UML-ERP Workshop

Writing an SDD with Enterprise Architect

Data Driven Report Engine

Jose Zouain

UML - ERP Workshop
Writing a Software Design Document (SDD)
Data Driven Report Engine

Designing an Enterprise Resource Planning (ERP) System with the UML modeling tool, Enterprise Architect.

Jose Zouain

This book is for sale at https://leanpub.com/umlerpworkshop-report-sdd
This version was published on 2022-October-25

Table of Contents

Introduction

Anyone trying to develop a system within their organization has been faced with many competing demands. Demands come from those responsible for establishing the guidance for the organization, the industry standards that are designed to assist in establishing direction, the features provided by the available tool sets in this space, and those who are impacted by the direction as they attempt to do their daily work.

Many standards are large and difficult to implement, but once you have adopted the way you will be working, most standards should make your work easier to do, not harder. I will teach you how I have implemented in different organizations an SDD (Software Design Document), making changes as your experience grows using a robust tool, Enterprise Architect from Sparx Systems and Zeta Pro ™ examples. Zeta Pro ™ is an Enterprise Resource Planning System developed by Zeta Concepts, Inc.

Once the main Classes and other diagrams are essentially complete, a lot of work remains to be done on the structure of the SDD. There is no intention to release the code of Zeta Pro ™, each developer writes code differently. Coding is like writing; every developer has their own way of coding just as a writer does. Showing the diagrams of Zeta Pro from my point of view is better than showing the code. The system is not perfect and like any software development project, bugs can be found at runtime, unexpected cases or maybe design flaws can occur, but thanks to UML the process of revealing all of this is much easier.

Now that I have talked about the scope of the project I have to mention the tools that I have used:

- WithClass from Micro Gold Software
- Visual Paradigm from Visual Paradigm International
- Rational Rose from IBM Corporation
- Enterprise Architect from Sparx Systems - the tool I have been using for the last eighteen (18) years. I agree with the Enterprise Architect manual which states, Enterprise Architect is an intuitive, flexible and powerful UML analysis and design tool for building robust and maintainable software. From requirements gathering, through analysis, modeling, and implementation and testing to deployment and maintenance, Enterprise Architect is a fast, feature-rich, multi-user UML modeling tool, driving the long-term success of your software project.

As you get more comfortable, you will naturally start to find the tool better to work with. There is no shortage of articles about the right or best way to create an SDD, and there are lots of potential approaches, you can learn the concepts by completing inter-actives exercises, UML-ERP Workshop offers you that.

Copyright and Disclaimer

Following you will find the Statement of Confidentiality and Copyright that is included in each SDD document created. A Trademark and Internal section is also included. I would be glad to receive any kind of feedback to correct future updates of this eBook; all the information presented here is subject to change without notice.

About the Book

A lot of books have been written about the Unified Modeling Language (UML) with very good examples, although I have never seen a real life example of a working system on the market, but maybe I am wrong. In this eBook I will reveal all the business processes, Class Diagrams, Sequence Diagrams and Data Model on how we created a Data Driven Report Engine for the Zeta Pro™ System.

As a UML evangelist I have decided to show everybody the system I have designed and helped develop, Zeta Pro ™. These are open venues for both software developers who are learning how to develop or have been creating systems for quite some time.

Zeta Pro ™ has been in the market for more than 20 years. It is an Enterprise Resource Planning (ERP) System originally developed in the 90s for the DOS platform and later transferred and developed in an Object Oriented Programming language for the Windows platform using UML.

This eBook is for the Technical Analysts, Data Modeler and/or System Architect who will learn how to create a Software Design Document (SDD), identifying the tables and indexes (Primary and Foreign Keys) needed for the system, what classes to create in order to develop a system in any OOP (Object-Oriented programming) language of their choice, and how each class interacts with other classes in a Sequence Diagram.

Zeta Pro ™ (the system) is huge, big, Grande; therefore I will be releasing several eBooks covering a complete module or a section of a module. Also for those readers who want to see the system in action and follow the diagrams, send me an email for a single Zeta Pro ™ user license for one (1) year free.

Go to my website www.zetaconcepts.com and register to receive a single user license or send me an email to support@zetaconcepts.com requesting the single user license.

eBooks published:

- Inventory Control – BRD
- Data Driven Report Engine – SDD

In this eBook and subsequent ones I will show you how to design a system. I did it - why can’t you do the same? The only advantage that you will have over me is that no one supplied me with all the UML diagrams to design it as we are doing with all these eBooks.

You will leverage the techniques from these eBooks by discovering how UML can help you identify opportunities for your personal goals.

This is a complete, practical, hands-on book which consists of step-by-step instructions for creating an Enterprise Resource Planning System (ERP), covering everything you need to know about business processes, business rules, system processing and visualizing data at the Enterprise Level.

