Appendix A - Diagram Notation

The notation used in the diagrams and guidelines for construction of each diagram type, is detailed in my book Analysis Through Pictures (which can be downloaded free from my website @ http://www.lmunday.com/pages/publications.html).

Diagrams are used to model a structure or function according to 1 of 4 predefined views. These 4 views are:

Business View

This view captures the business structure and processes without any assumptions about technology or automation that might be used to fulfill the business objectives. Everything in the business view is assumed to be performed manually.

- **In the business view, I think of communication happening between people and that processes are performed using traditional office supplies such as paper and pen.**

System Functional View

This view captures the functions performed by a system. Each system has its own functional view. The functional view includes system use cases and system activity diagrams.

System Logical View

This view captures the data structure of a system. Each system has its own logical view. The logical view includes classes, data and relationships between them.

Deployment View

This view captures the system architecture and interfaces between internal and external systems. It describes the relationships between architecture components, the technology used by components and their interfaces and may include system configuration.

In addition to these views, 3 diagrams are used in the process that shows the mapping between views:

- A business to system mapping diagram maps business components to system functional components.
- An activity to data mapping diagram maps system functional components to logical view components.
- An activity to user interface mapping diagram maps system functional components to user interface components.

♦ **The user interface view is out of scope for this book.**

Diagram components are coloured to indicate that work is performed manually or automated by a system:

- White background - Indicates manual work.
- Blue (dark background) - Indicates work that is automated by a system.
- Orange (light background) - Indicates work that may be both manual and automated.

Artifacts colours are used to distinguish them from processes, and have no other meaning.

Process View

The process view captures the Quality through Pictures activities, artifacts and roles. This view contains 2 additional diagram types that were created for this book:

- The process Overview diagram.
- The process activity flow diagram.

The notation is briefly described for each of the following diagram types.

Process Overview Diagram

The process overview diagram provides a high-level overview of a system development process.

It is similar to a traditional data flow diagram. The process overview diagram shows the activities of the process, their input and output data flows and connections to data stores, external actors and other activities. An activity is represented by a use case. Data flows show information that is input to and output from the activities. Data stores are represented by objects.

Figure 75 shows a simple process containing 1 activity, an external actor and a data store.

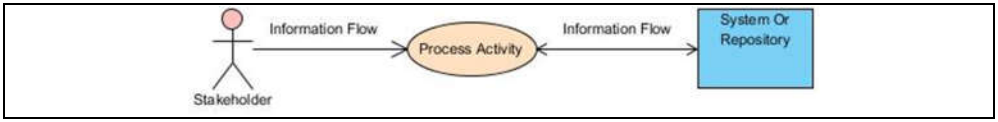

Figure 75 - Example Process Flow Diagram

An external actor supplies information to a process activity. The process activity writes to and retrieves information from a system repository.

Activity Flow Diagram

The activity flow diagram is a combination use case and data flow diagram. It shows use cases and data flowing into and out of that use case.

Each activity from the process overview diagram is captured as a use case on an activity flow diagram. The use case notation has been extended to include data flows between activities and system artifacts. Its purpose is to capture an overview of a single activity in the process. (For this reason the diagram does not include any decomposition or use case relationships, such as ‘includes’, ‘extends’ or ‘uses’.)

The components of the diagram are:

- Use case – Represents the activity performed by actors in order to achieve the described outputs.
- Primary actor – Is the person who is responsible for the success of the activity.
- Secondary actor – Are the people who support the success of the activity. Secondary actors may perform work in the activity or be recipients of work performed by the activity.
- Relationship – Connects an actor to the use case to indicate a primary or secondary actor of the use case. The primary actor is connected to the activity with a relationship containing an arrowhead. Relationships to secondary actors have no arrowhead. Relationships to actors are unlabeled, since they do not indicate information being passed.
- Artifact – Is a source for input data to the activity or represents a repository for data that is output from the use case.
- Data Flow – Connects the activity to an artifact or another activity. It indicates information input to, or output from the activity. Arrowheads on the data flow indicate the direction of the information. This may be an input flow an output flow or both.

