Requirement Analysis Techniques

Requirement Analysis, also known as Requirement Engineering, is the process of defining user expectations for a new software being built or modified. In software engineering, it is sometimes referred to loosely by names such as requirements gathering or requirements capturing. Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements. Here are the objectives for performing requirement analysis in the early stage of a software project:

- **From What to How**: Software engineering task bridging the gap between system requirements engineering and software design.
- **3 Orthogonal Views**: Provides software designer with a model of:
    - system information (static view)
    - function (functional view)
    - behavior (dynamic view)
- **Software Architecture**: Model can be translated to data, architectural, and component-level designs.
- **Iterative and Incremental Process**: Expect to do a little bit of design during analysis and a little bit of analysis during design.

What is Requirement?

A software requirement is a capability needed by the user to solve a problem or to achieve an objective. In other words, requirement is a software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation. Ultimately, what we want to achieve is to develop quality software that meets customers' real needs on time and within budget.

Perhaps the greatest challenge being faced by software developers is to share the vision of the final product with the customer. All stakeholders in a project - developers, end users, software managers, customer managers - must achieve a common understanding of what the product will be and do, or someone will be surprised when it is delivered. Surprises in software are almost never good news.

Therefore, we need ways to accurately capture, interpret, and represent the voice of customers when specifying the requirements for a software product.

Activities for Requirement Analysis

Requirements analysis is critical to the success or failure of a systems or software project. The requirements should be documented, actionable, measurable, testable, traceable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. Conceptually, requirements analysis includes four types of activity:

1. **Eliciting requirements**: the task of communicating with customers and users to determine what their requirements are. This is sometimes also called requirements gathering.
2. **Analyzing requirements**: determining whether the stated requirements are unclear, incomplete, ambiguous, or contradictory, and then resolving these issues.
3. **Requirements modeling**: Requirements might be documented in various forms, such as natural-language documents, use cases, user stories, or process specifications.
4. **Review and retrospective**: Team members reflect on what happened in the iteration and identifies actions for improvement going forward.

Requirements analysis is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people. Here are the main activities involve in requirement analysis:

- Identify customer's needs.
- Evaluate system for feasibility.
- Perform economic and technical analysis.
- Allocate functions to system elements.
- Establish schedule and constraints.
- Create system definitions.

Requirement Analysis Techniques

Requirement analysis helps organizations to determine the actual needs of stakeholders. At the same time, it enables the development team to communicate with stakeholders in a language they understand (like charts, models, flow-charts,) instead of pages of text.

Once the requirements are gathered, we document the requirements in a Software Requirements Specification (SRS) document, use cases or as User Stories, which are shared with the stakeholders for approval. This document is **easy to understand** for both **normal users** and **developers**. Any changes in the requirements are also documented and go through a change control procedure and finalized on approval.

Business Requirement vs Software Requirements

A business plan or project requires a variety of requirements to help define goals and establish a scope for the work that will be undertaken. Requirements also provide context and objective ways to measure progress and success. Once business requirements are established, software requirements are defined and developed in order to move a project forward.

Business Requirements

Business requirements relate to a business' objectives, vision and goals. They also provide the scope of a business need or problem that needs to be addressed through a specific activity or project. For example, if a trade association has an objective to promote the services offered by its members, the business requirements for a project might include creating a member directory that increases awareness of members. Good business requirements must be:

- Clear and are typically defined at a very high level.

- Provide enough information and guidance to help ensure that the project fulfils the identified need.
    - Understanding an organization's mandate, objectives or goals, a specific business need or problem that is being tackled
    - Should be clearly defined and understood before developing business requirements.
- The need or problem can relate to the organization or business in general or focus on a stakeholder group, such as customers, clients, suppliers, employees or another group.

Software Requirements

Software requirements break-down the steps needed to meet the business requirement or requirements. Whereas a business requirement states the 'why' for a project, a software requirements outline the 'what'.

For example, if the business requirement is to create a member directory for a trade association, the software requirements will outline who has access to the directory, how member register with the directory, who will have ownership of the data, what vehicle or vehicle will be used such as a website or paper-based directory, and so on.

