# LO3.1 - Explain methods of problem identification

Software projects are always intended to solve some business problem. They will address some need or want of their intended users.

However, the initial vision for the project is often not the final direction of the project. As the project progresses and real business requirements become better known, the direction or intent of the project may change. While planning for and adapting quickly to change are core parts of the agile process, some initial modeling and research is necessary before proceeding to in-depth requirements gathering methods as described in this learning outcome. Doing initial requirements modeling, or "Envisioning", can help to reduce the cost of a project by making the team or organization aware of potential problems earlier on in the project process.

Agile Modeling, by Scott W. Ambler, describes three essential parts of the project initiation process which can help to solidify your early understanding of a client's real needs.

1. **Usage model** – explore how users will work with the system and what each category of users’ needs/wants from the system. At this stage it shouldn’t be a complete picture.You can use techniques do determine an overview for needed functionally by:
   * Team/client brainstorming,
   * Client questionnaires,
   * Or shadowing.

Doing so can help provide the team with initial estimates of scope, time and cost.

1. **Domain model** – the domain model of a project captures the highest level business entities involved in the project and the relationships between them. The domain model captures terminology used by the client to describe business processes and procedures.

Use the same words as your client to describe people and processes your software will support. This can help facilitate faster, higher quality communication. The set of terminology clients’ use is often called domain language. Creating a dictionary for the domain language can be helpful in creating more meaningful dialog with the client.

1. **User interface model** – clients can have an initial vision for what their project will entail, ideas about workflow, and how the software will appear (interface).Sketching initial visions and talking about them with your client/stakeholders can be helpful in the initial scope and planning stages. Working through a rough user interface model (low-fidelity prototype) can be useful in guiding further requirement gathering.

# LO3.2 - Identify stakeholders, their interests, involvement and impact on a project

Stakeholders are those people or organizations who will be affected, positively or negatively, by a project. They are often described as having at least one of the three I's:

* **Interest** – end users, customers, or anyone else who may receive some benefit from the outcome of the project.
* **Involvement** – developers, managers, client contacts and support staff; anyone who will be directly or indirectly involved in the process of constructing the project.
* **Impact** – managers, clients, competitors; anyone whose decisions could change the direction or scope of the project.

Different stakeholders may be more interested in one part of the project than another. For instance:

* **Outputs** – stakeholders that are most interested in the outcome of the project (Example: end users)
* **Inputs** – stakeholders that are interested in inputs (example: sponsor that is only willing to invest so much money)
* **Process** – stakeholders that are most interested in the project processes (example: project team members)

It is important that the "right" people be in contact - while the salespeople and executives often involved in project initiation may believe that they have complete information about their needs, it is important to go beyond those who make themselves immediately available to find a more complete set of *stakeholders* for a project.

There are obvious stakeholders: end users, client contacts, the development team and managers on both sides. However, not every stakeholder is going to be obvious. Brainstorming as a team and with known project stakeholders can be an effective tool for identifying non-obvious stakeholders.

## Brainstorming

Brainstorming is a simple technique which works best when a few simple rules are followed:

* Provide a basic overview of the problem to be solved.
* Allow a variety of people to contribute or combine ideas as this is how the greatest benefits often result.
* Allow people the opportunity to do some pre-thinking on the problem.
* If the brainstorming group members are not familiar with each other you may need to do some kind of ice-breaker.
* All ideas are written down using: Post-it notes, whiteboards, paper, etc.
* Express ideas in the participant’s own words. Be careful not to change someone’s ideas.
* Do not allow criticism of anyone’s ideas.
* Do not permit one person or group of people to dominate the brainstorming session.

It is important to ensure requirements coming from different stakeholder groups are consistent and do not conflict with one another. It is important to look for areas of commonality. Sometimes requirements will be agreed upon by the whole team while other times a project champion will decide which requirements make the final cut.

Once you have identified stakeholders’ and their potential impact you can then work to develop tactics to get the most effective support possible for your initiative and reduce any obstacles to successful implementation of your program.