Figure 76 shows a simple activity flow diagram containing a primary actor, a secondary actor and a repository.

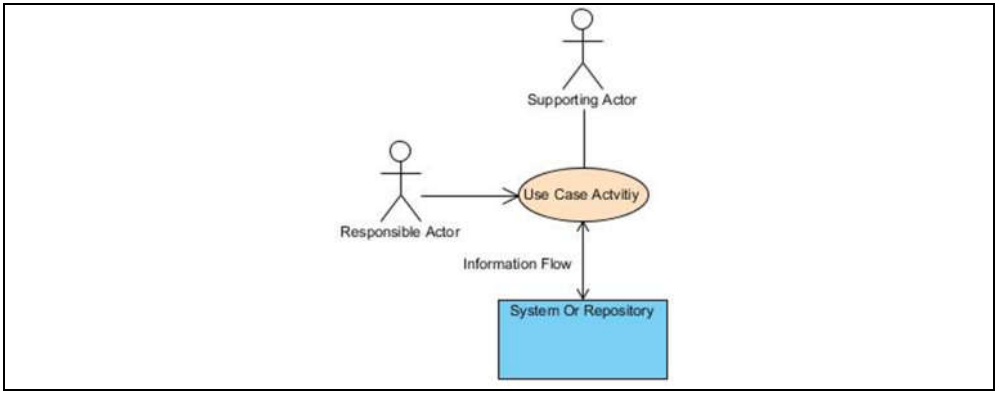

Figure 76 - Example Activity Flow Diagram

Every process flow diagram contains exactly 1 activity and 1 primary actor. There may be any number of secondary actors and input flows. There should be at least 1 output flow.

- **Although it may be possible to conceive a situation where an activity gets its input information from its actors, if nothing is produced by the activity one should question its purpose.**

Use Case Diagram

There are 2 types of use case diagram that are described in the process. The business use case and the system use case. Both use similar notations. Both diagram types include use cases connected to actors by relationships.

- **When drawing use case diagrams I prefer to separate use cases as much as possible, since this helps with maintenance of the diagrams and gives the reader the impressions that each use case represents a distinct piece of functionality. This is why actors are repeated throughout the diagram.**

Business Use Case Diagram

A business use case diagram captures business actors and what the business does, without detailing how they do it. The primary actors on the diagram represent roles of the business that are responsible for a specific objective. Secondary actors on a business use case diagram represent people in an eternal organization, who provide input to the process.

The business use case diagram notation is the same as that described by the Rational Unified Process. The components of a business use case are:

- Use case – Represents the work performed by people who are business workers. All work in a business use case is performed manually. (A business use case is system independent.)

- Primary actor – Is the role played by the person who is responsible for the success of the use case. (This is not necessarily a role that performs work in the use case.)
- Secondary actor – Someone who contributes to the success of the business use case, without necessarily performing any work described the use case. (A secondary actor may be someone in a different department who provides information to the workers of the use case.)
- Relationship - Connects actors to the use case. The connector to the primary includes an arrowhead at the use case end of the connector. Connectors to secondary actors do not contain arrowheads.

♦ **Note that connectors indicate relationships actors have with a business use case and do not contain any information.**

Figure 77 show an example business use case diagram with 1 primary andd 1 secondary actor.

♦ **A business use case diagram may contain several business use cases. I use the diagrams to organize business use cases by department.**

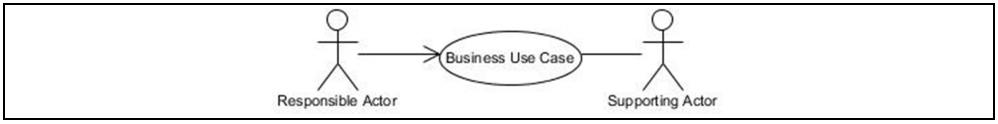

Figure 77 - Example Business Use Case Diagram

A business use case diagram contains 1 or more business use cases with exactly 1 primary actor connected to each use case. Any number of secondary use cases may be shown connected to a use case on the diagram.

