge because they believe that their estimates will be used as targets that they will be judged against.

So, instead of trying to guess how long something (not fully understood) may take to complete, agile methods make judgements about how big something is relative to other things. Size is represented by units that are called Story Points.

Story Points

The process starts with the team allocating a value to a requirement. This value is agreed as representing the size of the requirement. Other requirements are then considered in terms of their size relative to the first requirement. Generally, a team chooses the smallest story first and then compares other stories to it.

Story points are used to make broad estimates of items in the product backlog. No deep analysis, philosophical discussion or soul searching is required. Some of the items in the backlog may never reach a development iteration.

It's common for teams to allocate story points using numbers in the Fibonacci sequence. Fibonacci adds the last two numbers in the sequence to produce the next.

Hence 1, 2 added together make 3 giving 1, 2, 3. Now 2 and 3 are added to produce

1, 2, 3, 5. The next numbers will be 8, 13, 21, 34 and so on.

The reasoning is that Fibonacci numbers' increasingly large spread discourages the estimators from fooling with figures below the decimal point while arguing minutely detailed differences between the size of one user story and another.

Tasks

A user story will take development team effort to produce as a function or feature in the software. During iteration planning, an estimate of what's doable within the timeframe must be made. This means that a story must be decomposed into the tasks required to deliver the story.

Team members agree to take responsibility for their chosen tasks and for completing them on time. To do so, each team member will need to estimate the effort required to complete each of her/ his tasks.

Planning Poker

Planning poker is a commonly used agile technique which adds a team element to estimating.

A moderator selects a story, epic or theme and invites team members to provide an estimate. The team can discuss the story and seek more details (which may or may not be available). Each team member will then write down her/ his story point estimate but will not reveal it.

Once the moderator confirms that everyone has written an estimate, the team is asked to show their "hand" (like poker). Discussion of different estimates then ensues with different players explaining the reasons for their score. Clearly, the aim is to reach consensus.

Agile planning poker cards can be purchased but paper or Post It™ notes will suffice.

Velocity

Velocity is calculated as the number of story points delivered by a team at the end of an iteration, or iterations to date, on a project.

Velocity brings reality to estimating. For example, a team estimates three stories at eight points each. They predict that all three stories can be completed in the first iteration. If their estimate had been correct and they'd delivered all three stories, their velocity would have been 24. However, at the end of the iteration only two stories met the definition of done so their velocity was 16.

At the next iteration planning session there are three stories; one is eight points, the second is 13 points and the third is five points. The team will reflect on their velocity and estimate accordingly. They may decide that they are working more effectively now and can complete the first and second stories (total 21 points). Alternatively, they may decide that they can do only a little better and agree to delivering stories two and three (18 points).

Whatever their decision, they will find out how well they estimated at the end of iteration two. This is a major benefit of the agile approach; feedback is frequent, fast and unavoidable.

Early iterations will be harder to estimate accurately because there is limited data to calculate velocity. Typically, by the third or fourth iteration, a reliable average or mean will be relevant to be calculated.

10 Stimulate Collaboration and Continuous Improvement (K3)

Timing	85 minutes
Terms	Retrospective, collaborative games

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

10.1 Introduction
No learning objectives
10.2 Retrospectives (K3)
LO-10.2.1: Identify the purpose and deliverables of a retrospective (K1)
LO-10.2.2: Engage in a retrospective (K3)
LO-10.2.3: Create a persuasive case for including agile retrospectives in projects that are already short of time and resources (K3)
LO-10.2.4: Recall the tools, techniques and skills that a BA may use to bring value to agile retrospectives (K1)
10.3 Collaborative games (K2)
LO-10.3.1: Summarize the benefits to be derived from collaborative games (K2)
LO-10.3.2: Recognize the philosophy behind collaborative games (K1)
LO-10.3.3: Participate in a collaborative game (K2)

10.1 Introduction

In order to receive early feedback, so that the product can be fast and easily adapted to the customer needs, an environment ready for communication and feedback is desired. This will lead to continuous improvement, and leads to growing value towards the customer at a faster pace.

This kind of environment can be nurtured with different techniques, like retrospectives and collaborative games.