# LO3.3 - Describe Requirements

A requirement is a statement that identifies a necessary attribute, capability, behaviour, characteristic, or quality of the software. Requirements specify what the software will do, as well as differences in how that activity or behaviour should be implemented for different actors or stakeholders.

Requirements are the life of any software project. They describe the end goal of the project - they are the definition of completion for the project itself; when all of the client's requirements have been satisfied, the construction of the project is complete. Requirements can be broadly divided into two major categories:

## Functional / Behavioural requirements

* These describe the functionality or behaviour that the software must provide. They include what activities a certain actor must be able to perform, what business rules must be followed while performing those activities, how the end users will interact with the system.
* Example: “Employees must be able to add information about a vehicle to their parking spot”.
* It describes some activity that an actor (employee) must be able to perform using the functioning system.

## Non-Functional/Non-behavioural requirements

* These requirements describe the “how” side of the system. They often place constrains on the design or implementation of the system, imposed due to legal, reporting or standards requirements on the business.
* Examples include:
  + Technical features (what programming language to use, what server software should be supported)
  + Availability (uptime or accessibility requirements)
  + Security (protocols around information security and disclosure)
  + Performance (minimum and maximum response times, throughput)
  + Interoperability (compatibility with other systems, standards compliance)
  + Reliability
* You need to remember to specifically request information from the client about what types of limitations such as those above will be placed on the project.

## Characteristics of a good requirement

The following are characteristics of good requirements:

* Correct – accurate description of the functionality to be delivered.
* Feasible – It must be possible to implement
* Necessary – The requirement must be needed for the system.
* Unambiguous/concise/explicit – Can only be interpreted in one way and should relay the exact message.
* Verifiable – it must be possible to test/verify that the software meets the requirement.
* Complete – contains enough detail to convey what the new system must do under all circumstances.
* Consistent – use consistent terminology, preferably domain language. Use a logical consistent path with no missing data or elements.

# LO3.4 - Practice common methods and activities for discovering requirements

The list of stakeholders created as part of the earlier brainstorming process can be used to create a list of user roles, or actors. Identifying actors is an essential part of the process of gathering project requirements.

**Actor** – some entity which performs an activity using the system being created, or interacts with it in some way.

Actors are a form of domain language. Different actors will use and be affected by the software in different ways. They will have different goals and responsibilities, and each will have to perform different tasks in order to accomplish those goals.  
  
Actors are not always people and can be other software systems, servers, or services.

## Gathering Requirements

Not every requirement is the same; different requirements will be discovered using different techniques, and often you will discover more details about existing requirements using different techniques.

Gathering requirements is an ongoing, iterative process.

### Techniques:

**Interviews**

* Gather a range of actors (user roles) and ask them questions about what their needs for the software include. Try to determine how the users currently solve problems, what information they need and what tools they are currently using.
* Try to contact actual end users, but sometimes proxies (Manager/supervisor) may be acceptable.
* A con is that it is difficult for users to remember specific details when removed from the context of performing the task.

**Questionnaires**

* Questionnaires are usually created ahead of time and filled out as time allows.
* Often not suitable for discovering new requirements, but can be well suited for clarifying existing ones.
* One-way form of communication.

In the case of both interviews and questionnaires, care should be taken to ensure that questions are:

* Open ended - open to exposition and expansion, not simply yes or no questions. In most cases, especially early on in the project, it is a better idea to allow stakeholders to let their ideas flow out freely, even if the topic may stray a bit. More information is better.
* Context free - questions shouldn't imply that one answer is the correct one. Stakeholders should be free to answer as they see fit, without pressure from the project team.

**Observation or Shadowing**

* A project team member “rides along” with end users to observe their work and document the current way of doing things.
* A rich communication channel
* Can often discover hidden details of how a process works.
* Useful after delivery of initial versions of software. Feedback can be gathered quickly.