♦ **Note that the business use case and its actors are shown with a white background. Since the white background indicates a manual process, it can be deduced that these are business actors and a business use case.**

System Use Case Diagram

The system use case diagram notation is similar to that used in the business use case diagram, except that a use case on the diagram represents the work performed by a system, and its interaction with its actors. As with the business use case, the system use case has a single primary actor and zero or more secondary actors. The difference is that the primary actor represents a person who initiates the system use case. Secondary actors may be people or systems that interface with system.

A system use case captures automated steps that are performed by the system. The system use case does not capture manual steps of an activity (except where a person/actor is interfacing with the user interface). If a manual (or external system) step is required to complete the system use case, then the use case

should not make any assumptions about how that step was performed. The use case waits until it can continue to the next system step. (In this manner, there is no confusion about what is in scope for the system and what is out of scope.)

- **Every step that is captured by a system use case is performed by the system that the use case is describing. If several external system or manual steps are required to move to the next step, then end the use case and continue the activity steps in a subsequent system use case. When the process steps become manual or they are performed by another system, this system has fulfilled it purpose.**

The notation used for system use cases is the same as that used in the Rational Unified Process. Its components are:

- Use case – Represents the activity performed by the system being modeled.
- Primary actor – The actor who initiates the activity in the use case (a primary actor may be a person or an external system).
- Secondary actor – is a person or system interacts with the system during the system use case activity
- Relationship – Is a connector between the use case and one of its actors. An arrowhead on the connector indicates the actor is the primary actor of the use case. A connector with no arrowheads indicates a secondary actor.

- **None of the examples in this book use the extends, uses or inheritance relationships. These relationships add complexity to the use cases, so I discourage their use, except in circumstances where significant steps in a use case are repeated. The simpler the use cases, the cheaper it is to understand and maintain them.**

Figure 78 contains an example system use case diagram.

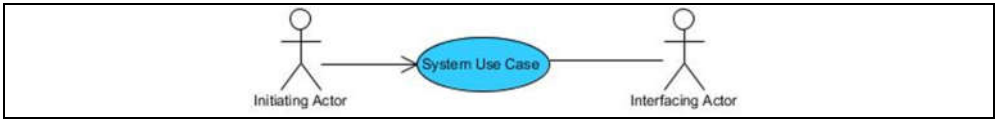

Figure 78 - Example System Use Case Diagram

- **The system use case is shaded to indicate that it describes automated functionality.**

A system use case diagram contains any number of system use cases. Each use case is connected to exactly 1 primary actor, and any number of secondary actors.

- **A system use case diagram captures use cases for a single system. In this manner, the diagram can be located in the same model package as its use cases.**

Activity Diagram

The activity diagram notation used in this book uses a simplified variation of the UML notation.

Activity diagrams are primarily used to detail the steps performed by a use case. They show the flow of the activity in the use case from a start state to a stop state. Between the start and stop states, actions capture the steps that are performed during navigation through the diagram. Actions are connected in sequence using directed transitions. Each connector represents an event that transitions the flow between components on the diagram. Forks and decisions are used to show branches and parallel flows between the steps in the activity diagram. Transitions connect all of the components in the diagram to show all possible paths through the activity. All activity diagrams begin with a single start state and terminate with one or more end states.

The components are the same in the business activity and system activity diagrams, but there differences between what these components represent.

The actions in a business activity diagram represent actions performed manually by business workers. (Even if the step involves access to a system, it should represent what the step is achieving, and not how it is being achieved.) The actions in a system activity diagram represent work performed by the system.

Transitions represent events that cause 1 step to end and the next to begin. In the business activity diagram, these events are the result of a business action. In the system activity these events are occur as a result of something recognized by the system.

Decisions in a business activity diagram are made by the actors that are shown on the business use case diagram. Decisions in a system activity diagram are made by the system described by the use case.