10.2 Retrospectives (K3)

10.2.1 Purpose of a Retrospective (K3)

LO-10.2.1: Identify the purpose and deliverables of a retrospective (K1)
LO-10.2.2: Engage in a retrospective (K3)
LO-10.2.3: Create a persuasive case for including agile retrospectives in projects that are already short of time and resources (K3)

People (can) learn from the past and a retrospective, as the name implies, is a team meeting (or workshop) specifically scheduled to look back on the team's performance and reflect on how it might be maintained or improved. The retrospective is a formal implementation of the continuous improvement principle that is a cornerstone of agile and lean thinking.

Retrospectives are typically held at the end of iterations and projects. The team discusses what worked well and what could work better. Further discussion focusses on how to habituate successful working methods and improve on areas of concern.

Scrum (www.scrumalliance.org) recommends a maximum of one hour's retrospection for each week of the sprint (iteration). To use this time effectively, retrospectives require structure and facilitation.

Retrospectives are usually structured as:

- What did we do well? It's good to start with the positive lessons and success is worth continuing. Sometimes successes are not repeated because they aren't formally recognized and actively reinforced.
- What did we do not so well? How issues arose and what we didn't do that we could have done.
- What lessons have we learnt? What we should take forward to subsequent iterations and/ or projects. Identify those items that will return most benefit.
- What will we do to capitalize on what we've learnt? Lessons learnt will have no meaning if actions are not taken to (re)enforce them. This requires a plan of action points. Action points define the action, who is responsible and when it will be completed.

Retrospectives require facilitation so give participants tasks to perform with clear deliverables and timebox their efforts.

Retrospectives are frequent in the agile lifecycle (every one to four weeks depending on sprint duration) which means there are many opportunities to improve. However, this frequency carries a danger that the team just "goes through the motions" because they become bored with the same, repetitive procedure. So, seek to vary the approach, tasks and techniques.

A BA should have an array of facilitation tools at their disposal and use them to vary one retrospective from another.

10.2.2 Tools, Techniques and Skills (K1)

LO-10.2.4: Recall the tools, techniques and skills that a BA may use to bring value to agile retrospectives (K1)

Facilitation

Retrospectives provide an ideal opportunity to use the facilitation skills of a BA for added value. Candidate tools and techniques for enabling effective retrospectives include:

Brain writing

- Affinity diagramming
- Nominal group technique
- Edward de Bono's Six Thinking Hats and PMI (plus, minus, interesting)
- Fishbone diagramming
- Mind maps
- Post It™ note exercises

Many more tools will be useful in this context.

Additional retrospective ideas and templates can be found at the Agile Retrospective Resource Wiki (www.retrospectivewiki.org)

The Five Steps Process

In their book Agile Retrospectives: Making Good Teams Great (Pragmatic Bookshelf - 2006) Esther Derby and Diane Larsen suggest a five step approach to successful retrospectives.

Step 1: Set the Stage
As any BA should know, it's important to setup a workshop with an agenda that sets the objective(s) to be achieved. It is equally important to ensure that all participants engage and contribute to the meeting.
At the start, have each team member say what s/he is expecting to gain from the retrospective and what s/he will bring to the retrospective. Possible techniques include a Post It™ exercise, brain writing and mind mapping.

Step 2: Gather Data
In this context data will be both objective (such as the team's velocity) and subjective.
How each team member feels about their experience during the iteration is important and will highlight issues that created a sense of achievement, frustration, demotivation and so on.

Mind mapping, Post It™ notes and brain writing are candidate techniques for step 2.

Step 3: Generate Insights
In general, statistical evidence will highlight achievements and problems but the subjective information will often provide greater insight to the sources of successes and failures. PMI may be useful here.
Grouping issues (perhaps using affinity or Fishbone Techniques) may help find patterns and trends to address rather than focusing purely on individual items.

Step 4: Decide What to Do
What, who and when are useful headings in this step; what will we address during subsequent iterations, who will be responsible and when will that person complete the chosen task.

Finding the highest priority actions is commonly based on what will return the greatest benefits and what is possible within a reasonable timescale (normally the next iteration). A simple grid (below) may be of help.