Techniques for Discovering Business Needs

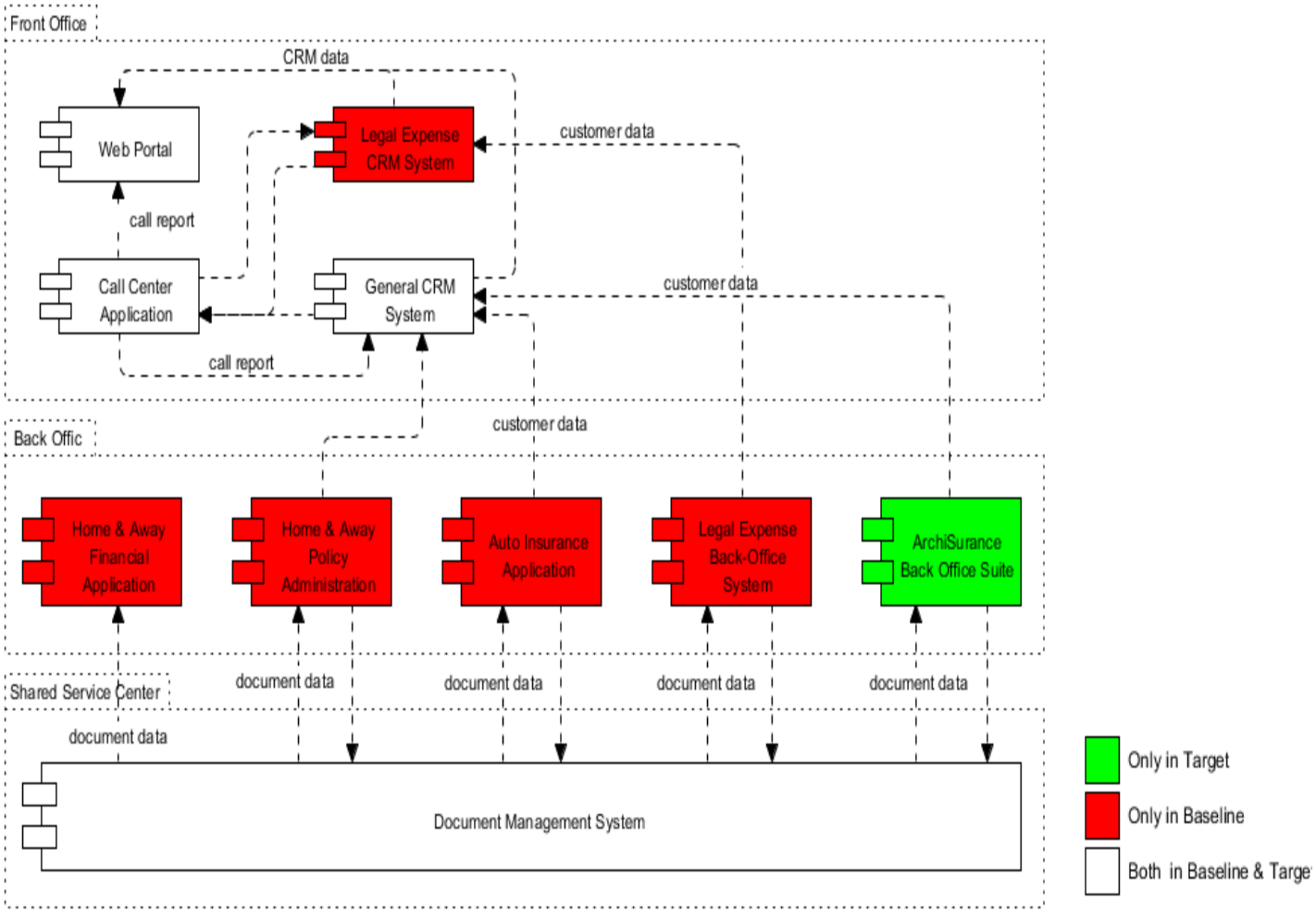
There are various requirement analyzing techniques that can be used as per the business improvement and software development process. Listed below are some of these techniques.

Gap Analysis Using BPMN or ArchiMate

A gap is often said to be "the space between where you are and where you want to be". Gap analysis is a comparison process between baseline and target business scenario. In other words, gap analysis is the study of what a business is doing currently and where it wants to go in the future, and is undertaken as a means of bridging the space between them. The goal of the gap analysis is to identify gaps in optimizing performance. This provides a business with insight into potential improvement. It answers questions like what is the current state of the project? Where do we want to be? etc.