**Workshops or brainstorming**

* Use user interface mockups on a display to do a UI walkthrough and use it to drive the requirements discovery process with end users.
* It helps some clients/end users feel more comfortable when being walked through the process rather than just speaking about it.
* Build a visual prototype of the software as your users are talking about their workflow.  
  As you build the prototype, ask:
  + What will the user most likely want to do next?
  + What mistakes might the user make here?
  + What might confuse the user here?
  + What additional information can the user provide, or what would they require?
* Use models and visuals wherever possible.
* Use this as an opportunity to expand your knowledge of the client’s domain language.

**Blitz planning session**

* Resembles a standup meeting or brainstorming.
* Project team and stakeholders gather around a table. Each person needs a set of index cards and a marker
* Each person writes a feature, requirement, or fact about the project, says it aloud and puts the card in the center of the table.
* Has a 15-20 min timebox
* Usually the Scrum Master or the Product owner will gather up the cards and read them aloud.
* The client will give a rough priority of each card

An example of blitz planning in action can be found here: <http://web.archive.org/web/20140930141155/http://devblog.point2.com/2009/09/08/blitz-planning-at-point2/>

See the requirements gathering section of AgileModeling.com for additional methods for gathering requirements. [http://agilemodeling.com/essays/agileRequirements.htm#TechniquesForElicitingRequirement](http://agilemodeling.com/essays/agileRequirements.htm" \l "TechniquesForElicitingRequirement)

In all requirements gathering techniques, ensure that you thoroughly document proceedings and any requirements that are discovered. If possible, record the sessions, but ensure that you ask permission of participants first. While informal documentation of requirements, using jot notes, meeting minutes or index cards is a good first start, you will learn more formal requirements documentation guidelines in a future learning outcome.

# LO3.5 - Describe methods of identifying risk

**Risks** – are potential factors that may impact the successful completion of the project. The primary component of risk is uncertainty. A risk which is certain to occur is not a risk; it is a fact. Any event or factor that can affect the outcome of the project (positively or negatively), but for which the likelihood or impact is uncertain can be considered a risk.

It is important to identify as many risks to your project as possible, so that you are prepared to deal with any situation those risks could bring. Having up-front planning is important to give teams context when a problem does occur. Thinking about possible risks ahead of time can lead to the team making design or planning decisions which can avoid those risks altogether.

A list of risks are included as part of the project charter but a risk that has no chance of occurring, or cannot be controlled is not worth including as part of project planning.

SWOT is used for identifying internal and external factors that are both positive and negative to achieving the project’s objective.

**S - Strengths**

* Includes skills, assets and information the project team has which make the project more likely to succeed.
* Usually positive but it’s possible that the team does not have a complete knowledge of their strengths.

Examples:

* team members who have already completed projects similar to the one being analyzed,
* a lack of competitors in the project's functional area
* a client who is highly available and motivated.

**W - Weaknesses**

* factors that could place the project or team at a disadvantage compared to competitors.

Examples

* competition for skilled project team members within the company,
* compensation issues
* team has limited experience using the software systems the customer requires.

**O - Opportunities**

* Factors outside of the project team, client or company’s control
* Gives an advantage to the likelihood of the success for the project.
* Uncertainty exists in whether or not the opportunities will end up being beneficial to the project.

Examples

* a favourable economic climate
* a strong demand for the type of project the team is creating.

**T - Threats**

* factors outside of the control of the team, company or client.
* Can damage the likelihood of a successful project.
* These are frequently the targets of the risk management and control techniques.
* They are often more difficult to avoid and control than the internal risks categorized under weaknesses.

Examples

* the possibility of a competitor completing a similar project more quickly
* a rapidly changing technological environment
* weak economic conditions.