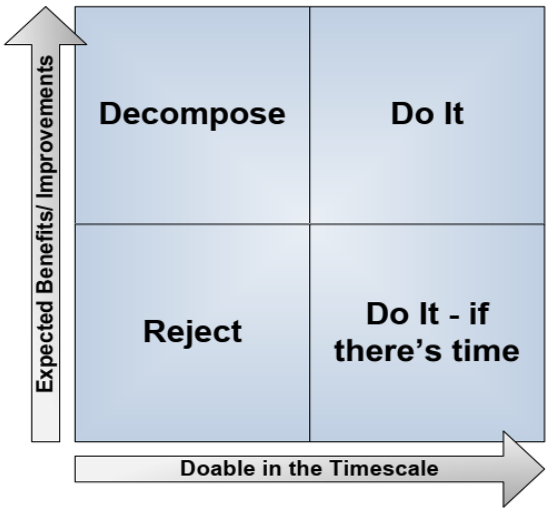


Figure 20 – Assessing Retrospective Actions

Step 5: Close the Retrospective

Confirm observations made and decisions taken. Provide the opportunity to clarify and ask questions. Ensure there is complete agreement and, if not, resolve any disagreements.
Capture the work completed. This may be photographs or the notes produced during the retrospective. It demonstrates the value that you place on the team's effort and will be useful for future retrospectives.

From time to time, it is worth considering a (brief) retrospective of the retrospective.

That is, what went well in this retrospective, what was not helpful and what can we take forward to make future retrospectives more valuable.

And finally, voice your own appreciation of the team and thank them for their effort.

10.3 Collaborative Games (K2)

LO-10.3.1: Summarize the benefits to be derived from collaborative games (K2)
LO-10.3.2: Recognize the philosophy behind collaborative games (K1)
LO-10.3.3: Participate in a collaborative game (K2)

Collaborative games are used in a variety of contexts from marketing, strategy planning and business risk analysis to psychotherapy, scientific problem solving and education. The power of collaboration is well documented and is predicated on a belief (from Gestalt psychology) that "the whole is greater than the sum of its parts".

Games seek to replicate a common human experience wherein a person comes to a new and, possibly, sudden useful understanding by receiving input from others and/ or having an apparently unrelated experience that creates a "Eureka!" moment (attributed to Archimedes, the ancient Greek scholar).

Collaborative games apply metaphors or parallels to the actual subject under consideration. A collaborative game is "like" real life but in a safe environment and can help to build common understanding, team spirit, broader insights and relevant new, possibly radical, ideas and solutions.

Collaborative games have a specific objective and apply rules to the game-players.

This normally requires facilitation by a game master or referee who lays out the rules and maintains focus on the outcome. At the end of the game the team assesses the lessons learnt and insights gained to identify valuable actions or decisions to be taken.

11 Avoid Waste (K2)

Timing	15 minutes
Terms	Lean, waste, Lightweight documentation

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

11.1 Eliminating waste (K2)
LO-11.1: Articulate the Lean philosophy and give examples (K2)

11.1 Eliminating Waste (K2)

11.1.1 Introduction

Eliminating waste is at the heart of the Lean philosophy. Any activity that the customer would not expect to pay for in the product price should be considered wasteful. Waste increases cost and reduces profit. It is anything that doesn't add value to the end-product.

Clearly, there are exceptions (perhaps regulatory compliance or audit requirements) but Lean organizations seek to minimize the effort expended on such issues.

11.1.2 Lean Philosophy (K2)

LO-11.1.1: Articulate the Lean philosophy and give examples (K2)

This philosophy sits well in the agile domain which values "*working software over comprehensive documentation*" and techniques in the preceding modules of this course adhere to this approach. Some guidelines offered include:

- Co-locate with the stakeholders and team to avoid time wasted by walking about
- Reduce reliance on emails, SMS, IM and phone – face to face conversation is likely to be less ambiguous, more reliable and have added benefits such as building rapport and generating ideas
- Obtain, Analyze and specify requirements with the same models
- Keep models as simple as possible – don't use notation or add details that aren't useful to a stakeholder
- Pay attention to excellence - fixing defects is very expensive
- Take responsibility for completion (to the definition of done) so that incomplete delivery doesn't impact downstream activities
- Make decisions just-in-time (last responsible moment from Real Options) to avoid unnecessary refactoring (revision) work

Lean thinking should be a permanent state of mind for those participating in agile development teams and business analysts should (with their training and experience of process analysis and redesign) be particularly well equipped to identify waste and ways to eradicate it.

12 Review (NA)

Timing	NA minutes
Terms	Limitations, role, tools

Learning objectives

The following objectives identify what you will be able to do after the completion of each module.

12.1 Beyond Software Development (NA)

LO-12.1.1: Understand the limitations of the agile context in the role of the business analyst (NA)

LO-12.1.2: Identify what course content should be considered for inclusion in work-place practices and why (NA)

12.1 Beyond Software Development (NA)

12.1.1 Introduction

Products and customers are entities that go outside of the Software Development industry, and exist basically in all industries.
Such as the production team and stakeholders.