ArchiMate - Gap Analysis

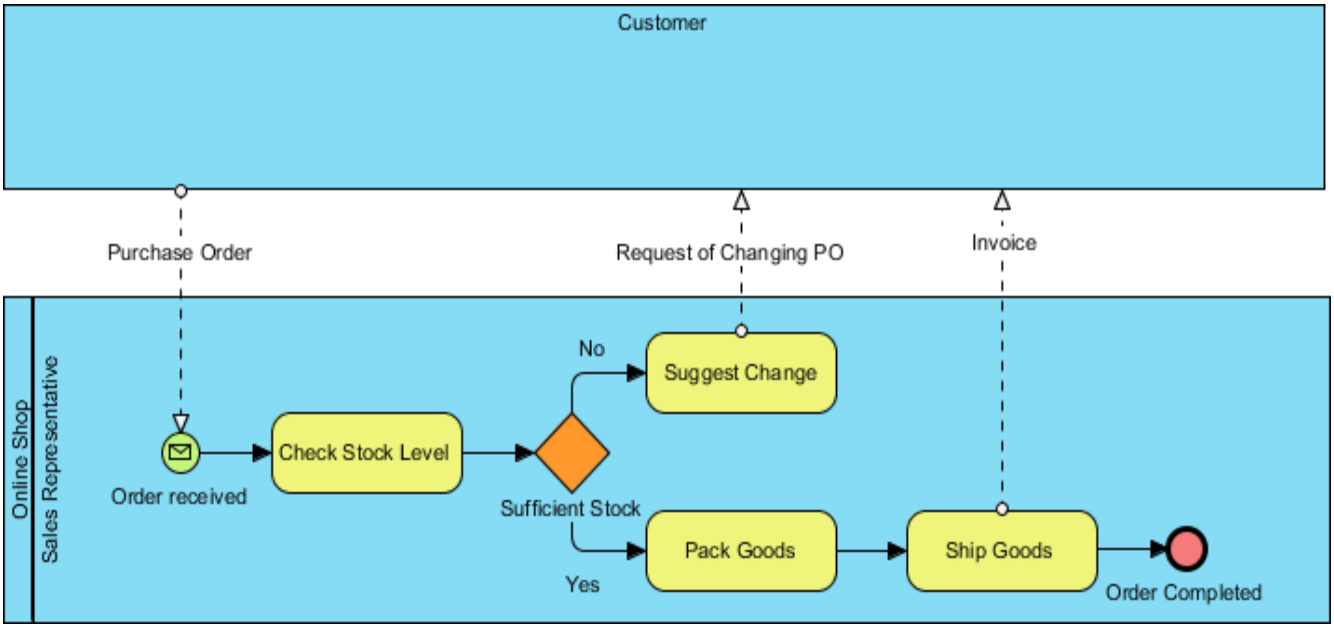
ArchiMate is an open and independent enterprise architecture modeling language to support the description, analysis and visualization of architecture within and across business domains in an unambiguous way. It's a perfect modeling tool and with the required notation to perform gap analysis.



