

REQUITOPIA

A NEW PARADIGM IN REQUIREMENTS MODELING

User Guide

Version 0.8A

July 18, 2007

Copyright 2007, Isotope 28

www.isotope28.com



Requitopia - Quick Reference Guide

Requitopia - User Guide

Thomas A. Bullinger

Copyright (c) 2007 by Isotope28

All rights reserved. Printed in the USA.

Available online at: <http://www.isotope28.com/Requitopia>

While every precaution has been taken in the preparation of this material to make the information as complete and accurate as possible, the author assumes no responsibility for errors or omissions, or for damages resulting from the use of this information contained herein, which is provided on an "as-is" basis.

Contact:

Thomas A. Bullinger

Isotope28

2500 Turk Hill Road

Victor, New York 14564

<http://www.isotope28.com>

mailto: tom@isotope28.com

Table of Contents

User Experience	5
General Workflow	6
Creating an Application	7
Create a Problem Statement	10
Create the Application Context	11
Create Use Case Diagrams and Use Cases	14
Create Use Case Narratives	16
Refactoring Use Cases	22
Priority and Status of Use Cases	23
Create Participant RC Cards	24
Create Relationship Diagrams	26
Next Steps	30
Notation Reference	32
Glossary of Terms	36

Introduction

Requitopia is a requirements modeling tool designed to facilitate the gathering and capture of information to generate functional specifications for products under development. While the primary target audience is business persons in the software development business, Requitopia can be applied to any product needing requirements including: software desktop applications, business process reengineering, embedded systems, information technology infrastructure, electrical and mechanical devices.

The basis of the functional requirements are use cases as defined by Ivar Jacobson, in his book *Object-Oriented Software Engineering: A Use Case Driven Approach*. In addition to use cases, supporting information is also created to help understand the structure of the use cases and the relationships between the various participants of the use cases. The supporting information includes:

- ◆ A complete navigation model rendered in the form of a Mindmap.
- ◆ Context diagrams to identify the stakeholders.
- ◆ Stakeholder profiles to detail stakeholders benefits and contributions.
- ◆ Relationship diagrams to illustrate the relationships between the participants.
- ◆ Use case diagrams to supplement the visual structure and context of the use cases.
- ◆ Use case descriptions to capture the context, preconditions, workflow, and results of each use case.
- ◆ Extensions in the form of variants, exceptions, technology and data to use case workflows.
- ◆ Responsibility-Collaboration descriptions to capture the knowledge and behaviors of each participant in a use case workflow.

Requitopia provides for four levels of abstraction for modeling requirements. The multiple levels of abstraction provide for varying levels of detail allowing the modeler to focus on the abstractions appropriate for each level. The four levels of abstraction are those recommended by Alistair Cockburn in his book "Writing Effective Use Cases". They are:

- ◆ Business - Requirements defined solely in the business domain without regard to technology
- ◆ System - Requirements which define the roles and responsibilities of the architectural components as derived from the Business requirements

- ◆ Subsystem - Requirements defining the details required for the implementation of the behavior and data internal to each architectural components.

In addition, a navigation pane is provided for the entities defined as classes. The class navigation pane is ideal for viewing the details of a class, or providing an external link to implementation files.

All supported diagrams utilize the Unified Modeling Language (UML) version 2.0 as approved by the Object Management Group (OMG). A simplified version of UML is employed at the high levels of abstraction to facilitate communication to UML lay-persons.

Requitopia provides a complete, traceable model of requirements consistent with the Software Engineering Effectiveness Model (SEEM). As such, a complete software development cycle can be modeled with the tool.

User Experience

The Requitopia tool harkens back to an early time in our lives, and draws on that experience for the user interface. Rather than providing 50 different ways of doing things, hundreds of menu choices, and many cryptic buttons and toolbars, we have tried to simplify the interface as much as possible.

The simple premise is that of pencil and paper. Before computers made our lives more complicated, a problem statement would be written in this simple way:

- ◆ Pull a piece of paper from a stack of clean sheets and place it on the desk.
- ◆ Grab a pencil and start writing on the paper.
- ◆ When done writing, tired of looking at the paper, or if more space is needed on the desk, the paper with the problem statement is filed away. The paper can be pulled from the drawer and revised at any time.

This simple paradigm is applied to all diagrams and artifacts with few exceptions. New artifacts (blank pages) are created off the right mouse button from a context-sensitive menu. There are no invalid operations possible within the tool, and the built-in configuration management facility ensures that entered information is never lost. The tool performs many of operations normally required of the user, and greatly simplifies interaction with the tool. The results are clean, very efficient, and some might think spartan. We hope that in this case, “less is more.”

In addition to a spartan user interface, Requitopia has been created with built-in configuration management. The advantage is that information is never deleted or lost in the model. If an item is “Removed” from a diagram, only its representation on the diagram is removed, and the entity is not deleted from the model (as it may also be used on another diagram!) When an item is “Deleted” from the navigation pane, it is removed from all diagrams, and deactivated in the model so it does not appear in the navigation panes. However, the entity still exists in the database if recovery is required. Each

change to an entity also creates a new version of that entity. Previous versions are available through the catalog feature.

The penalty for the degree of automation built into Requitopia is that the tool may be difficult to use in situations that we have not foreseen. If you find this is true in your case, please contact us and we'll help you figure out how to address the issue, or update the tool to accommodate your needs.

General Workflow

The workflow for creating a requirements model for an application is consistent with the SEEM methodology. The high-level SEEM process flow is shown in figure 1.

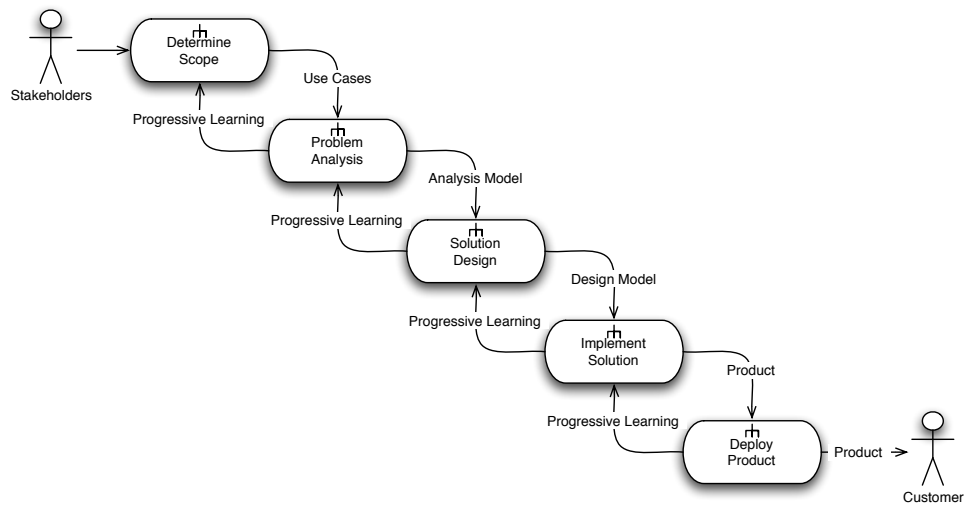


Figure 1 - SEEM Overview

The requirements gathering process begins with determining the scope of the application under consideration. Figure 2 represents the details of the first activity set in the SEEM process.

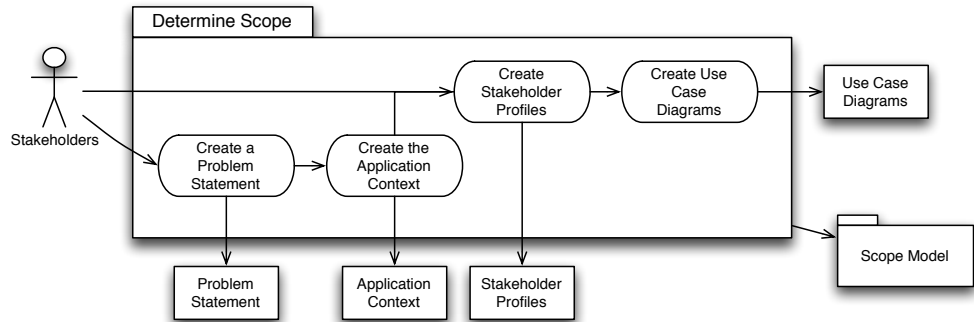


Figure 2 - Determining Scope

The Requitopia tool follows the activities as described by the SEEM process, and begins with defining an application, then follows the remainder of the specified activities. Each of these activities will be explored in turn.