Figure 76 contains an example Activity Diagram.

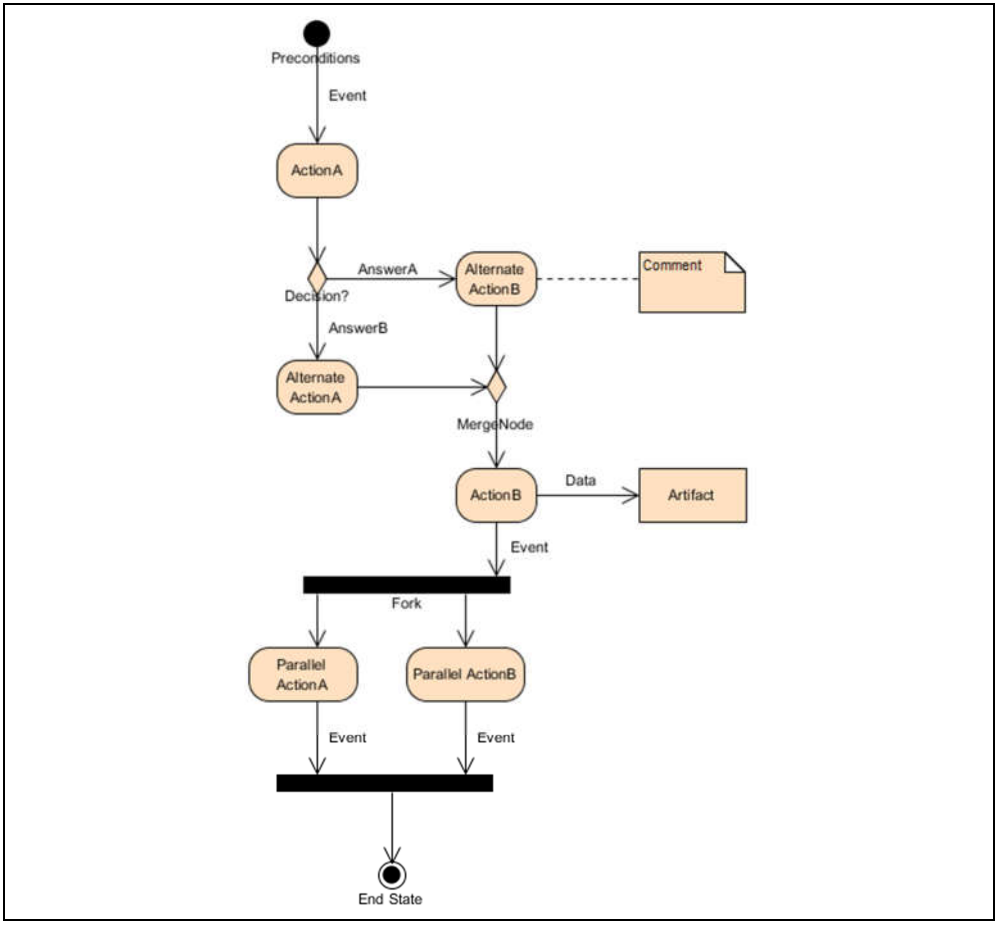

Figure 79 - Example Activity diagram

The diagram is generally read from top to bottom. The activity presented in the diagram can execute only if the start state's preconditions are satisfied. An event causes the first action after the start state to execute. This event is represented by the transition executing the start state. As further events are recognized actions are executed in sequence, until a stop state is reached.

One of the alternate actions *Alternate ActionA* or *Alternate ActionB* is executed during a pass through the diagram. (The decision symbol named *Decision?* ensures that exactly one of these actions is executed.)

Both parallel actions *Parallel ActionA* and *Parallel ActionB* are executed during every pass through the diagram. (The fork symbol labeled *Fork* ensures that both actions are executed.)

Comments may be added to the diagram to add clarification.

Artifacts may be added the diagram represent data that is used by or output from actions.