About the Author

José Zouain is the original author and principal maintainer of Zeta Pro ™, and has been writing code for about 20 years. As an UML evangelist since 1999 and a Sparx Enterprise Architect ™ Guru since 2002, José has been planning and discussing this project with friends for several years, and finally has decided to go ahead and move forward with it. This eBook is a living breathing entity. Every eBook is a piece of history up until the publication date, so we welcome comments for further refining. Each subject could fill an entire eBook on itself. We struggle to include all subjects in as short as possible while still getting the point across.

An independent consultant with over twenty-five (25) years of experience in different phases of the Software Development Life Cycle (SDLC), managing advance programming, system maintenance and problem determination, he has fulfilled a number of roles. Among his responsibilities he has held different titles such as Senior Technical Analyst, Senior Business Analysts, Project Manager, User Acceptance Manager, and OOP (Object Oriented Programming) Developer.

Online

Website of the author

www.zetaconcepts.com

www.zetapro.com

Email of the author

support@zetaconcepts.com

How this Book is Organized

Chapter 1: “The Basic Knowledge”, presents the concept of Enterprise Resource Planning (ERP) and Unified Modeling Language (UML), also presents what is a Software Design Document (SDD). If you are proficient in UML, feel free to skip this chapter. There are, however, some important definitions and sections that highlight some areas that are relevant to the rest of the eBook.

Chapter 2: “Managing Classes with EA”, is an overview of how to create classes, class diagrams with its connection and sequence diagrams with its messages.

Chapter 3: “Enterprise Architect Hints”, defines all the hints needed to succeed and all the knowledge accumulated over the last 15 years that will make your life easier using Enterprise Architect as the tool.

Chapter 4: “Data Modeling”, is an overview of how to create data model diagrams with its connections.

Chapter 5: “Preface to the SDD”, explaining how is the SDD divided and why we have used this approach based on hints provided.

Chapter 6: “Data Driven Report Engine – SDD”, lastly what everybody wants to see, is the Software Design Document (SDD) for a Data Driven Report Engine that Enterprise Architect generates from all the information entered and diagrams created based on the templates we have designed.

Chapter 7: “Class Design”, explains the class diagram by grouping the classes needed for the data driven Report Engine.

Chapter 8: “Component Design”, explains an overview of the architectural breakdown of the system. The Report Engine is to simplify and centralize all reports with a data driven system.

Chapter 9: “Data Design”, explains the Data Model (tables) used for the data driven Report Engine.

Chapter 10: “Enumeration”, explains the enumerations linked to the Class BoxesWindow.

1 The Basic Knowledge

In this chapter we will be briefly touching on the definition of ERP, UML, and SDD.

Enterprise Resource Planning (ERP)

ERP is an industry term for the broad set of activities that helps a business manage the important part of its business. ERP software applications can be used to manage product planning, parts purchasing, inventories, interacting with suppliers, providing customer service, and tracking orders, it can also include application modules for financial accounting and fixed assets management. Typically the ERP system is integrated with a relational database system.

Enterprise Resource Planning Systems are typically developed based on a large framework. In reality, this framework is created and designed by different users (analyst, architects, developers, data modelers) by providing a solution to a customer. It is therefore of interest to make the development process as efficient as possible and one way to do this is to provide them with the specialized models. This book presents our ideas for such domain specific modelling environments, where the graphical notation is inspired by UML, but incorporates specific aspects unique to the specific framework we are considering.

IT leaders are fed up with the perpetual-license-plus-maintenance approach to enterprise software purchases, and they are tired of running on the “upgrade” treadmill simply for the sake of keeping their software maintenance status current. Here is a look at how you can break free and develop your own ERP system, with all the screens, business rules, data models and processes; and use it either internally or in the cloud.

The deployment of an ERP system can involve considerable business process analysis, employee retraining and new work procedures; you can’t simply turn it on and expect it to run without training.

You can use the diagrams to visualize the system from different perspectives, as no complex system can be understood in it’s entirely from one perspective. Diagrams are used for communication, for example, a class may appear on one or more Class Diagrams; it might be represented in a State Machine Diagram, and have instances appeared on a Sequence Diagram, and each diagram will provide a different perspective.

Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering which is designed to provide a standard way to visualize the design of a system.

It was created and developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software during 1994–95, with further development led by them through 1996. In 1997 it was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since. In 2005 the Unified Modeling Language was also published by the International Organization for Standardization (ISO) as an approved ISO standard. Since then it has been periodically revised to cover the latest revision of UML.