Creating an Application

Requitopia is nothing more than a sophisticated user interface on top of a data model containing the requirements and associated information. The first task is to create an application within the database as a container for all our subsequent requirements model elements. When Requitopia is started, the user is presented a view into the database repository for applications. Any applications currently in the repository will be listed in the upper section of the window, and the user can select any of the existing applications to open it. For new applications, the user need only type the new name into the text field at the bottom of the window, and hit return. The new application is created and opened. Figure 3 illustrates the repository view for creating and opening applications.



Figure 3 - Application Repository View

Once the application is opened, the navigation view of the application is presented. Figure 4 illustrates the navigation pane for an example application.

Requitopia - Quick Reference Guide

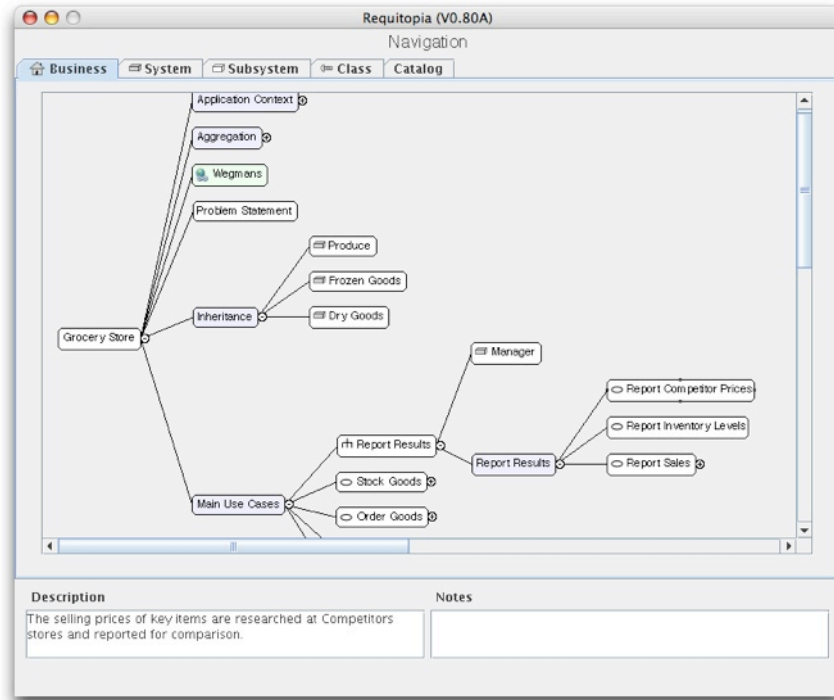


Figure 4 - Application Navigation View

The central portion of the navigation pane is a mindmap of the application and its structure. The primary node on the left side of the navigation pane represents the application under development. The components of the application are represented by the sub-nodes branching to the right. The nodes can be expanded or contracted as desired by selecting the + or - control on the right of the navigation node. If the node does not have any children in this context, the expansion control is not visible. Selecting a node with a single click of the mouse presents the description of the object and its notes in the text areas at the bottom of the navigation pane. Selecting a node with a double-click of the mouse will open the details of the node in a separate window.

Four levels of abstraction are represented in the navigation view by four tabs found at the top of the navigation pane. The first tab is for the high-level, or business requirements model. The other tabs represent increasing levels of detail for systems, then subsystems and classes. The final tab is a catalog of all the entities in the requirements model, and provides limited access to the built-in configuration management features. The other tabs will be discussed later.

Create a Problem Statement

Requitopia has been created with a unique user interface which simplifies the user interaction with the tool, and provides for the most economical interaction with the modeling elements. Most operations within the tool are accessed via a context-sensitive menu activated with the right mouse button. Only those actions appropriate for the entity under the mouse cursor will be displayed in the menu.

To add a problem statement to the application, move the mouse cursor to a location over the application node in the navigation pane, and press the right mouse button. Figure 5 demonstrates the menu selections from the application in the navigation pane.

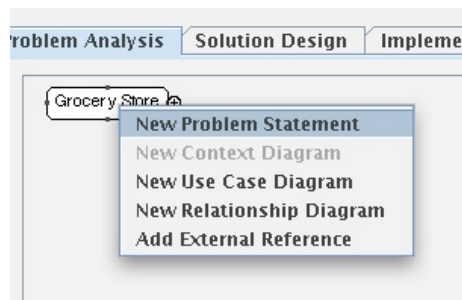


Figure 5 - Application Main Menu

To add a problem statement, select the menu option for "New Problem Statement". A new node will be added to the navigation pane to represent the problem statement, and the problem statement view will be opened in a separate window. The problem statement is free-form text, and the text can be pasted from another document, or typed in manually. The problem statement text is a short description of the problem to be solved, including any constraints or limitations to consider. The problem statement should be suitable for an "elevator pitch". Figure 6 is a representation of a problem statement.

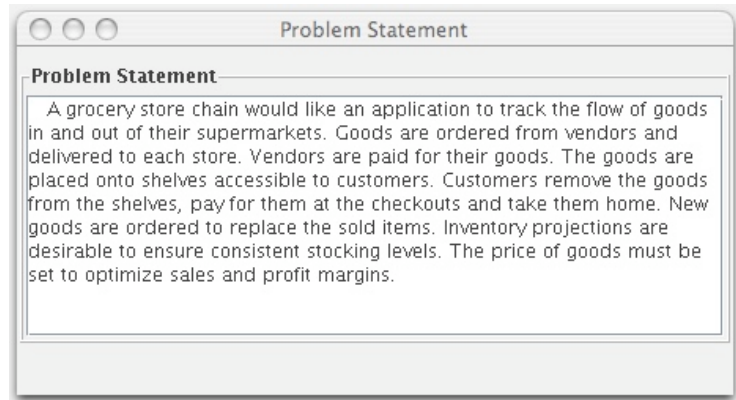


Figure 6 - A Problem Statement

Only one problem statement can be added to an application, and cannot be added to other entities.

Create the Application Context

The application context diagram provides a visual catalog of the stakeholders of the application under development. The purpose of the application context is to brainstorm and record the potential stakeholders and explore their contributions and benefits to the application.

To create an application context diagram, move the mouse cursor to the application node in the navigation pane, press the right mouse button and select the menu option for "New Context Diagram". A new context diagram will be created for the application, added to the navigation pane, and opened in a separate window. The default context diagram contains only the application represented by the box in the center of the drawing.

Stakeholders are added to the diagram by moving the mouse cursor into the drawing pane for the context diagram, placing the mouse cursor where you would like the stakeholder initially located. Press the right mouse button and select the menu option for "New Stakeholder". A new stakeholder is created with a default name, added to the diagram, and their description is opened in a separate window. Change the default name of the stakeholder to the desired name, and add a brief description, contributions to the application, and / or benefits of the application to the stakeholder. Close the diagram when the information is completed to your satisfaction, and the navigation node and stakeholder label on the diagram will be updated to match. The stakeholder detail view can be opened at any time by double-clicking on the stakeholder in the context diagram, or on its node in the navigation pane.

Existing stakeholders can be added in a similar fashion via the mouse menu.

Figure 7 is an example of an application context diagram, and figure 8 a stakeholder profile.

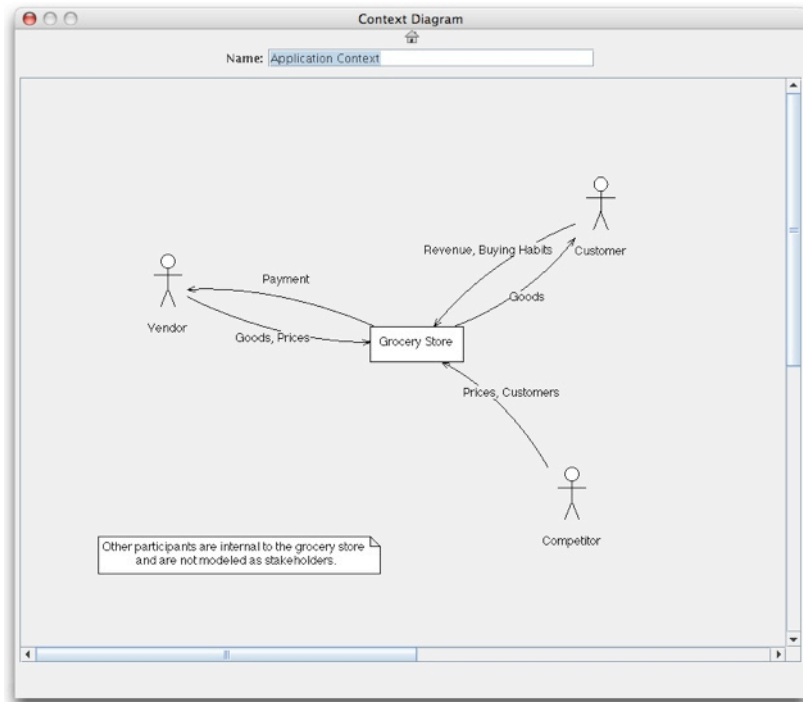


Figure 7 - The Application Context

A context diagram can only be added to an application in the problem analysis model, and only one can exist.