- **The diagram is shaded to represent that actions may be performed manually or automatically.**

Business Activity Diagram

The business activity diagram captures a process that satisfies a business need. Each action on the diagram represents a step that a business worker performs in achieving that business need.

- **Actions on the business activity diagram are coloured white to indicate that they are performed manually.**

System Activity Diagram

The system activity diagram captures the steps performed by the system when interacting with external actors, in order to satisfy a functional requirement. Actions represent functionality performed by the system. Events represent interactions between the system and its actors. Actors are users of the system, or interfaces to other systems. This are shown connected to the parent system use case on a system use case diagram.

- **Actions on the system activity diagram are shaded blue to indicate that they are performed by a system.**

Textual Representation Of The Activity Diagram

Activity diagrams may be written in a textual form as steps of a use case. The Use case steps take the form of:

- Precondition – This represents the start state of the activity diagram.
- Steps – Steps represent the actions and the transitions between those steps. Each step is given a unique number within the use case. This allows the step to be referenced by any other step in the same use case.
- Postcondition – This represents a stop state in the activity diagram.
- Main Path – This is the expected sequence of steps through the use case. It is preceded by the precondition and terminates with the expected postcondition.
- Alternate Path – This is a sequences of steps in the use case that may or may not lead to the expected postcondition. An alternate path is initiated from a use case step in a different path. It terminates by returning to a step in a different path, or by reaching an unexpected (or error) postcondition.

An example textual use case is shown in Figure 80.

Precondition	
<start state description>	
Main Path	
1. <transition description>	
2 <action description>	
The use case ends	
Postcondition	
<stop state description>	
Alternate Path Name	
At step <initiating step>	
10 <transition description>	
11 <action description>	
The use case returns to <returning step>	<alternate postcondition>

Figure 80 – Example Textual Use Case

The step numbers run in sequence from top to bottom of the use case.

There may be several alternate paths in the use case. Each alternate path ends with a postcondition, or it returns to another path in the use case.

- **The use case examples in this book do not use a table.**

Business To System Mapping Diagram

This diagram is the connection between business and system use cases. It maps work that is the responsibility of people in the business, to system activity. It shows which business actions are automated and which are performed manually.

The components of the diagram are:

Business Action – is an action taken from the business activity diagram

System Use Case – is taken from the system use case diagram

Trace Relationship – identifies which system use case satisfies a business action

Figure 81 shows an example of the components on a business to system mapping diagram.

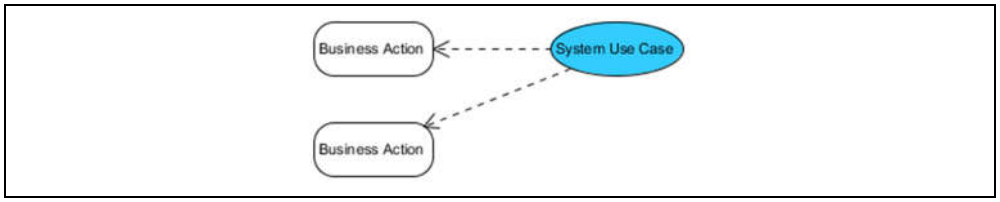

Figure 81 - Example Business To System Mapping Diagram

A business action is satisfied by exactly 1 system use case. A system use case may trace from several business actions. The diagram shows that both business steps are satisfied by the same use case.

- **If a business step is satisfied by more than 1 system use case, I recommend splitting that business step into several actions, so that there is no more than 1 system use case tracing from a business action.**

Class Diagram

The class diagram shows the logical components of a software application. These are depicted by classes that are connected by relationships. There are several types of relationship; each provides a specific meaning when connecting classes.

The class diagram notation is standard UML. This is book is not going detail class diagram notation.