BPMN - As-is and To-be Analysis

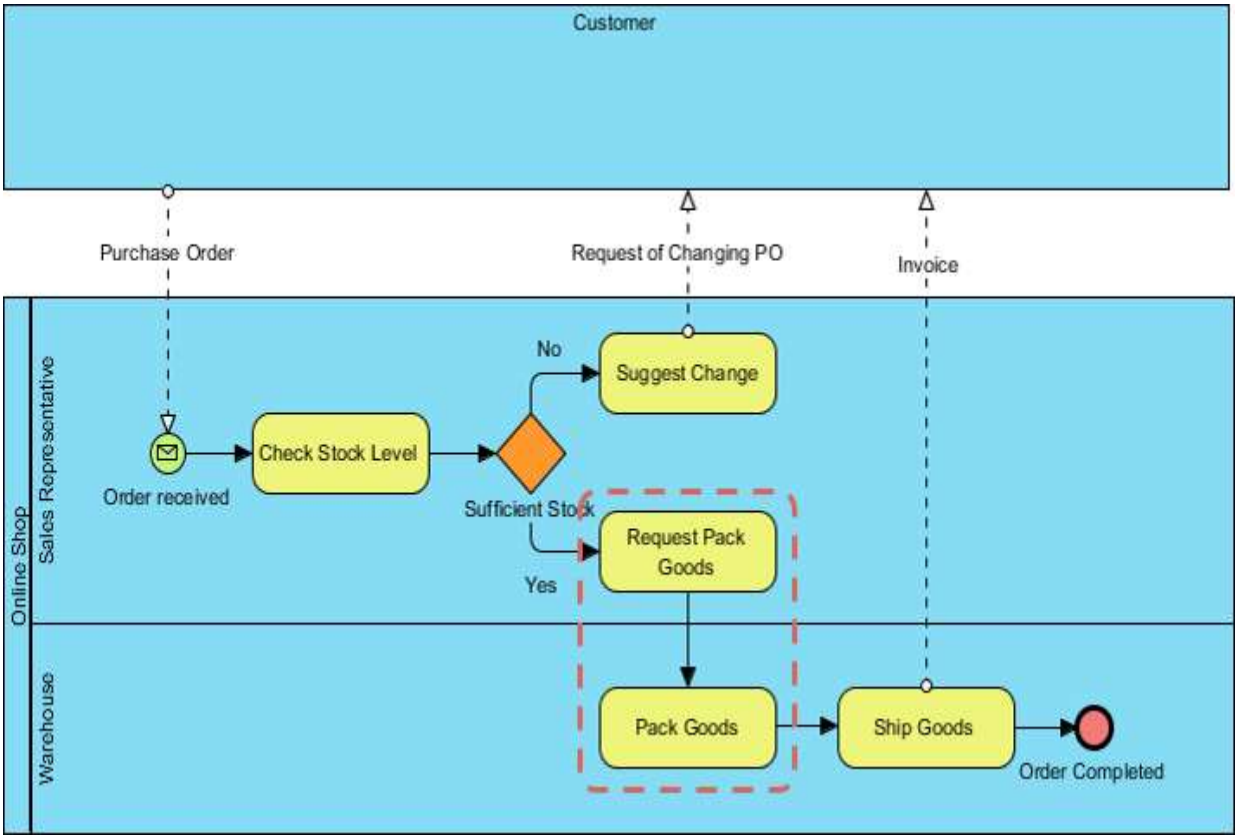
As-is Process

The example that we are going to demonstrate is about **current situation (as-is process)** of an online shop that sells goods. The process begins with the sales representative receives a purchase order from a customer and proceeds to check the stock level. If there is enough stock on hand to meet with the order, the sales representative will pack them. The process ends with shipping them along with an invoice. In case of insufficient stock, the sales representative will suggest the customer to amend the purchase order.



To-be Process

Let's just say that our business has grown so much that we now have a warehouse to keep our stocks. So we are looking for ways to improve our current (as-is) process to better allocate the new resources. Furthermore, we will show an example of modeling the enhancement below in a to-be process diagram.