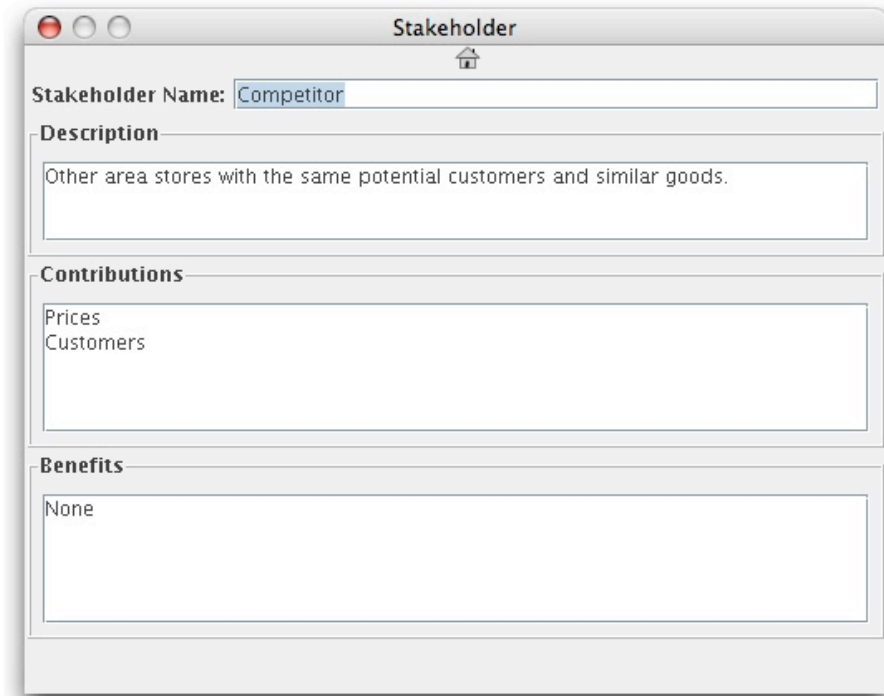


Figure 8 - A Stakeholder Profile

To add a connection between a stakeholder and the application, move the mouse cursor over the source stakeholder or the application on the context diagram, and press the right mouse button. Select the menu option for "Add Connection", then move the mouse cursor to the destination stakeholder or application, and select the destination with the left mouse button. While connecting, a line will be drawn from the source object to the mouse cursor. When the destination is selected, the connection is added to the drawing, and a dialog is opened for entering an optional label on the connection. Enter a label into the dialog box, and hit return or close the box. The label on the connection is added to the context diagram. The label is usually a brief representation of the contributions or benefits found in the stakeholder profile. The connection label can be changed by double-clicking on the connection with the left mouse button. The same connection dialog is opened and the label can be updated.

On the context diagram and all other diagrams, notes can be added to supplement the default drawing elements. Notes are added in the same manner as stakeholders via the right mouse button menu. Move the mouse cursor to the location you would like the note to appear, press the right mouse button and select the menu option for "New Note". A new note is created and placed on the diagram with default content. A dialog box is opened to update the default text. A note is free-form text, and can contain multiple lines

of information. Figure 9 is a simple note, and illustrates multiple lines of content with deliberate line breaks added for setting the aspect ratio of the note on the diagram. The note can be seen rendered on a diagram in figure 7.

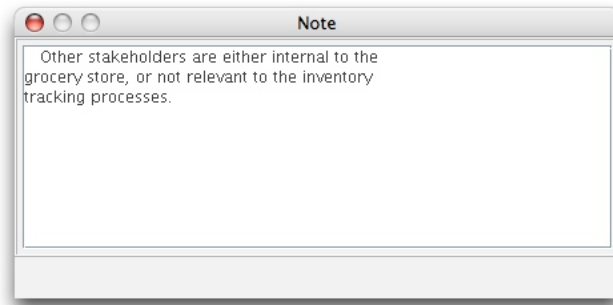


Figure 9 - A Note

Notes can also be connected to other objects in the context diagram, and are connected in the same manner as the other objects. The connection between a note and another drawing element defaults to a UML dependency relationship shown as a dashed arrow from the note to the object to which it refers. Note need not have connections.

On the context diagram, as with all other diagrams, the elements can be moved freely to other locations. Move the mouse cursor over the drawing element you would like to move, then press and hold the left mouse button. Drag the mouse while holding the mouse button down to the desired location of the drawing element and release the mouse button. The drawing element and all its connections will be relocated to the new position.

Drawing elements can also be removed from the diagram. To remove a drawing element, move the mouse cursor over the item you would like to remove, press the right mouse button and select the option to "Remove". The selected drawing element is removed along with any connections. Stakeholders that have been added to the diagram, then removed still exist in the application, and can be restored onto the diagram by adding "Existing Stakeholders".

Create Use Case Diagrams and Use Cases

The primary item of interest for requirements are use cases -- as they represent the functional specifications for the application at the varying levels of abstraction. The modeling elements discussed thus far are merely tools to assist in the identification of the use cases for the application. As use cases are the primary purpose of the requirements modeling effort, they are the centerpiece of the Requitopia tool.

A use case model is best created in the form of a hierarchy. To facilitate the hierarchy, the application can have only one use case diagram. All other use case diagrams are derived from the use cases found on the main use case diagram. To add the first use case diagram, move the mouse cursor over the application node in the navigation pane, and select the menu option for a “New Use Case Diagram”. A new diagram is created with a default name and added to the navigation tree. The new diagram is opened in a separate window.

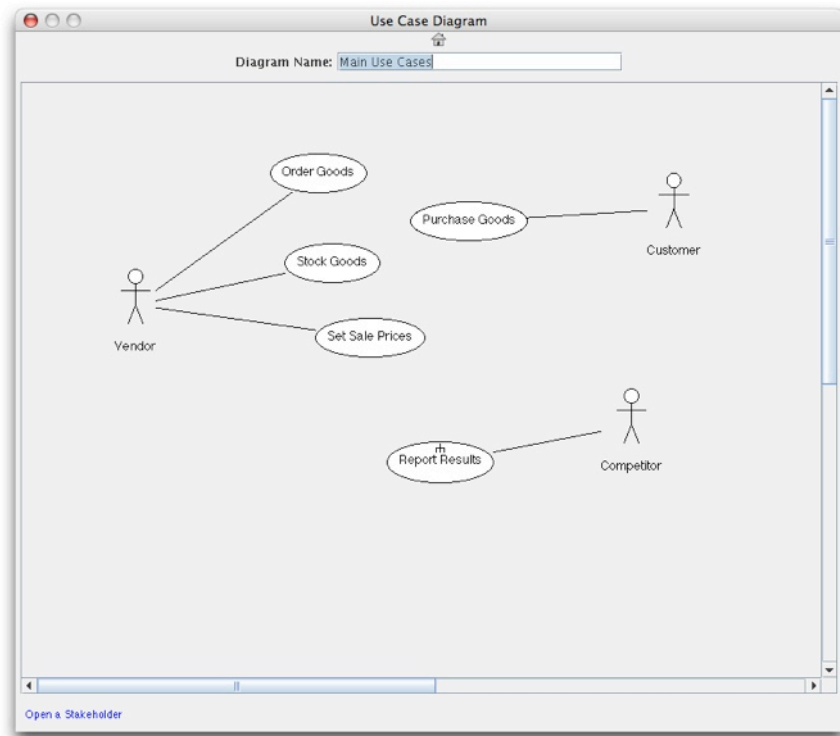


Figure 10 - A Use Case Diagram

Use cases and actors are added to the diagram in a same fashion as additions to the other diagrams. Use cases can be added as needed. Actors can be added to the diagram, but actors are constrained to the set of actors that make sense in the context of the use case diagram. For example, the problem analysis use case diagram is limited to adding existing stakeholders as actors. As the actors in this context must be outside the scope of the application, this is appropriate. Stakeholders are identified in the context diagram for the application, and once defined can be added to the use case diagram.