<https://www.youtube.com/watch?v=RQ44D_DPuaA> (3 little pig swot analysis)

A SWOT analysis is often performed using a grid similar to the following. Prior to entering into the risk management phase, the project team and other project stakeholders can brainstorm risks and assets and place them into the SWOT categories. Keep in mind that any project-affecting factor which has an uncertain chance of occurring, or for which the extent of impact is uncertain, can be considered a risk. In the end, risks appearing under the Harmful column are more likely to be considered as part of risk mitigation planning.

|  |  |  |
| --- | --- | --- |
|  | **Helpful** | **Harmful** |
| **Internal** | Strengths | Weaknesses |
| **External** | Opportunities | Threats |

# LO3.6 - Describe Risk Mitigation Planning and Quality Assurance

**Mitigation** – the action of lessening in severity or intensity.

**Risk Mitigation Plan**

* also called a Risk Response plan
* indicates how specific risks will be dealt with if they arise
* what action steps are required to deal with the risk
* this should lessen the impact of the risk on the overall project
* gives team members a strategy and clear sense of the actions they are expected to take
* must continue to be reviewed and managed on a regular basis as the project develops.

Here are some examples of common project risks:

* Customer review and feedback cycle too slow. – *time delays*
* Unexpected budget cuts. – *can’t implement full functionality at original cost*
* Unclear roles and responsibilities. - *time delays, work not getting done.*
* Stakeholder input is not sought, or their needs are not properly understood.
* Stakeholders changing requirements after the project has started.
* Stakeholders’ adding new requirements after the project has started.

Some companies keep track of risks that have occurred in the past – in a “Risk Inventory”. New projects refer to this when they are creating their own Risk Mitigation Plan.

**Risk Mitigation**

When working on any project there are three main areas of risk mitigation that must be considered. Those main areas are:

* 1. **Identification** - an orderly attempt to identify potential risks to the project plan. When risks are identified, a project manager (iterative) or a team (agile) can work towards avoiding them when possible and controlling them when required. (SWOT)
  2. **Quantification –** the activity of assessing and evaluating the effect a risk could have on the project.During risk quantification the concerns are risk probability and risk impact.

The effect a r

* **Risk Probability** – the likelihood of the risk occurring.  
  Rate each risk probability either on a scale of Improbably to Highly likely or it could be on a percentage scale 1%-99%
* **Risk Impact** – the extent of the damage resulting from the risk.  
  This can also be rated on a scale from negligible to Catastrophic. You could also determine estimated dollar values.

To determine how important a risk is one will need to consider the combination of probability and impact. It’s helpful to do this by using a Risk Management table.

**Mitigation**

**Risk mitigation** – reduces the likelihood of the event or risk from occurring. A risk mitigation plan is used to manage it.

Usually A Risk Mitigation Plan includes a Risk Management table.

A Risk Mitigation Plan or a Risk Response Plan indicates how specific risks will be handled if they arise. The Plan outlines what actions are required to deal with the risk. The goal of this plan is to lessen the impact of the risk on the overall project. The plan will outline each team member's expected actions should a risk occur. The risk plan should be reviewed and updated throughout the life of the project.

Usually A Risk Mitigation Plan includes a

**Risk management table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk Event** | **Likelihood**  **(1 = low 10= high)** | **Severity (1 = low 10= high)** | **RPN** | **Mitigation** | **Owner** |
| **Computer is not ready in time for project launch** | 4 | 8 | (32) 2 | Order computer early.  Track order status.  Confirm delivery. | AL |
| **Website not ready in time for project launch** | 4 | 9 | (36) 1 | Be ready 3 weeks prior to launch. Use a simple version. | SM |
| **Banking facilities not ready** | 2 | 10 | (20) 3 | Order immediately. | JT |
| **Lose a group member** | 1 | 10 | (10) 4 | Keep communication open. Create benefits and a work environment so that group member’s don’t want to leave |  |

**Risk Event** = all identified risks that are well detailed.

**Likelihood** = probability of the risk occurring. Often an estimate by the team.

**Severity** = how serious the overall effects are on the triple constraints (time/cost/scope)

**RPN** = (Risk Priority Number) – Calculated by Likelihood \* Severity. The larger the RPN the more serious and likely the risk is to occur.

**Mitigation** = The mitigation plan or steps for each risk item.

**Owner =** Person responsible for handling/monitoring the mitigation **OR** accepting the risk.

Once each RPN is calculated the table can be sorted by which risks have a higher RPN and are more likely to occur. So from this calculation you can tell that Risks that have high impact (Severity) with low probabilities and risks that have low impact with high probabilities are less of a concern. Risks with high impact and a high probability of occurring are usually the risks a project group will need to concern themselves with.