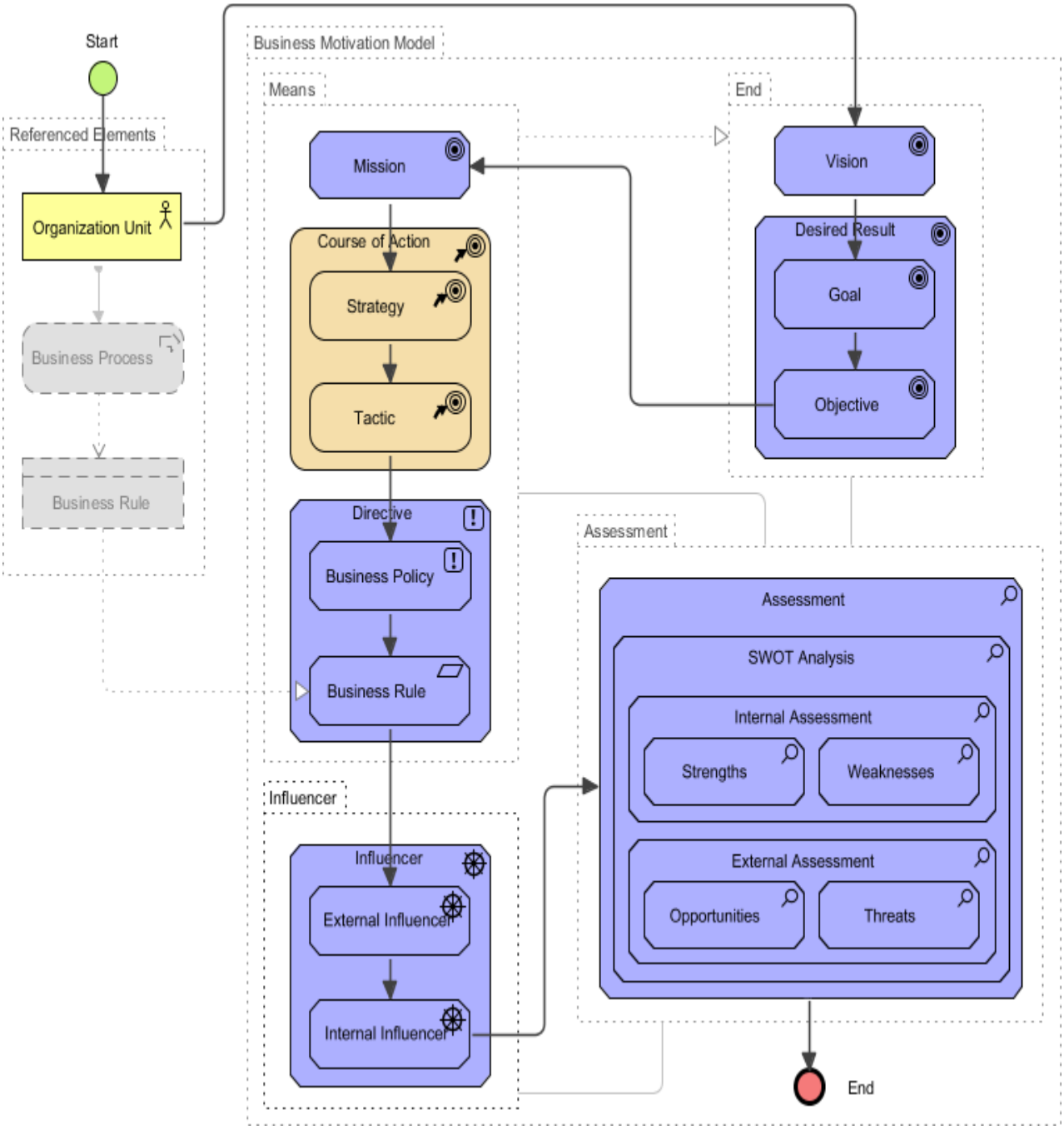


Business Motivation Model

If an enterprise prescribes a certain approach for its business activity, it ought to be able to say why and what result(s) is the approach meant to achieve. The Business Motivation Model (BMM) is an OMG modeling notation for support of business decisions about how to react to a changing world. An enterprise would use it by acquiring a BMM modeling tool

and then creating its own BMM - populating the model with business information specific to the enterprise. There are two broad purposes:

- To capture decisions about reaction to change and the rationale for making them, with the intent of making them shareable, increasing clarity and improving decision-making by learning from experience.
- To reference the outcomes of the decisions to their effect on the operational business (e.g. changes made to business processes and organization responsibilities), providing traceability from influencer to operational change.



- **End** - define what an enterprise wants to be - the states it desires to be in.
- **Means** - define what an enterprise has decided it needs to do to achieve its ends.
- **Influencer** - is something that an enterprise decides might affect it.