Move the mouse cursor to the desired location, and select “New Use Case” or “Existing Stakeholder” to add a new diagram element. A new use case added to the diagram with a default name, and a detailed view is opened in a separate window. Change the default name in the detailed view, and add other details as desired. The diagram and the navigation tree will be updated when the detailed view is closed. Existing stakeholders are added as actors and can be updated or renamed by double-clicking their icon in the use case diagram.

Connections between actors, stakeholders or use cases are created as before, selecting the menu option to “Add Connection” over the source of the connection, then selecting the destination object. Connections between actors or stakeholders and use cases default to a UML protocol, represented by a simple line. Connections between use cases default to a dependency as represented by a dashed line with an arrow head to indicate direction. The default line label for a use case dependency is «includes» to represent the relationship between two use cases. The «extends» relationship is not supported by lines, but by abstractive decomposition (nesting) of use cases.

The use case diagrams (and the navigation tree) illustrate the structure of use cases, and their relationships to each other, but are not sufficient to communicate the details of a use case. Use case descriptions are the primary source of information in the requirements model, and provide the necessary details for each use case.

Create Use Case Narratives

The completion of the first set of use case diagrams provides the structure of the use cases, but does not provide the content of the use case descriptions. To delve into use case descriptions, we cross into the next phase of the SEEM process, Problem Analysis. Figure 11 is the activity diagram for the problem analysis phase of SEEM.

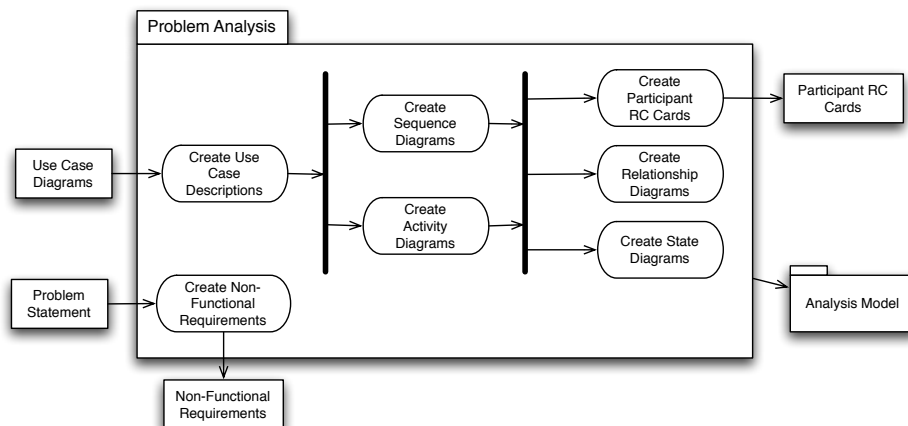


Figure 11 - SEEM Problem Analysis Activities

A well-defined use case represents one complete flow through the “system”, and accomplishes a measurable goal or result for the participants in the use case. Use case descriptions have evolved over the years, and the content of the descriptions is now fairly standard. Requitopia provides for the following content for each use case description:

- ◆ Name - The tag by which each use case is identified.
- ◆ Description - A brief description where the name is not self-descriptive.
- ◆ Preconditions - Conditions that must be true for this use case workflow to be valid.
- ◆ Workflow - A step-by-step description or script of how the use case participants interact to accomplish the results of the use case. Stakeholders, participants, subsystems and objects are color-highlighted for clarity. Each line in the workflow is considered one activity.
- ◆ Results - A list of measurable goals or results of the use case.
- ◆ Extensions - A list of alternate use case workflows that differ from the primary workflow. Supported extensions include variants, or workflows that still achieve the specified results, exceptions, or workflows that do not achieve the specified results, and technology and data extensions. In Requitopia, each variant and exception is a full use case description tied to the parent use case. The technology and data extensions are text-only fields.

Figure 12 illustrates a use case description.

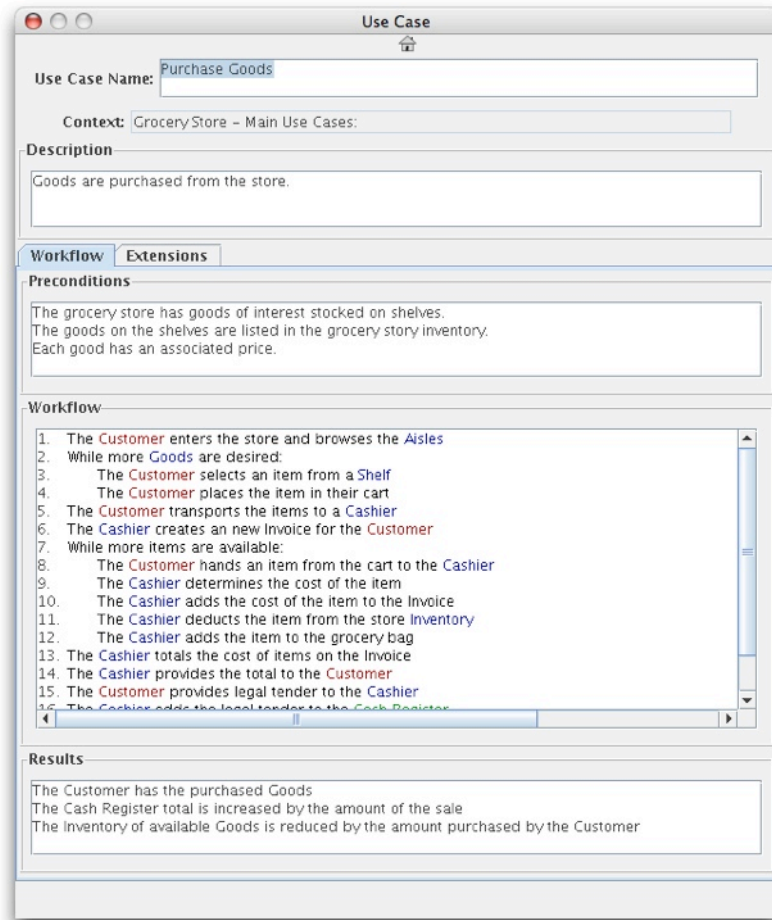


Figure 12 - A Use Case Description

Each of these descriptions are accessed by double-clicking the left mouse button on the use case ellipse in a use case diagram, or double-clicking the node for the use case description in the navigation pane. The name of the use case can be split across multiple lines if desired, and will be rendered on use case diagrams in multiple lines, each centered on the ellipse. Multi-line names allow the modeler to control the aspect ratio of the ellipses on the use case diagrams. The description of the use case is free-form text. The description text, and all other text fields are limited to a finite number of characters. The text fields will stop accepting characters when the limit is reached.

Preconditions are free-form text, but are often represented as a list of pre-conditions, sometimes referring to other use cases that satisfy the preconditions.

The workflow is the “meat” of the use case, and the most difficult portion to create. Each activity in the workflow describes one step in the process in the form of “who does what to whom”. By careful consideration of the “who” and the “what” in each activity, most ambiguities are addressed.

For example, the activity “The money is placed into the cash register”, is less ambiguous if phrased “The cashier places the money into the cash register”.

The flow of activities in the workflow is also crafted to ensure consistency of the flow of control and the flow of information among the participants. For example, in the following two activities, there is an implied step in the middle.

“The cashier hands the invoice to the customer.”

“The cashier places the money into the cash register.”

The implied middle step should be added to ensure consistency of flow, and is simply: “The customer hands money to the cashier.”

To facilitate the identification of the “who” and the “whom” in each activity, Requitopia automatically highlights any stakeholders (dark red), participants (dark blue), subsystems (green) or objects (light blue) in different colors. To help with the creation of the participants, subsystems and objects while creating and updating the workflow, any proper name can be selected, then used to create a new participant, subsystem or object as desired. Simply select the text for the desired name, then select the desired creation option from the popup menu. A new entity will be created and the selected text used as the default name.

As entity names are likely to change over the course of the development effort, the workflow activities will be examined for entity names when a name is changed. If an entity name is changed, and it is used in a use case workflow (or variant or exception), the name will be changed in the workflow to ensure consistency across the entire model. The use of proper names, and the creation of associated entities is strongly encouraged to promote a well-formed model.

Results are free-form text, but are also represented as a list of results. Each of the listed results will have one or more representations in the workflow that describe how the results are achieved.