A lot of books have been written about UML, however we will briefly touch on the following topics as needed for this project, defining them in simple way:

- **Functional Requirement** – What a system must do, the functions it should perform.
- **Non-Functional Requirement** – What the system must be, it describes the system attributes.
- **Use Case** - Is a sequence of actions or steps performed by an actor and the system, which yields an observable result. A Use Case describes the functionality that will be built in a proposed system, for example adding an Inventory, updating an Inventory and deleting an Inventory are all Use Cases combined into one Use Case.
- **Actors** – Anything that interfaces with the system, a human or machine entity that interacts with the system to perform meaningful work.
- **Activity Diagram** – Is a graphical representation of the flow of steps involved in the execution of a system.
- **Pre-Condition** – Is the state that the system must be in for the use case to be able to start.
- **Post-Condition** - Lists possible states that the system can be in at the end of the use case execution.
- **Trigger -** Is the initiator of a use Case, it is what causes the Use Case to start.
- **Basic Path** – Also called the Happy Path is an unconditional set of steps that describe how the use case goal can be achieved and all related stakeholder interests can be satisfied.
- **Alternate Path** - Is a step or a sequence of steps that achieves the use case’s goal following different steps than described in the main success scenario. But the goal is achieved finally. RESULT POSITIVE.
- **Exception Path** - Is anything that leads to NOT achieving the use case’s goal. RESULT NEGATIVE. Just to be clear: **Cancel is** not an Alternate Path, it’s **an Exception Path**.
- **Includes relationship** - A relationship between two use cases in which one use case 'includes' the behavior. This is indicated where there a specific business use cases which are used from many other places.

- **Extends relationship** - A relationship between two use cases in which one use case 'extends' the behavior of another. Typically this represents optional behavior in a use case scenario, which is adding behavior to a use case without changing the original use case.
- **Invokes relationship -** Indicates that a use case at some point causes another use case to happen. One use case invokes another in the same way a function invokes a peer function.
- **Precedes relationship** - Indicates that a use case must be completed before another use case can begin. Indicates that one use case needs to precede another within a logical sequence.
- **Class Diagram -** is a Structural UML diagram type of a model that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
- **Sequence Diagram –** is a Behavioral UML diagram type showing object interactions arranged in time sequence. It represents the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
- **State Machine Diagram -** is a Behavioral UML diagram type which shows discrete behavior of a part of a designed system through limited state transitions, specifying the sequence of events that an object goes through during its lifetime in response to events.
- **Communication Diagram -** is a Behavioral UML diagram type which shows interactions between objects and/or parts using sequenced messages in a free form arrangement.
- **Deployment Diagram -** is a Structural UML diagram type which models the run-time architecture of a system. It shows the configuration of the hardware elements (nodes) and shows how the software elements and artifacts are mapped onto those nodes.
- **Data Model –** is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities. A data model explicitly determines the structure of data. Usually data models are specified in a data modeling language. A data model instance may be one of three kinds:
    - **Conceptual data model** – is the most abstract form of data model. It is helpful for communicating ideas to a wide range of stakeholders because of its simplicity.
    - **Logical data model** – it help to define the detailed structure of the data elements in a system and the relationships between data elements. They refine the data elements introduced by a Conceptual data model and form the basis of the Physical data model.
    - **Physical data model** – is a representation of a data design as implemented in a database management system.

Software Design Document (SDD)

A Software Design Document (SDD) also called Software Design Description or Software Design Specification, is a written description of a software product that a software designer or architect writes in order to give a software development team overall guidance to the architecture of the software project. There are two kinds of design documents called High Level Design Document (HLDD) and the Low Level Design Document (LLDD). When writing a software design document, your goal is not to impress the stakeholders; it is to clearly and accurately describe your solution to your team. The best way to achieve that is to keep your language plain and illustrative. Use simple words, short sentences, bullet lists, and helpful diagrams wherever you can. Get everyone involved in your team and treat the SDD as a living document, by updating the documentation as you make changes to the original solution and keeping all the stakeholders in the same page.

The process of creating an SDD consists of the following steps, after receiving the BRD with the requirements, use cases, activity diagrams, wireframes or user interface (UI) design, the Architect and/or the Technical Analyst in charge of creating the SDD must go over each step. For the purpose of this SDD will concentrate on the High Level Design Document (HLDD), but we could add more which adds the necessary details to the current project description to represent a suitable model for coding.

1. Review the requirements with the development team
2. Review and discuss the user interface or wireframe based on requirements
3. Change or accept the Use Cases, Activity Diagrams and Business Rules
4. Define with the development team the classes and create the Class Diagram
5. Define with development team the Sequence Diagrams
6. Define the tables and create the Data Model

Any of the steps must include if any changes to the BRD, the current and the proposed solutions with a sign-off from the stakeholders; and if requested an Alternative Solution with the pros and cons of the alternatives, like 3rd-party or open source solutions.

Whether the design document was worth the time and effort you invested in writing it is not judged by whether the project gets successfully implemented. Writing the SDD forces you to think through different parts of the technical architecture of your solution. Feedback from your team helps you identify the riskiest part of it. If it turns out to be impossible to implement, you only spent a few days on the process instead of wasting months only to abort the project later.