No matter the methodology used for a project, the BA role is an important one.

This is given by the fact that every customer will have in the beginning goals, for which he desires the product. The BA has the responsibility to transform the goal into needs that the product must meet.

12.1.2 Limitation of the BA role and practices in Agile Context (NA)

LO-12.1.1: Understand the limitations of the agile context in the role of the business analyst (NA)

LO-12.1.2: Identify what course content should be considered for inclusion in work-place practices and why (NA)

The limitations of the BA role in an agile context comes from two sources:

- Definition of the role in the company
- Definition of the role in the team

The definition of the BA role in a company can span from clarifying business and stakeholders needs, connection to the market, keeping the pace with technology, to clarifying the customer needs, down to clarifying the requirements and supporting the team in handling each increment.

From the team point of view, the BA role and responsibilities can be shared among the team members, making the BA role apparently not needed.

In both situations, when trying to create a role that spans from business to team and/or vice-versa (as a Product Owner would partially be), there will appear tensions and responsibilities with objectives that can be contradictory.

In order to overcome such situations, the person that takes up the BA role in an agile environment, or the one that picks up a BA responsibility (or more than one) should have clear what is the outcome of that responsibility, and which tools s/he should use, and in which phase of the process.

Basic knowledge that should be taken from this course, by anyone who likes to stay connected to the bigger picture, would be:

- The extended traditional SDLCs
- The extended Agile SDLC
- The scope of the Business Analysis
- The three levels of business analysis.