A use case variant is a use case with a variation in the workflow that still achieves the desired results of the use case. There may be many variants included to represent the different paths that can be taken at each step of the workflow. Variants are added to a use case by selecting the tab for variants and exceptions in the use case description, placing the mouse cursor into the variants portion of the form and selecting the “New Variant” option from the mouse menu. Figure 13 illustrates the extension tab of a use case description.

The screenshot shows a window titled "Use Case" with the following fields and sections:

- Use Case Name:** Purchase Goods
- Context:** Grocery Store - Main Use Cases
- Description:** Goods are purchased from the store.
- Workflow** and **Extensions** tabs are visible.
- Variants:** Customer pays with a credit card
- Exceptions:** The customer has insufficient funds
- Technology:** (Empty field)
- Data:** (Empty field)

Figure 13 - Use Case Extensions

A new variant with a default name is created, and the entire contents of the parent use case is copied into the variant description. The new variant description is opened in a separate window for updating. The name of the variant should be changed to reflect the nature of the variant, and the workflow modified accordingly. Any changes to the workflow from the parent use case will be highlighted in yellow. The format of a variant description is slightly different, primarily in that variants are linked to the parent use case. Figure 14 shows a description for a use case variant.

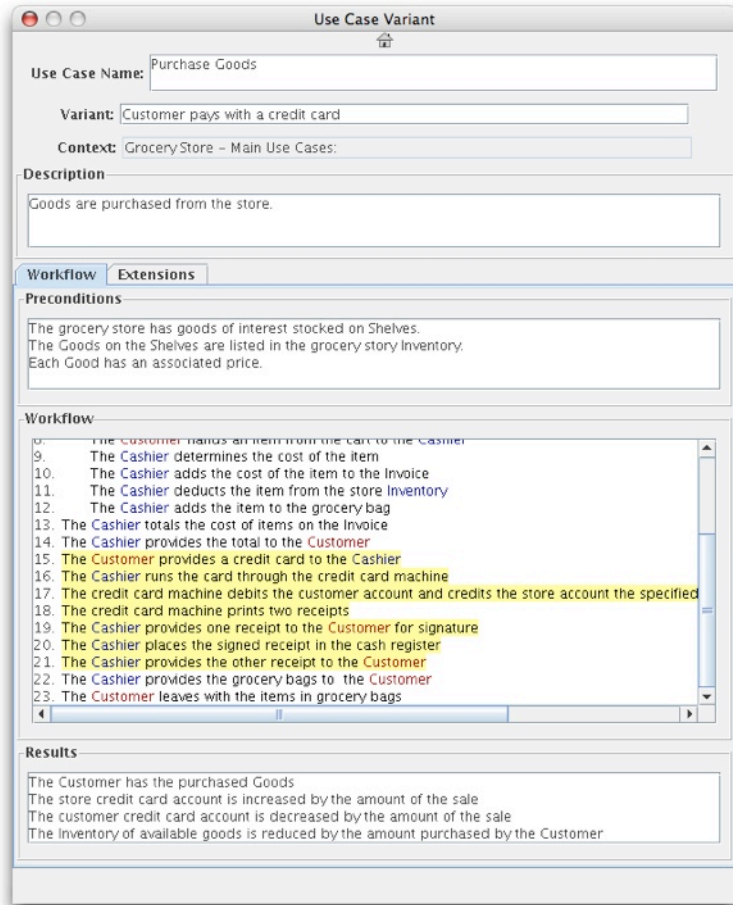


Figure 14 - A Use Case Variant

As the variants are created from the parent use case, we recommend that the default use case, or “greased path” use case be completed prior to creating the variants.

Exceptions are similar to variants, but they represent a workflow that does not achieve the desired results. Exceptions are created in the same manner as variants, and are derived in whole from the parent use case. The default name is changed, and the workflow updated to represent the exceptions. The results for exceptions are also updated as the results for an exception is, by definition, different than the parent use case. Valid results may be “none”.

Technology extensions provide a hint at eventual implementation issues. Data extensions provide a view into the types of information that the use case may need to handle. Both will be factored into the eventual test plans for the entity.

Each use case may have no variants or exceptions, or may have ten or twenty of each. To derive the list of variants, examine each activity in the workflow and brainstorm “what-if” scenarios to identify relevant variations. The same brainstorming is applied to exceptions by considering each activity in the workflow for what can go wrong.

Variants and exceptions provide insight into the complexity of a use case, and eventually drive the test case for the realization of a use case in the final application. As variants and exceptions can be numerous, they are not included in the navigation view, and can only be viewed or accessed through the parent use case.

Refactoring Use Cases

A single use case diagram is a practical container for up to nine use cases. Any more than this, and the reader can not absorb the complexity of the content and relationships in the diagram. To provide for more complex use case models, Requitopia supports hierarchical nesting of use cases into an arbitrarily complex use case model.

As a use case model grows and evolves over time, a single use case can quickly grow to contain more than the recommended 7 +/- 2 use cases. Requitopia provides a simple and effective means to manage this complexity through the creation of abstract use cases. To create an abstract use case in Requitopia, create a use case on a use case diagram and provide it a meaningful name. Once created, drag an existing use case onto the new use case. When one use case is dragged onto another, Requitopia automatically creates a sub-diagram for the target use case (if needed), and moves the dragged use case onto the new diagram. The dragged use case is removed from the current diagram, and the abstract use case ellipse on the use case diagram is adorned with the UML subsystem icon. All connections to the refactored use case are removed. The new use case diagram is opened for viewing. Other use cases can also be dragged onto the abstract use case, and they are moved in turn to the new diagram. Once the use case diagrams are closed, the navigation tree is updated to reflect the new structure of the use cases.

Figure 15 highlights the portion of figure 10 with the abstract use case.



Figure 15 - An Abstract Use Case

The subsystem icon representing the abstract use case, also provides a navigation mechanism to the sub diagram. As with other diagram elements, double-clicking on the use case will open the use case description. If the use case is abstract, double-clicking on the subsystem icon in the ellipse will open the sub-diagram of the use case.

The mechanism for drag and drop is terrific for refactoring a use case into another use case, but cannot be used to promote a use case in the other direction. If a use case requires promotion from a sub-diagram to a parent diagram, another mechanism is provided. In the sub-diagram, move the mouse cursor over the use case you would like to promote. From the mouse menu, select the option for "Promote Use Case". If a parent use case diagram exists, the use case will be remove from the sub-diagram and re-assigned to the parent diagram. All existing connections will be removed.

When refactoring use cases, the descriptions, variants and exceptions will follow the use case regardless of the diagram on which it appears. Requitopia maintains the links between all the use cases and diagrams to ensure correct traceability.

Priority and Status of Use Cases

Use cases are a complete behavior of the system from start to finish, and they provide a measurable result or goal for the user. As such, use cases make a terrific tool for managing a project. To facilitate management by use cases, Requitopia provides two important features for use cases: prioritization and status.

As the requirements modeling effort progresses beyond the initial phases, the number of use cases grows exponentially as each participant resulting from the problem analysis use cases has their own set of use cases derived from the participant RC card. To better manager the explosion of use cases, each use case can be tagged with an optional priority. The priority tag provides a visual reminder of the importance of each, and is used to plan future analysis activities. Three priority levels are provided: low, medium and high. Figure 16 shows the default priority of none, along with the three available levels as seen in the navigation pane.

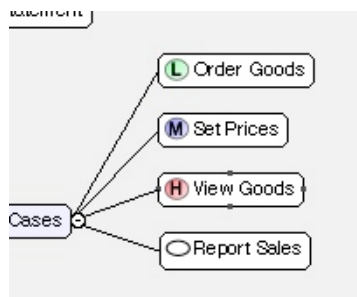


Figure 16 - Use Case Priorities

The priority of a use case is set by moving the mouse cursor over the use case of interest, and selecting the desired priority from the popup menu.

In addition to setting priority of use cases, it is equally useful to communicate the on-going status of a use case. The status of a use case is captured as a percentage of the use case which is complete, and five status levels are supported from 0% to 100% in increments of 25%. The status is represented by a line along the bottom of the use case node in the navigation pane, which is longer and darker for higher completion percentages. Figure 17 illustrates the different status levels.

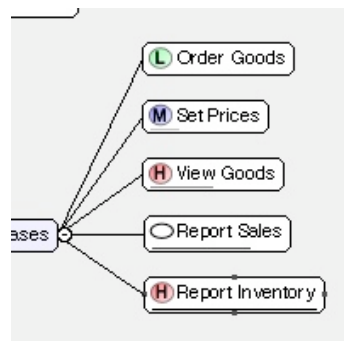


Figure 17 - Use Case Status

Create Participant RC Cards

The goal of the problem analysis phase of SEEM is to identify the participants in the various use cases as defined in the business domain, and to assign them roles and responsibilities according to their role in the use case workflows. The use case descriptions provide the “who does what to whom”, which can be readily interpreted into roles and responsibilities for participants.