A SDD contains the following sections:

- Cover page
- Copyright and Contact Information (optional)
- Introduction (Purpose, Glossary, Overview, Scope, References, Timelines, Revision)
- Architectural Design
- Interface Design
- More to follow in the eBook…

2 Managing Classes with EA

Project begins by gathering requirements from various stakeholders (people who have relevant interest in the old or new systems). A key task for the architect and/or technical analyst is to refine and confirm the requirements and agree with the business that they represent exactly what they want of the new or modified system by creating new diagrams.

Topics for this Chapter

Creating a Class Diagram

Adding Elements to a Class Diagram

Adding Attributes to a Class

Adding Operations to a Class

Assigning Enumerations to a Class Attribute

Connecting Class Elements in the Class Diagram

Creating a Sequence Diagram

Adding Elements to a Sequence Diagram

Connecting Elements in a Sequence Diagram

Inserting Combined Fragments in a Sequence Diagram

(Describing the different types of combined fragments with examples, later is applied at the SDD).

3 Enterprise Architect Hints

Following are the EA hints that I can give to you when creating a BRD and/or an SDD. The goal of this eBook is to give you the essentials so that you will know how to read UML, use UML and create UML based diagrams. More information can be found in other books available on UML, but you can grasp the visual world of UML and how you can apply this knowledge to your projects based on what you see here.

Topics for this Chapter

4 Data Modelling

Data Modeling is the act of exploring data-oriented structure.

Topics for this Chapter

5 Preface to the SDD

In the next chapter will be showing the Software Design Document (SDD) for a Data Driven Report Engine included at the Zeta Pro ™ system. An Enterprise Resource Planning (ERP) System developed by Zeta Concepts, see the use case diagram for the nine (9) modules and the setup..