**Quality Assurance** – determining whether products meet or exceed customer’s expectations.

Typically a series of technical reviews are conducted during the design, and implementation phases to help assess the software quality.

**Technical Review**

A technical review is a meeting conducted by members of the software team. Before the meeting each person involved is provided with a copy of the work that has been completed. The team members have the opportunity to look for any mistakes, oversights, or discrepancies. The purpose of the meeting is to mention all problems with the product so that it can be revised. At the end of the meeting, the team determines what further actions are required by the person(s) whose work is being reviewed before the work is finalized

Types of technical reviews:

Design Review

Code review

# LO3.7 - Explain the importance of planning for change

Agile teams and agile team members must try to minimize the size and impact of the assumptions they make.

The greatest risk to any software project is Building the wrong software. The likelihood of the risk occurring increases when we:

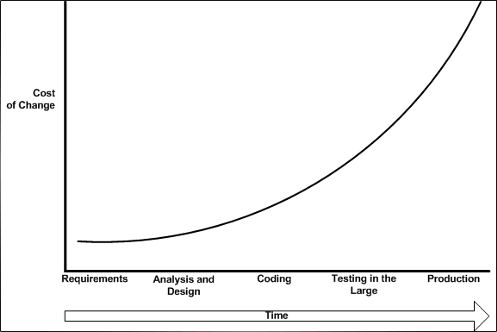
* Make assumptions about what the customer needs
* Rely on a single or indirect source of information about customer needs.

One must never make assumptions about the functionality of the project. All assumptions must be clarified with stakeholders since they are the ultimate source of information about the project.

To minimize the size and scope of the assumptions one can:

* Build quick prototypes to test out an idea,
* Research online to find solutions by others who’ve had the same problems, or
* Talk to other team members.

The earlier a risk or possible change can be identified, the less likely it is to have a long term effect on the cost or deliverability of the project.



Planning for change can take many forms, many of which are described as part of this course.

* Creating a risk mitigation plan and re-evaluating it as the project and outside environment change.
* Continually re-evaluating the priority of requirements in order to ensure that the software the client actually needs is being produced.
* Deliver updated prototypes of the product to the client for evaluation as units of functionality are completed.
* Be open to and invite client feedback
* Use evidence of the time taken for past work to inform time estimates for future work.
* Build flexibility into the design of the project software. ( use design patterns, use software design best-practices)

Constant feedback is the key component of change management – in every case where risk can affect a project, the impact of the risk can be minimized by responding to the changes introduced by the risk more quickly. Use: retrospectives, internal/external feedback tracking tools, and/or maintain regular communication – helps reduce the impact of risks.

# Readings:

A code review checklist from Fog-Creek software, which was one of the founders behind StackOverflow.com.

<http://blog.fogcreek.com/increase-defect-detection-with-our-code-review-checklist-example/>

A video presentation on how code reviews can help with quality assurance and decrease the cost of change.

<http://www.infoq.com/presentations/agile-code-reviews>

Code reviews at Atlassian, a company which creates tools to implement agile processes

<http://blogs.atlassian.com/2009/11/code_review_in_agile_teams_part_i/>

Head First Software Development, Page 51-52 "Put assumptions on trial for their lives".

<http://agilemodeling.com/essays/agileRequirements.htm> <http://agilemodeling.com/essays/activeStakeholderParticipation.htm>

<http://agilemodeling.com/essays/agileRequirementsExample.htm>

<http://agilemodeling.com/essays/prioritizedRequirements.htm>

<http://agilemodeling.com/essays/agileRequirementsBestPractices.htm#PrioritizedStack>

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

<http://agilemodeling.com/artifacts/constraint.htm>

<http://blogs.atlassian.com/2013/07/agile-requirements-documentation/>

<http://www.batimes.com/robert-galen/10-indicators-that-you-dont-understand-agile-requirements.html>