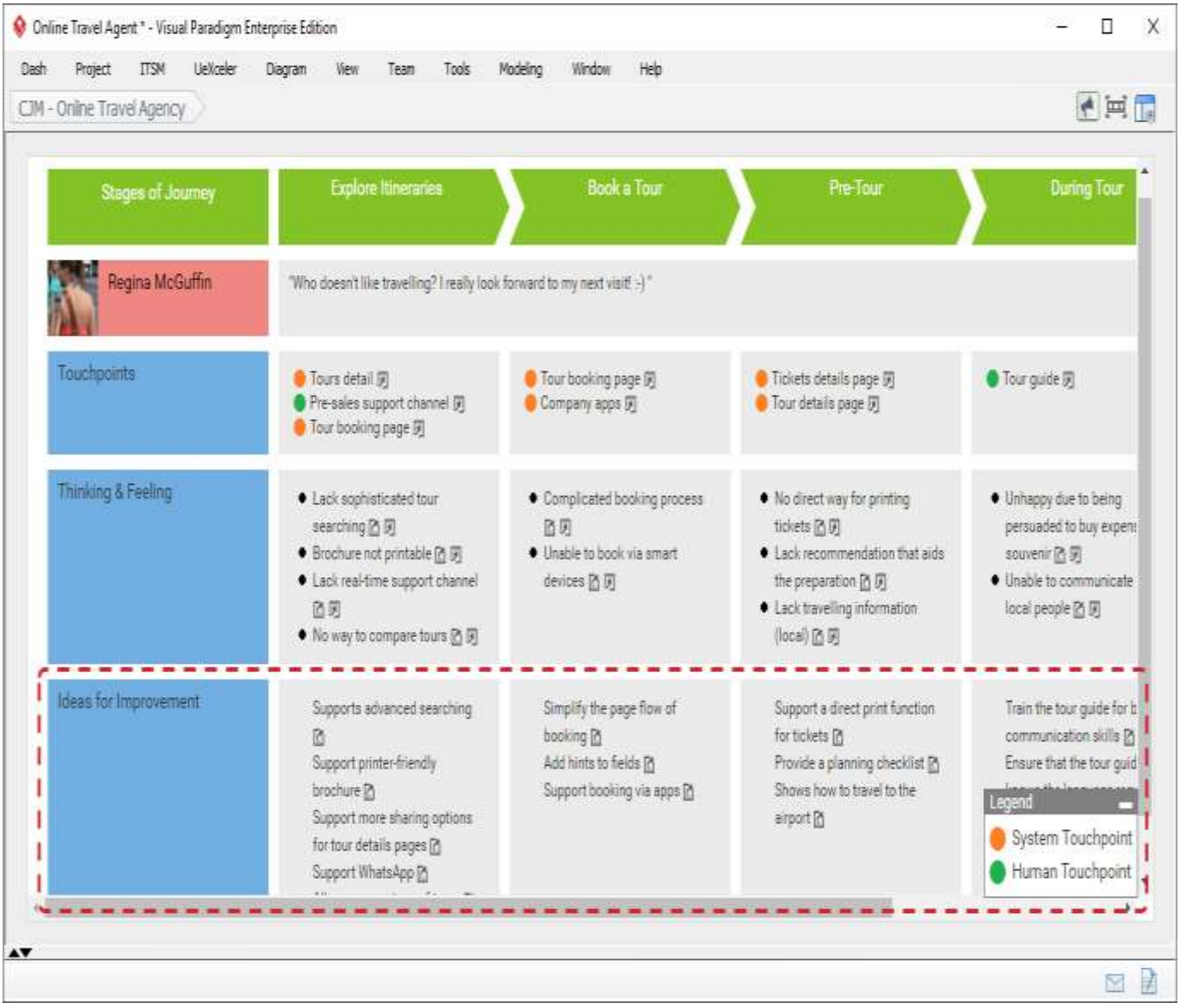
- **Assessment** - When an influencer causes a significant change, the enterprise makes an assessment of its impact, identifying risks and potential rewards. There may be multiple assessments, perhaps from different stakeholders.

Customer Journey Mapping

Customer Journey Map is a powerful technique for understanding what motivates your customers - what their needs are, their hesitations, and concerns. Although most organizations are reasonably good at gathering data about their customers, data alone fails to communicate the frustrations and experiences the customer experienced. A story can do that, and one of the best storytelling tools in business is the customer journey map.

Customer journey map uses storytelling and visuals to illustrate the relationship a customer has with a business over a period of time. The story is being told from the perspective of customer, which provides insight into the total experience of the customer. It helps your team better understand and **address customer needs** and **pain points as they experience your product or service**. In other words, mapping out the customer journey offers your business the chance to see how your brand first engages a potential customer, and then moves through the touchpoints of the entire sales process.

Finally, the team can propose the improvement or actions to be taken against each of the touchpoints. These proposed actions can be potential source of software requirements.