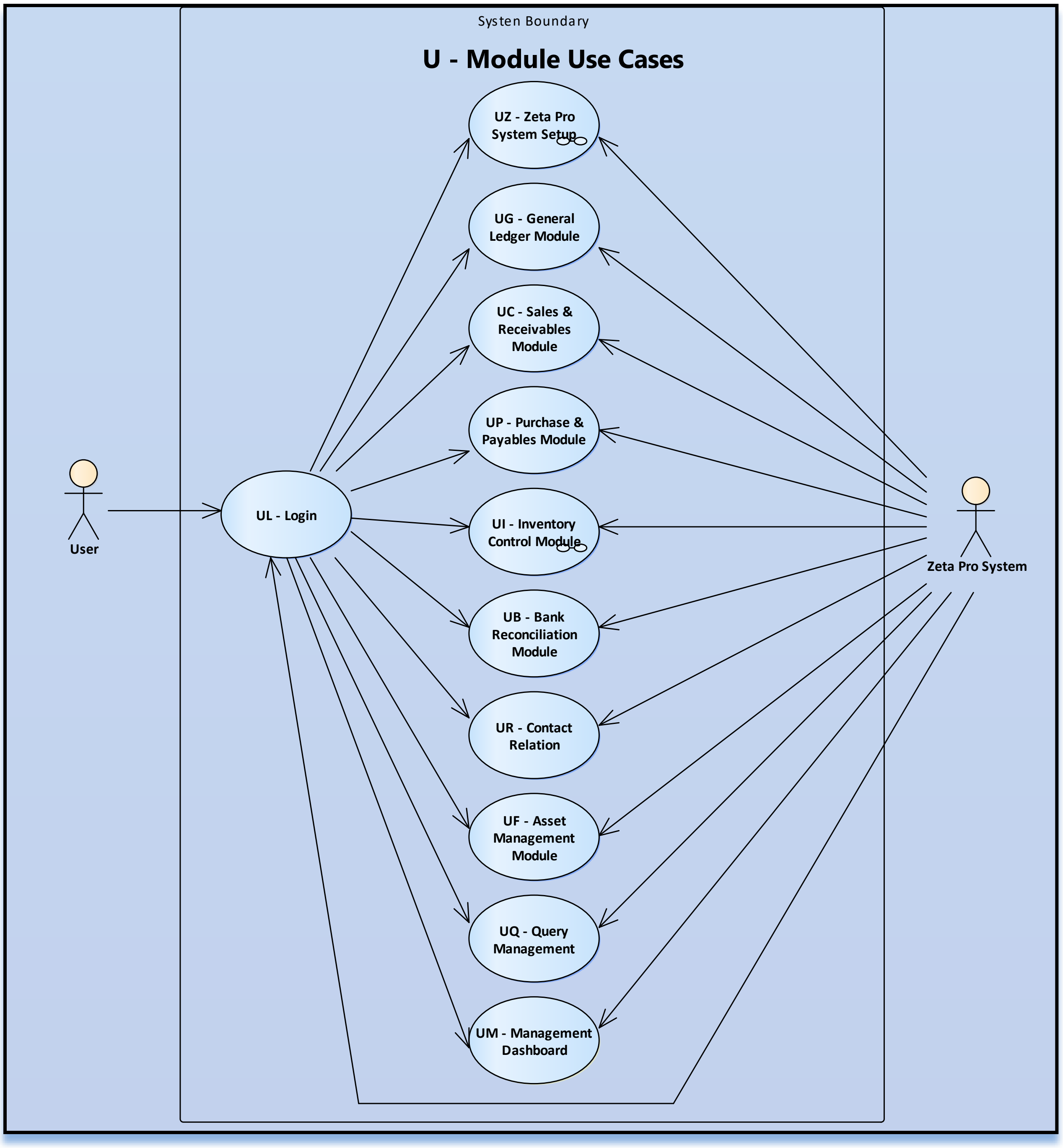

Systen Boundary
U - Module Use Cases
UZ - Zeta Pro System Setup
UG - General Ledger Module
UC - Sales & Receivables Module
UP - Purchase & Payables Module
UI - Inventory Control Module
UB - Bank Reconciliation Module
UR - Contact Relation
UF - Asset Management Module
UQ - Query Management
UM - Management Dashboard
UL - Login
User
Zeta Pro System

Figure 1: Zeta Pro Use Case

The Report Engine

To make it easy for the reader and to understand what the report engine is, will make a brief scenario what the stakeholders of the system wanted. Any ERP system has a lot of reports, this particular one, Zeta Pro™, originally had 1,300 and currently has 1,800 reports and growing. They wanted an engine to run all the reports without a lot of development time in the future.
After a lot of meetings with the team, including Business Analyst, Technical Analyst and the Architect; they have decided in order to meet the stakeholder's expectations that they would have to come up with some type of data driven design, which needs no development or programming time when done, by only adding rows to the tables to add new reports to the system.

Describing the ERP system and taken from the eBook **Inventory module - BRD**, in the prior page is a Use Case Diagram representing Zeta Pro™ nine (9) modules, at the same time each module has four (4) sections (Process, Maintenance, Reports and Setting). The **Report** section in each module can have from 50 to 300 reports per module. Let me show you another use case diagram representing the sections.