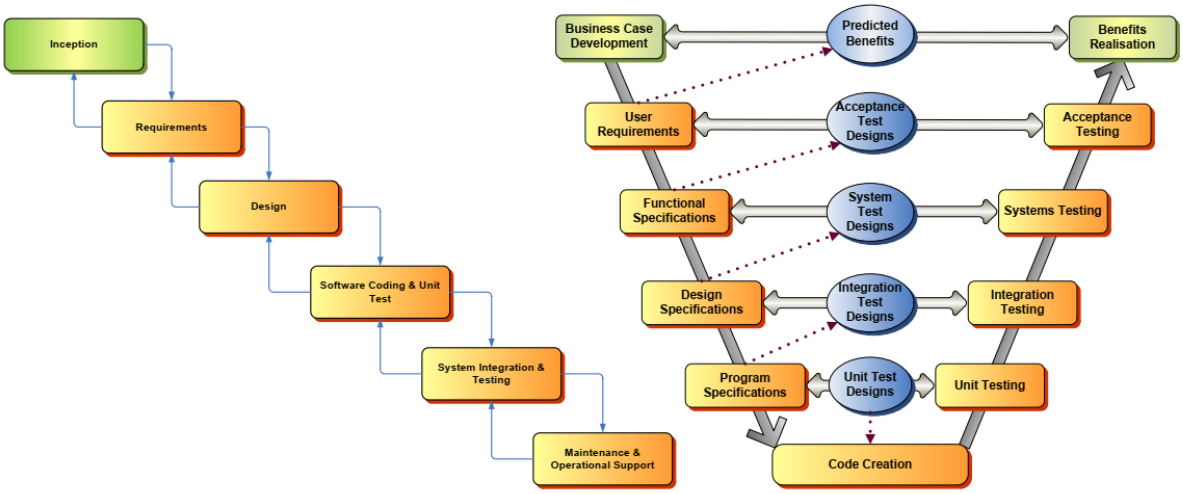


Figure 21 – Extended Traditional SDLCs

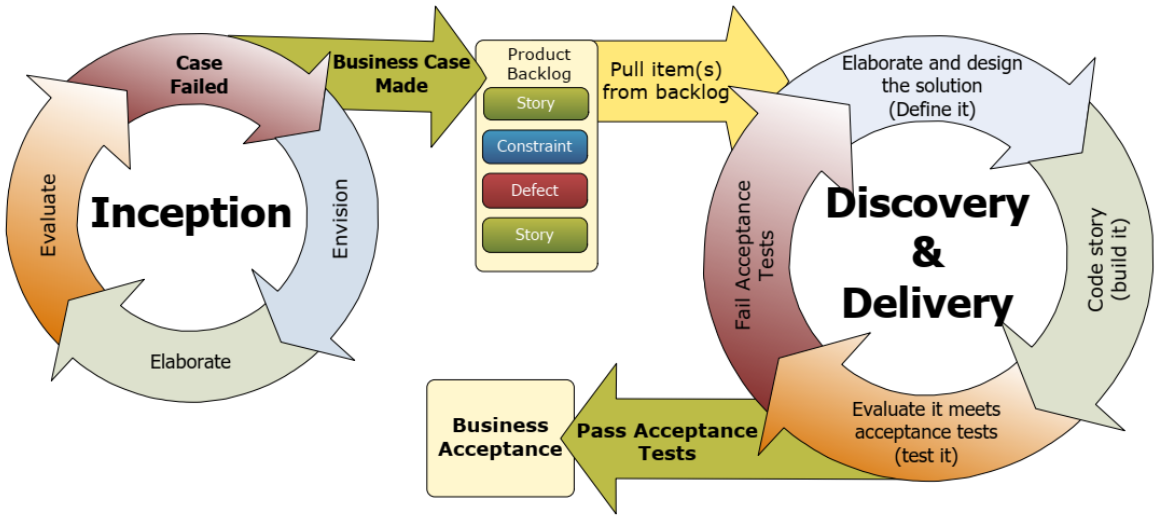


Figure 22 – Extended Agile SDLC

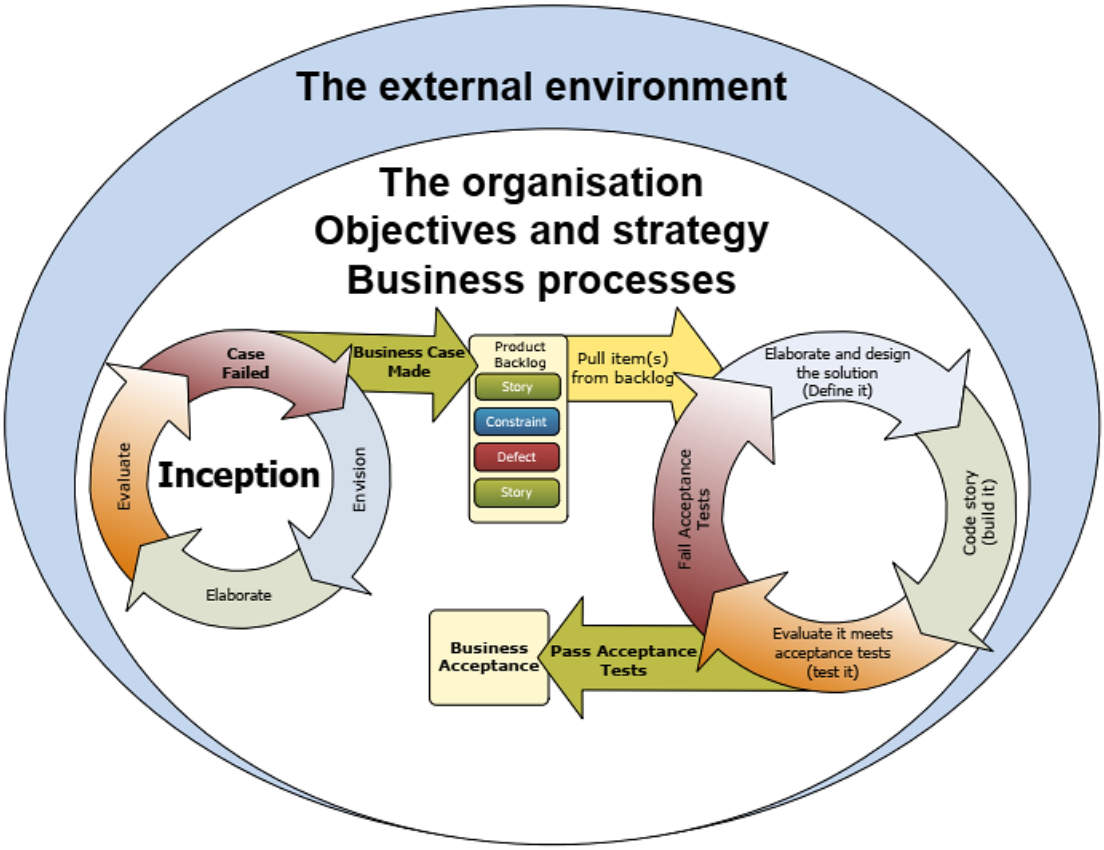


Figure 23 – The Scope of Business Analysis