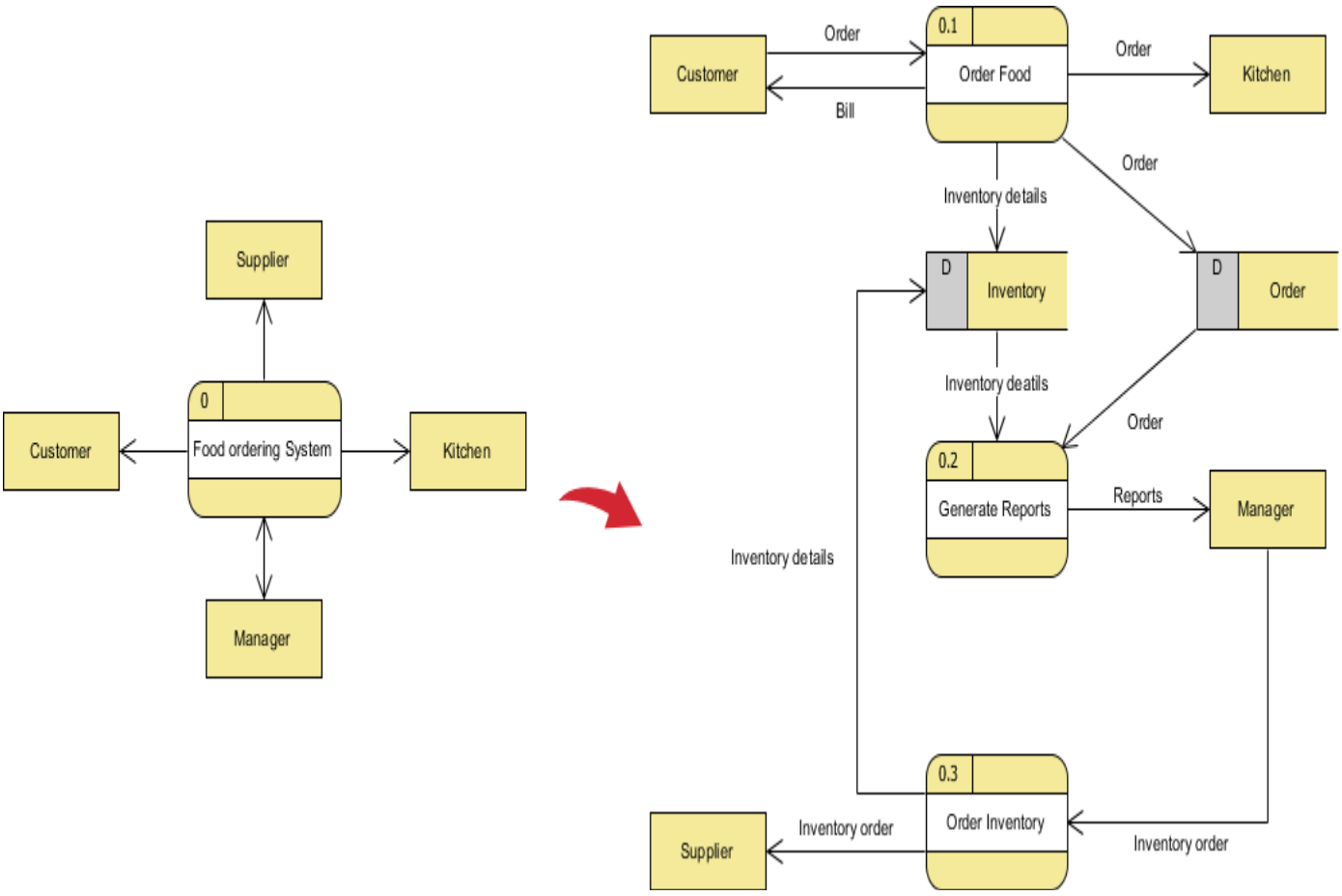


Techniques for Identifying Software Requirements from Business Needs

Data Flow Diagram

A Data Flow Diagram (DFD) can be designed early in the requirement elicitation process of the analysis phase within the SDLC (System Development Life Cycle) to define the project scope. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated.

For instance, if there is a need to show more detail within a particular process, the process is decomposed into a number of smaller processes in a lower level DFD. In this way, the Content Diagram or Context-Level DFD is labeled a "Level-0 DFD" while the next level of decomposition is labeled a "Level-1 DFD", the next is labeled a "Level-2 DFD", and so on.