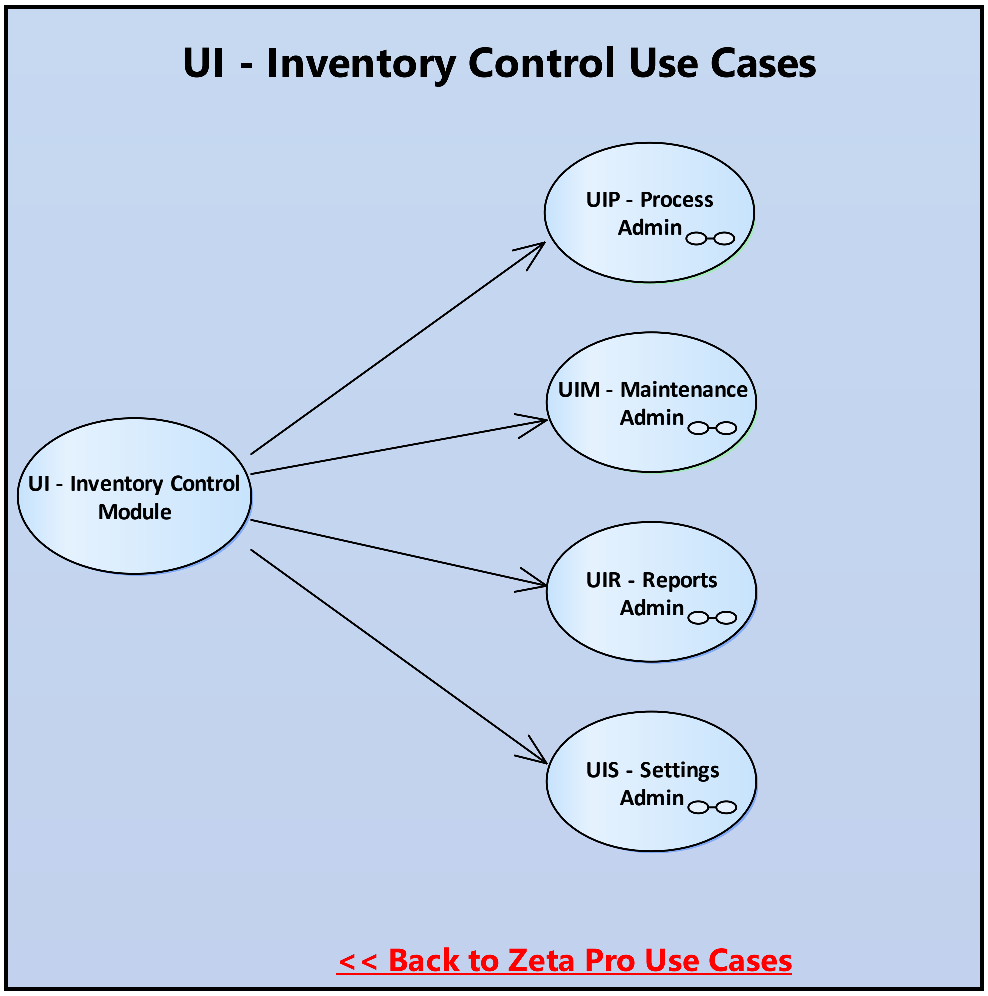

UI - Inventory Control Use Cases
UIP - Process Admin
UIM - Maintenance Admin
UI - Inventory Control Module
UIR - Reports Admin
UIS - Settings Admin
<< Back to Zeta Pro Use Cases