- **The UML class diagram notation is freely available online and it is detailed in my book, Analysis Through Pictures.**

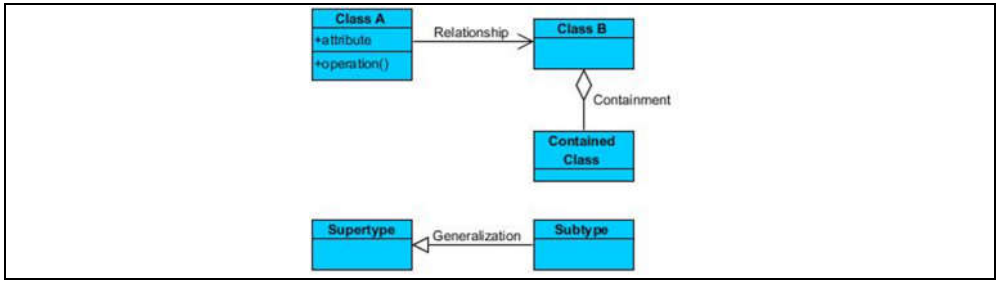

Figure 82 - Example Class Diagram

The example class diagram shows Class A connected to Class B with an association type relationship. The Contained Class is shown as part of Class B using an aggregation relationship. Supertype and Subtype classes show an example of the generalization relationship.

Deployment Diagram

The deployment diagram captures the structure of physical hardware nodes, the software components that execute on each node, and how the different components are connected.

The objects on a deployment diagram are nodes and components. Nodes are physically connected. Components are connected by software interfaces.

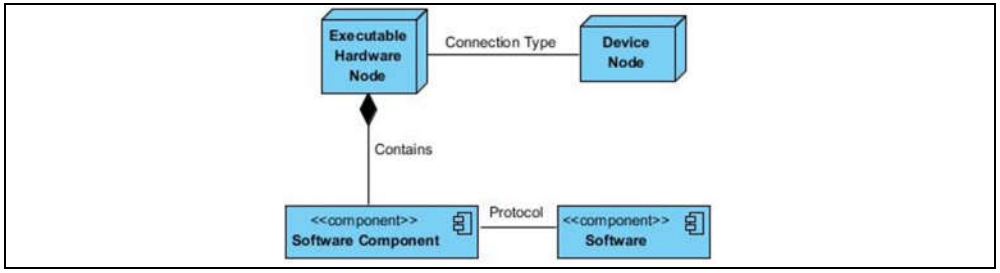

Figure 83 - Example Deployment Diagram

The example diagram shows a device node connected to a hardware execution node. The connection represents the physical interface between the 2 nodes. The execution node hosts a software component. This interfaces with another software component. The interface represents the protocol used by the software components to communicate with each other.

Sequence Diagram

The sequence diagram shows interactions between actors and system components over time. In this book, sequence diagrams are initiated by an external actor and all interactions are externally visible.

The sequence diagram is read from top to bottom. Each actor or system component has an associated lifeline running from top to bottom of the diagram. Lines connecting these lifelines represent events that are communicated between these actors and system components.

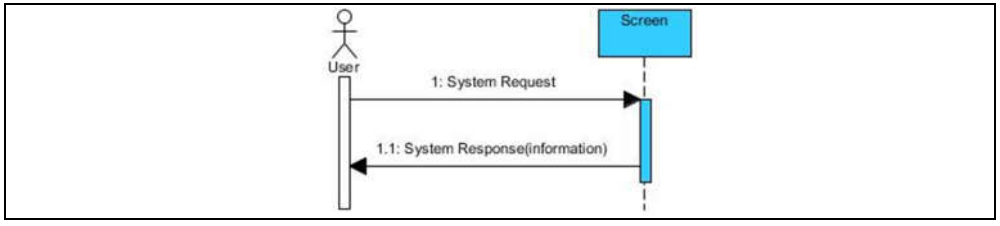

Figure 84 - Example Sequence Diagram

The example sequence diagram shows an actor communicating with a system via its user interface. The user makes a system request from a user interface screen. The system responds to the user with a response on the same screen. Lifelines under the User and the Screen components show a related sequence of events.

- **Sequence diagrams are equally useful for describing internal system design details.**