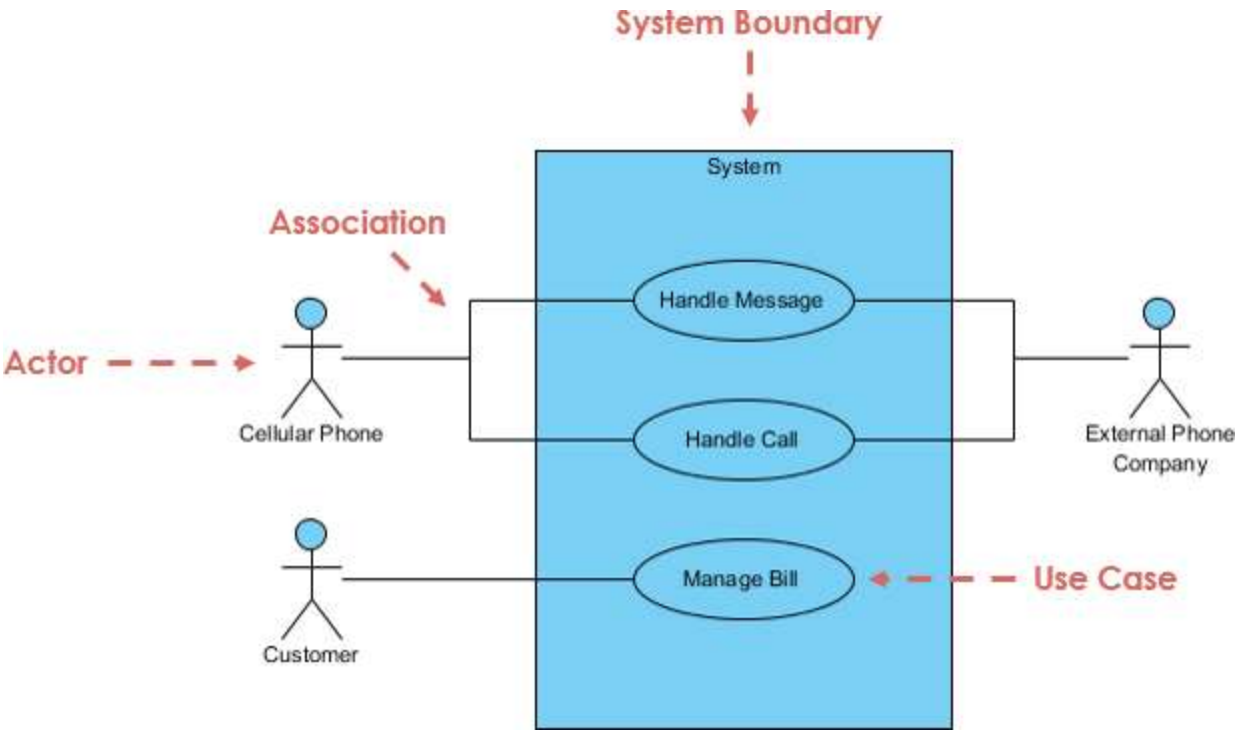


Use Cases

A UML use case diagram is the primary form of system/software requirements for a new software program under developed. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (such as UML). A key concept of use case modeling is that it helps us design a system from end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

- A use case diagram is usually simple. It does not show the detail of the use cases.
- It only summarizes some of the relationships between use cases, actors, and systems.
- It does not show the order in which steps are performed to achieve the goals of each use case.

A standard form of use case diagram is defined in the Unified Modeling Language as shown in the Use Case Diagram example below:



User Stories

A user story is a note that captures what a **user** does or needs to do as part of his/her work. Each user story consists of a short description written from user's point of view, with natural language. Unlike the traditional requirement capturing, user story focuses on what the user need instead of what the system should deliver. This leaves room for further discussion of solutions and the result of a system that can really fit into the customers' business workflow, solving their operational problems and most importantly adding value to the organization. User stories are well compatible with the other agile software development techniques and methods, such as scrum and extreme programming.

User Stories vs Use Cases - The Similarity

If we consider the key component in both approaches:

- User Stories contain, with user role, goal and acceptance criteria.
- Use Cases contain equivalent elements: an actor, flow of events and post conditions respectively (a detailed Use Case template may contain many more other elements).

User Stories vs Use Cases - The Difference

- The details of a User Story may not be documented to the same extreme as a Use Case.
- User Stories deliberately leave out a lot of important details. User Stories are meant to elicit conversations by asking questions during scrum meetings.
- Small increments for getting feedback more frequently, rather than having more detailed up-front requirement specification as in Use Cases.