Figure 2: Inventory Control Use Case

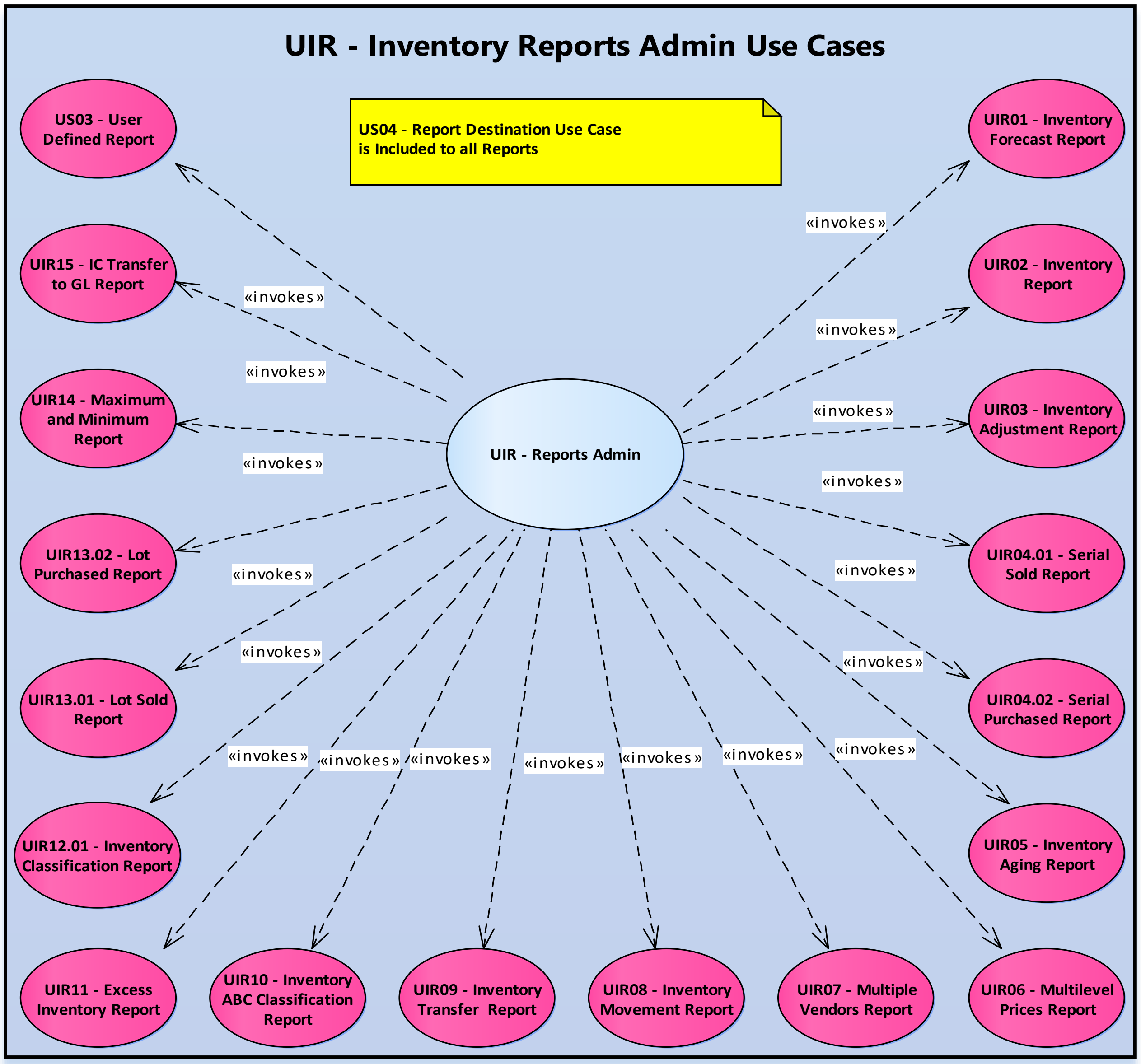

UIR - Inventory Reports Admin Use Cases
US04 - Report Destination Use Case is Included to all Reports
UIR - Reports Admin
«invokes»
US03 - User Defined Report
UIR15 - IC Transfer to GL Report
UIR14 - Maximum and Minimum Report
UIR13.02 - Lot Purchased Report
UIR13.01 - Lot Sold Report
UIR12.01 - Inventory Classification Report
UIR11 - Excess Inventory Report
UIR10 - Inventory ABC Classification Report
UIR09 - Inventory Transfer Report
UIR08 - Inventory Movement Report
UIR07 - Multiple Vendors Report
UIR06 - Multilevel Prices Report
UIR05 - Inventory Aging Report
UIR04.02 - Serial Purchased Report
UIR04.01 - Serial Sold Report
UIR03 - Inventory Adjustment Report
UIR02 - Inventory Report
UIR01 - Inventory Forecast Report

Figure 3: UIR Reports Admin

In this diagram you will see eighteen (18) use cases from the UIR – Reports Admin diagram (which is a continuation from the prior diagram UI – Inventory module Use Cases), the reports that comes out of these 18 uses case are really 288 reports.

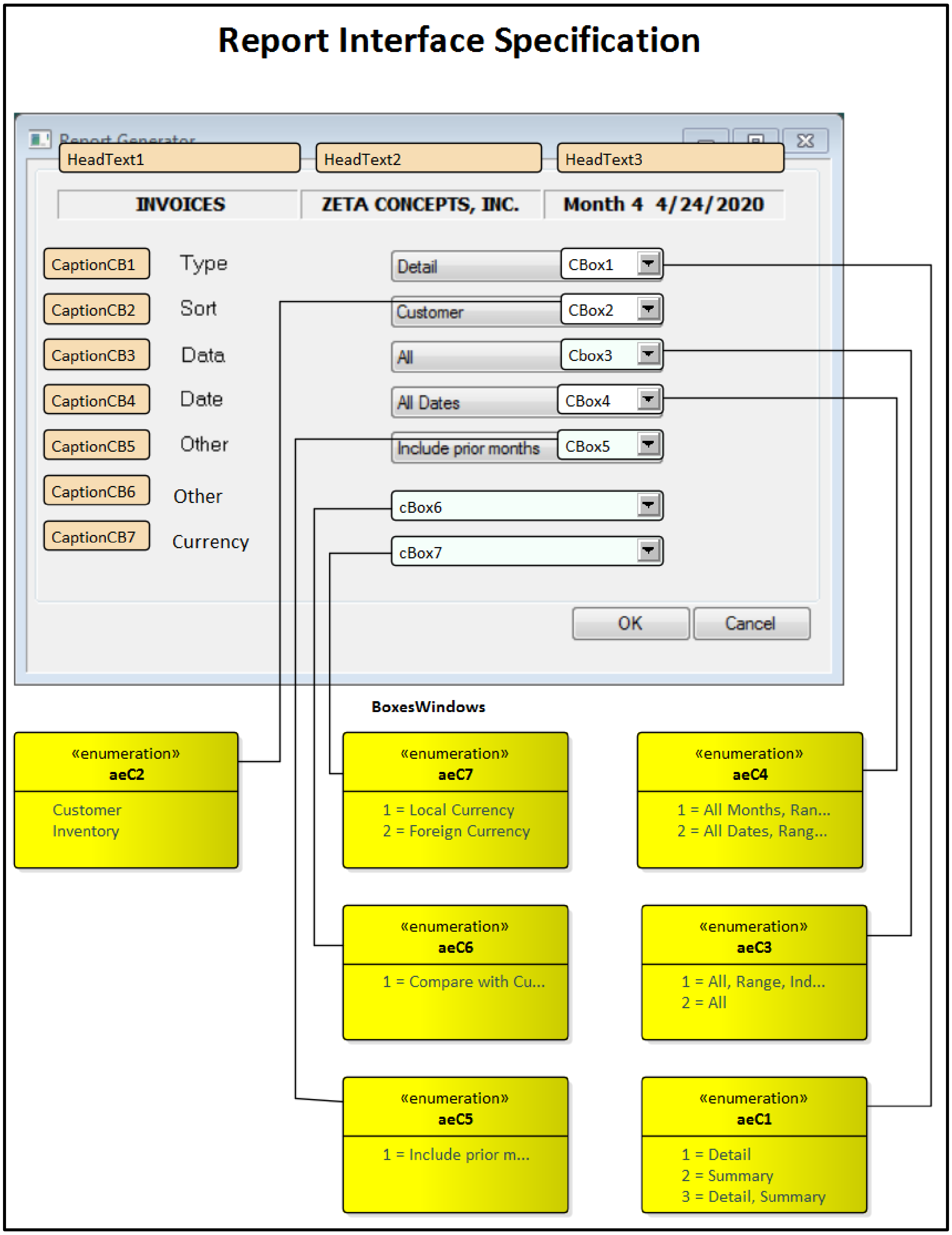
I am not going to go into a lot of details, because there is an eBook specifically for the reports at the Inventory module:

- UML-ERP Workshop, Writing a Business Requirement Document (BRD) for the Inventory module

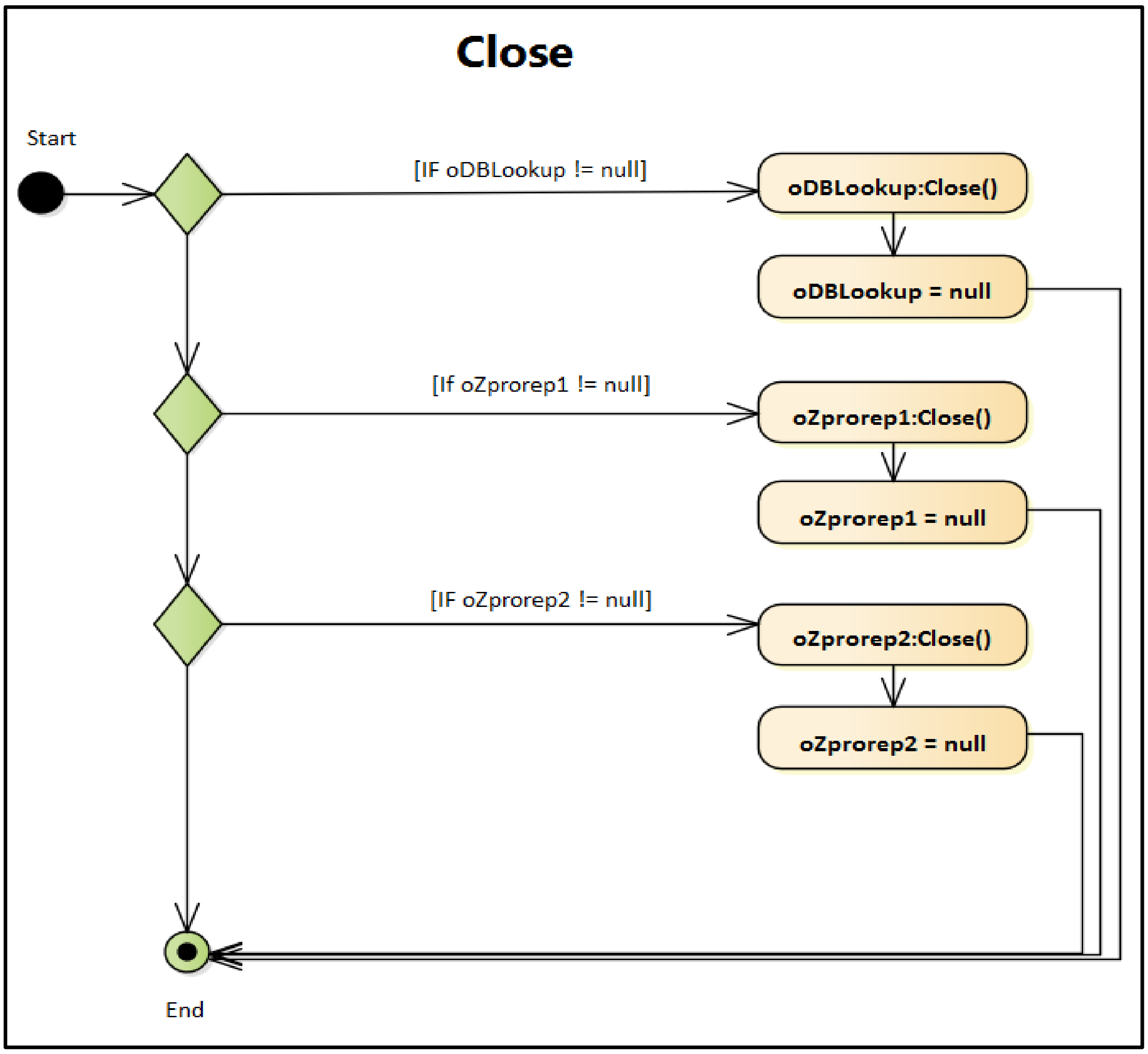
6 Data Driven Report Engine – SDD

Topics for this Chapter

Sample of the Interface Design Specification



Sample of an Activity Diagram of the Close operation for the BoxesWindow class



UML - ERP Workshop

Writing an SDD with Enterprise Architect

Anyone trying to develop a system within their organization has been faced with many competing demands. Demands come from those responsible for establishing the guidance for the organization, the industry standards that are designed to assist in establishing direction, the features provided by the available tool sets in this space, and those who are impacted by the direction as they attempt to do their daily work.

This is a complete, practical, hands-on book which is comprised of step-by-step instructions for creating an Enterprise Resource Planning System, covering everything you need to know about business processes, business rules, system processing and visualizing data at the Enterprise Level.