When the use case workflows are examined, each of the nouns in the activities (the who, and the whom) can be added to the model as participants, subsystems or objects as described in the section on use cases. The verbs (the “what”) are added as behaviors of the participants.

In Requitopia, the best way to add a participant is via the use case workflow as described above. A new participant can also be added via relationship diagrams as described in that section. Once the participant is identified, the activities in which the participant is an active member can be listed via the “List Activities” menu option. In the navigation pane for the Solution Design, select the participant of interest from the list on the left. The participant is added to the navigation pane on the right. In the navigation pane, move the mouse over the node for the participant and select the menu option for “List Activities”. Each use case workflow will be searched for instances of the name of

the participant, and added to a comprehensive list. The list of activities is presented in a pseudo-RC card format as seen in Figure 16.

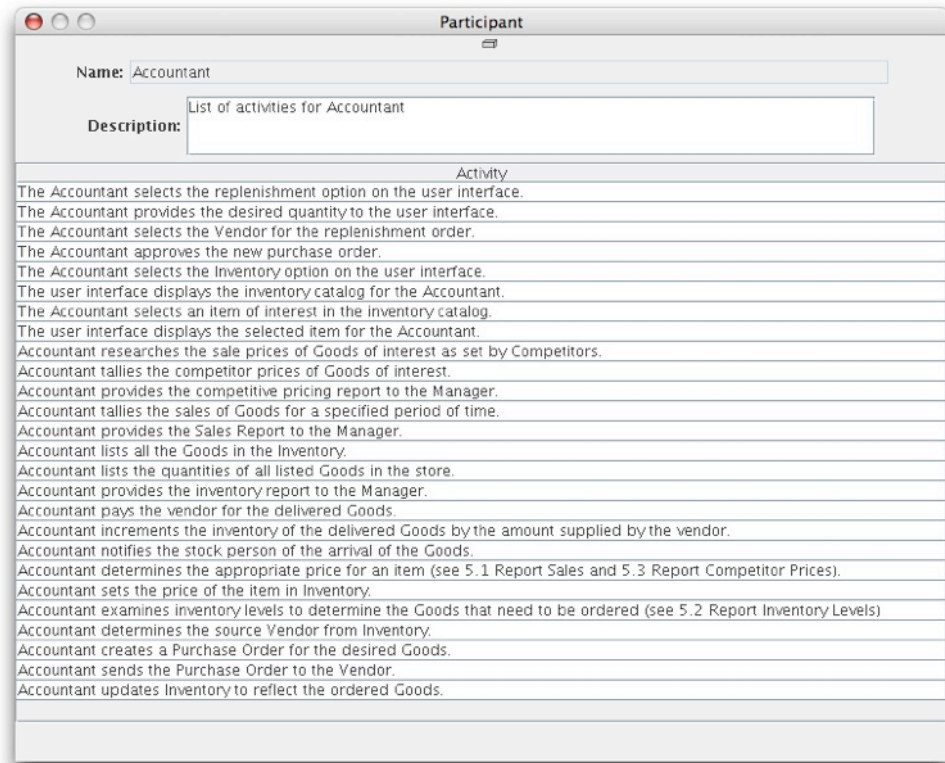


Figure 16 - A list of activities for a Participant

Participant roles and responsibilities are added to the behavior or knowledge section of the participant details. The behaviors for a participant are derived directly from the list of activities for the participant (which are derived from the use case workflow), and are the verbs found in the activities where the participant has a role. The behaviors are pulled from the activity and added as one line in the behavior section of the description. If the participant with the behavior collaborates with another, they are listed as a collaborator in the RC card. If the context of the activity listed is not clear, the use from which it was derived can be opened by using the right-mouse button over the activity in question selecting the option "Open Use Case".

When each of the activities are considered as behaviors for the selected participant, the results are examined and the required knowledge is extracted. The knowledge section of the participant description answers the question: "what does this participant need

to know to carry out their assigned behaviors?” In some cases, other participants may have the required knowledge, so they are listed as collaborations for that line item. Figure 17 represents an RC card for a participant.

The screenshot shows a window titled "Participant" with the following content:

Name:

Entity Type: System Subsystem Class

Description

The person or organization responsible for the financial and inventory aspects of the grocery store.

Knowledge	Collaboration
Know 0-N Goods	

Behavior	Collaboration
Order Goods	Goods, Vendor, Inventory, Purchase Order
Pay Invoices	Packing Slip, Inventory, Purchase Order
Update Inventory	Inventory
View Goods	Inventory
Generate reports	Create reports

Figure 17 - A Participant Responsibility-Collaboration Card

If the name of a stakeholder, participant, subsystem or object is changed, the responsibilities and collaborations are updated to match the new name to ensure consistency across the entire model.

Create Relationship Diagrams

Relationships diagrams capture the static relationships between the participants in the various use cases. The relationship diagrams illustrate the structure of the applica-

tion from a business domain perspective. As there may be many types of relationship diagrams desired, Requitopia supports any number of relationship diagrams for an application. We have found that two diagrams are often sufficient, one for illustrating the aggregation and composition relationships, and one for the inheritance, or classification relationships.

To add a new relationship diagram to a model, move the mouse cursor to the application node in the navigation pane, and select the “New Relationship Diagram” option from the popup menu. A new relationship diagram is created with a default name and added to the navigation tree. The new diagram is opened in a separate window. Figure 18 illustrates a relationship diagram for the aggregation of participants.

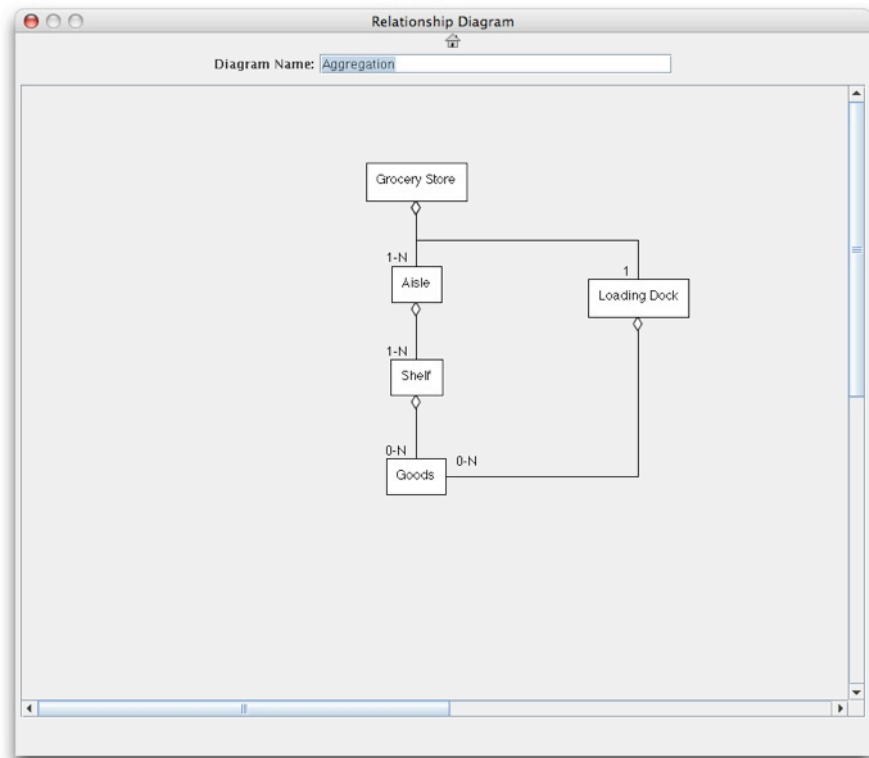


Figure 18 - A Relationship Diagram

Adding participants to the relationship diagram employs the same technique as other diagrams. Move the mouse cursor to the desired location, and utilize the popup menu to select the type of item to add. The diagram supports the addition of new participants, existing participants and notes.

Of more interest in this diagram are the relationships between the participants. Relationships are added by using the same technique as other diagrams to create a connection between two participants, then changing the characteristics of the connection. To add a connection, move the mouse over the source participant and select the “Add Connection” from the popup menu. Then select the destination participant. Requitopia will add a new connection between the participants.

Requitopia currently support five different types of static relationship on relationship diagrams. The various available types of relationships are accessed by moving the mouse cursor over the connection line of interest, and selecting the desired relationship from the popup menu. The default relationship is a UML aggregation, represented by a simple line with a white diamond at the source of the relationship. Aggregation represents loose containership where the source participant “contains” the destination participant.

A stronger type of containership is composition, which has a similar representation but a black diamond which implies lifecycle dependency. For composition, neither participant is viable without the other.

Association is a weaker form of relationship which merely implies some type of relationship between the participant exists, but does not add additional information. Association is a simple line with an open arrow head.

Dependency is another relationship already seen between notes and other diagram elements. Dependency implies that a change in one participant may effect the dependent participant. Dependency is represented by a dashed line with an open arrow head.

The last supported relationship is inheritance. Inheritance is used to indicate a taxonomy of types of participants. Inheritance is represented by a simple line with a white diamond at the source.

In addition to the relationships represented as lines connecting participants, the lines can have optional cardinality labels to indicate the number of participants in the relationship. Typically, only aggregation and composition relationships include cardinality. The relationship connections can also have an optional label on the line to help qualify the relationship. The optional components for a relationship are accessed by double-clicking the left mouse button on the relationship. The connection dialog presents the options available for that type of connection.

Requitopia provides sophisticated layout algorithms that automatically route the connecting lines on diagrams and attaches them to the drawing elements according to their relative geometry. The automatic layout allows for easy repositioning of diagram elements without having to manually move the associated connections. For relationship diagrams, the modeler may wish to override the default layout algorithms to clarify relationships. The connection options dialog accessed via a double-click on the connection of interest provides for a fixed connection location. Selecting the option for a side, top or bottom will lock that end of the connection to the designated side. The option for auto-

layout can also be reset as desired. Note that overriding the automatic layout option may result in very strange diagrams! Figure 19 illustrates the option box for an aggregation connection.

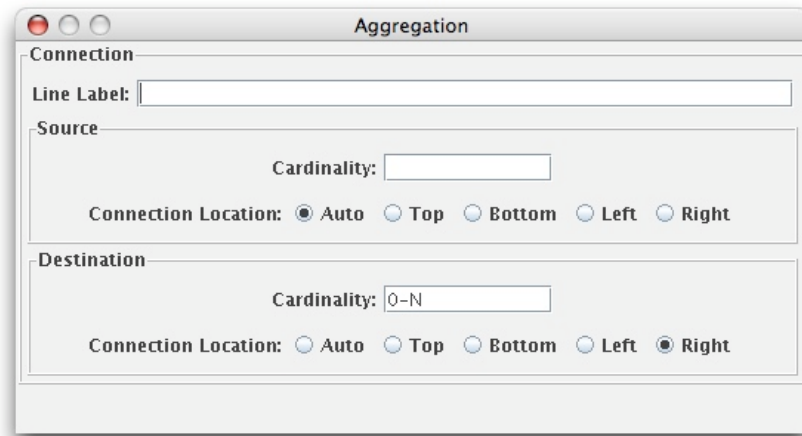


Figure 19 - Connection Options

Requitopia also support self-relationships, created by connecting a participant to itself. The layout options also apply as described above, and can be used to tailor the visual presentation of the relationship to your liking.

Relationship diagrams can also be added to other modeling entities in the other abstraction layers.

Next Steps

The discussion thus far has centered on the problem analysis portion of the SEEM methodology. Requitopia is currently geared toward this portion of the process, but is certainly not limited to the problem analysis. Per SEEM philosophy, the goal of problem analysis phase is the identification of participants in the problem domain, and assigning them appropriate roles and responsibilities. The participant descriptions, or participant RC cards accomplish this goal. To take the model to the next level of detail, the participants are each examined as a stand-alone system, and the same modeling process is applied.

The goal of the solution design phase is the identification of the architectural components of the system, and assigning them roles and responsibilities. At the solution design level, technology is introduced at a high level, and each technology component is described at a high-level of abstraction in the context of each participant.

Requitopia supports full traceability between phases by providing a complete model as a sub-model of the previous phase. For the solution design phase, each participant (and actor) is presented in a list of participants to the left of the solution design navigation pane. Selecting a participant with the mouse brings up the navigation model for that participant. At the solution design phase, each participant is now considered as the root of the model, and all the same diagrams and operations can be applied within the scope of the chosen participant. In general, the behaviors defined on a participant description / RC card is mapped into a set of use cases for the participant. Each use case is then described as before, but the workflow now identifies the architectural subsystems in the workflow activities. All other diagrams can be applied, including relationships diagrams, problems statements and context diagram as appropriate for the scope of the participant. Figure 20 represents one possible sub-model as shown in the navigation pane for the system design phase.

Requitopia - Quick Reference Guide

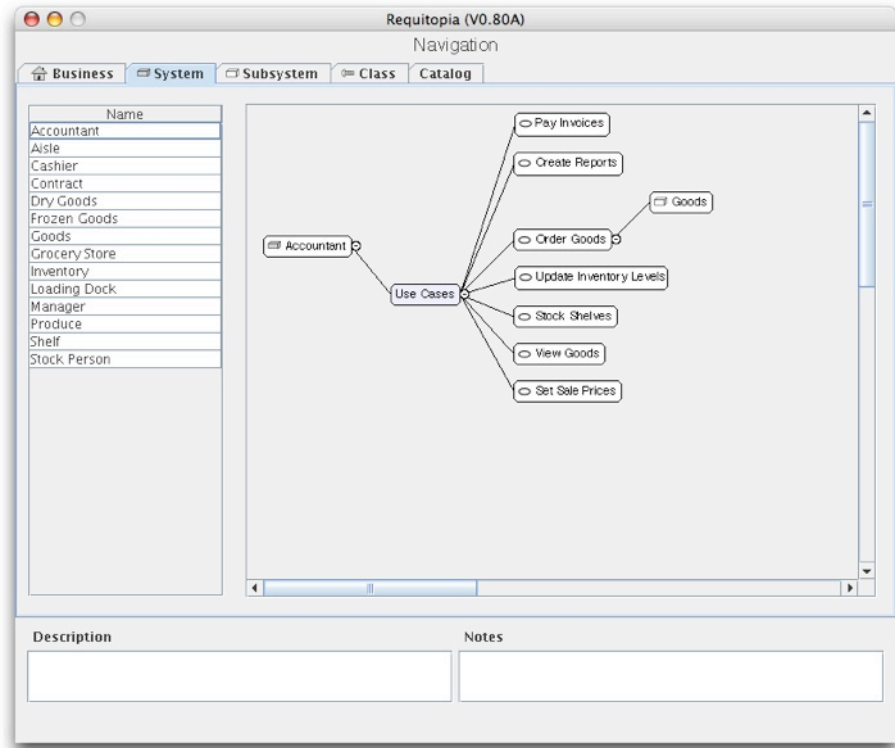


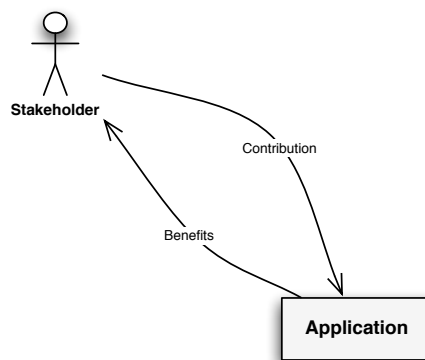
Figure 20 - Solution Design

In a similar vein, the subsystems identified in the solution design phase can be driven to another level of detail via the implementation tab on the navigation pane. The use cases resulting from the implementation phase provide detailed workflows for each component of the architectural subsystems, and can be utilized as functional specifications for development of the source code, business process descriptions, or other final artifacts of the development process.

Notation Reference

UML is a comprehensive, powerful notation that can express a wide range of information. Requitopia has selected a simple subset of UML to facilitate readability by UML lay-persons. The following graphics represent the limited set of UML notation employed by Requitopia.

Context Diagrams

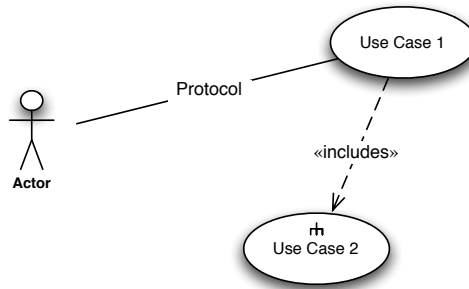


Context Diagram Notation

The primary purpose of the context diagram is to identify the stakeholders for an application. Context diagram notation includes stakeholders, the application, and the relationships between them. Stakeholders are represented by the UML actor icon. The application is represented by a class rectangle, and the contributions and benefits are represented by labeled associations between the two. Each of the drawing elements can be double-clicked to reveal their details.

In Requitopia, context diagrams occur at each level of abstraction. In the problem analysis model, the focus of the context diagram is the application, and the application is included in the center of the diagram. Stakeholders at this level are persons, organizations or systems outside the scope of the application. At the solution design level, the focus of the diagram is a participant, which takes the place of the application at the center. At this level of abstraction, stakeholders may be outside the scope of the application, or other participants outside the scope of the current participant. In the implementation model, the focus is a subsystem or object. Here the stakeholders can be external to the application, other participants, or other subsystems or objects.

Use Case Diagrams

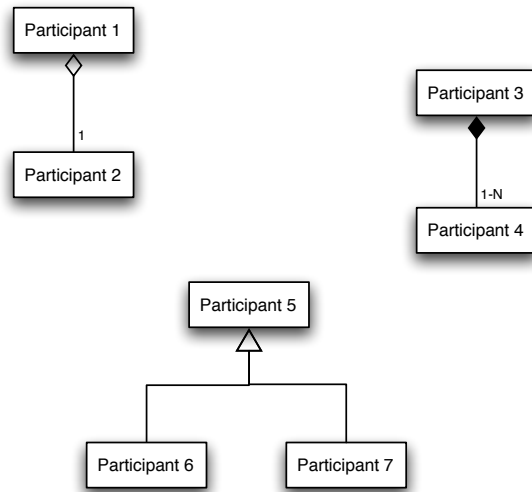


Use Case Diagram Notation

The primary purpose of the use case diagram is to identify the behaviors for an application. Use case diagram notation includes actors, use cases, and the relationships between them. Actors are represented by the UML actor icon. Use cases are represented by ellipses. Relationships between actors and use cases are UML protocols, and are represented by simple lines between them. Relationships between use cases are UML dependencies constrained by the «includes» relationship, and are represented by dashed, open-headed arrows with the «includes» label. Abstract use cases include the subsystem icon as seen in Use Case 2 above. The subsystem icon indicates that another diagram exists for Use Case 2, and includes more details about the available behaviors. In Requitopia, each of the diagram elements can be double-clicked to reveal the details of the entity. In addition, the subsystem icon within a use case ellipse can be double-clicked to open the associated sub-diagram.

In the problem analysis model, actors in the diagram are limited to the stakeholders as defined on the context diagram for the application. Use cases represent the desired behaviors for the entire application, and are often organized into a meaningful hierarchy. In the solution design model, use cases represent behaviors for a single architectural participant as defined by the use cases in the problem analysis model. Actors in the solution design model use cases diagrams can be stakeholders or other participants. In the implementation model, the use cases represent desired behaviors for the subsystems as defined by the solution design model. Actor at this level can be stakeholders, participants, or other subsystems or objects.

Relationship Diagrams



Relationship Diagram Notation

The primary purpose of the relationship diagram is illustrate the static relationships between participants, subsystems, or objects in the model. Static relationships do not change over time, and exist outside the operation of the application under development. There are three relationships of interest modeling in Requitopia: aggregation, composition, and inheritance.

Aggregation is represented by a white diamond at the parent end of the relationship, and indicates a loose container relationship. Aggregation is often referred to as the “has-a” relationship. In the example above, Participant 1 contains exactly 1 of Participant 2, but is not responsible for Participant 2. Aggregation implies independence of the two related entities, where each entity is viable without the other. An example of aggregation is the idea of a pair of boots in the trunk of a car. Take the boots out of the trunk, and the boots and car are still viable objects.

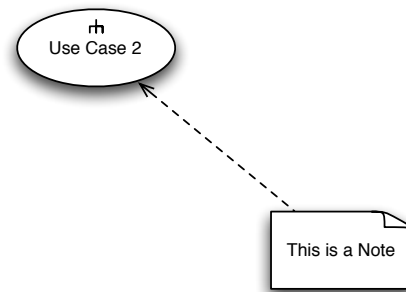
Composition is represented by a black diamond at the parent end of the relationship, and indicates a tight container relationship. Composition is often referred to as the “owns-a” relationship. In the example above, Participant 3 contains 1 to many of Participant 4, and is responsible for Participant 4. Composition implies that the entities in the relationship are inextricably bound to each other. As an example, a car contains an engine. Remove the engine from the car, and neither are viable objects. From a software perspective, composition implies lifecycle dependency. Recursive relationships, or a relationship to an entity itself is also supported.

Inheritance is represented by a white triangle at the parent end of the relationship. Inheritance indicates classification or taxonomy of entities, and is often referred to as

the “is-a” relationship. In our example, participants 6 and 7 are the same type of entity as Participant 5, but they each include additional knowledge or behavior for specialization. As an example, a Ford or Toyota are Automobiles.

Each of the relationships in the diagram can be double-clicked to reveal the details, and the type of relationship is determined via the popup menu. The lines include an optional label, and the cardinality of either end of the relationship can be specified. By default, each relationship line is automatically routed based on the relative geometry of the connected entities. If the auto-layout algorithms do not provide suitable results, the connections points for lines can be explicitly specified via the connection details. Be aware that manual layout options can result in some interesting diagrams!

Notes



Notes

Notes can be added to any diagram in Requitopia, and can be optionally connected to any other entity. Notes are free-format text, and can be created with embedded carriage returns so the author can control the aspect ratio of the result. Connections from notes to other entities default to a UML dependency represented by a dashed line with an open-headed arrow.

Glossary of Terms

Abstract Use Case	A use case acting as a container for more detailed use cases. The name of the abstract use case reflects the behavior of each of the detailed use case children. an abstract use case is identified by the UML subsystem icon on use case diagrams.
Activity List	A collection of activities for a selected participant, sub-system or object as extracted from all the use case workflows. The activity list is derived from a text-search of the workflow from each use case in the application. If the name of the selected entity exists in the workflow, the activity is extracted into the activity list.
Actor	A participant in a use case workflow external to the scope of the application. An actor may be a person, organization, system or artifact.
Aggregation	A static relationship where the parent or source in the relationship contains the destination elements in the relationship. Aggregation is loose coupling in that each participant in the relationship remains viable when separated. Aggregation is characterize by a "has-a" phrase.
Application	The "system" under analysis. This may be a software application, a business workflow, or an embedded product.
Automatic Layout	A feature that determines the layout and connection points for relationships between drawing entities.
Association	A non-specific static relationship between two modeling elements.
Classification Node	An arbitrary collection node for navigation panes to allow for grouping of information for participants. Classification nodes can only be added as children of participants in the Solution Design navigation pane.
Composition	A static relationship where the parent or source in the relationship contains the destination elements in the relationship. Composition is tight coupling in that each participant in the relationship is no longer viable when separated. Composition is characterize by a "owns-a" phrase.

Context Diagram	A graphical representation of the environment into which the application will be deployed. The goal of the context diagram is to identify all the relevant stakeholders.
Dependency	A static relationship indicating that a change to the destination participant will likely require a change to the source participant.
Elevator Pitch	A brief description suitable for sharing with fellow passengers on an elevator ride to your floor.
Inheritance	A static relationship indicating a classification taxonomy between elements. Inheritance is characterized by a "is-a" phrase.
Mindmap	A diagram to represent and navigate the structure of the requirements model.
Navigation	The mindmap representation of the requirements model for navigating and viewing the model contents.
Participant	A participant in a use case workflow internal to the scope of the application. Participants are defined in the problem analysis model and elaborated in the solution design model. Participants form the traceable connection from the problem analysis model to the solution design model.
Problem Statement	A short description of the problem to be solved by your application.
Relationship Diagram	A diagram illustrating the static relationships between participants in the model.
Stakeholder	A person, organization or legacy system with a stake in the application under development. Stakeholders are outside the scope of the development organization. Stakeholders are defined in the problem analysis model.
Stakeholder Profile	A description of a stakeholder and their relationship to the application
Subsystem	A component of the requirements model defined in the solution design model, and elaborated in the implementation model. Subsystems form the traceable connection from the solution design model to the implementation model.

UML	The Unified Modeling Language, an industry standard notation which defines the boxes, arrows and shapes of the content of diagrams.
Use Case	A complete behavior of the system under consideration from start to finish, with a well-defined goal or result.
Use Case Diagram	A diagram illustrating a collection of use cases and their relationships