	LRA Authoring Guide: Knowledge Artefacts			
	Programme	NPFIT	Document Record ID Key	
	Sub-Prog / Project	Logical Record Architecture	NPFIT-SHR-MODL-GUID-0036.03	
	Prog. Director	Ken Lunn	Status	Approved
	Owner	Ross Dixon	Version	1.0
	Author	Bob Cox	Version Date	19/03/2010

Logical Record Architecture Authoring Guide: Knowledge Artefacts

Version 1.0

Approved

Amendment History:

Version	Date	Amendment History
0.00	04/03/2010	First draft for comment
0.1	18/03/2010	Amended following internal review comments
1.0	19/03/2010	Approved

Forecast Changes:

Anticipated Change	When
Changes to reflect lessons learned from Business as Usual adoption	Monthly Review

Reviewers:

This document must be reviewed by the following :< author to indicate reviewers>

Name	Signature	Title / Responsibility	Date	Version
Steve Bentley		LRA Project Manager		
Kevin John		Service Analyst		
Peter Otter		LRA Business Analyst	12/03/2010	1.0
Denis Hutchison		LRA Business Analyst	12/03/2010	1.0

Approvals:

This document must be approved by the following: <author to indicate approvers>

Name	Signature	Title / Responsibility	Date	Version
Ken Lunn		Director of Data Standards & Products	19/03/2010	1.0
Ross Dixon		Head of Analysis	19/03/2010	1.0

Distribution:

All Model Owners, Custodians, Authors and Reviewers

Document Status:

This is a controlled document.

Whilst this document may be printed, the electronic version maintained in FileCM is the controlled copy. Any printed copies of the document are not controlled.

Related Documents:

These documents will provide additional information.

Ref no	Doc Reference Number	Title	Version
1	NPFIT-SHR-QMS-PRP-0015	Glossary of Terms Consolidated.doc	<enter latest>
2	NPFIT-SHR-MODL-GUID-0014	Analysis Authoring Guide	latest
3	http://www.omg.org/spec/UML/2.1.1/	UML Version 2.1.1 Specification	2.1.1

4	http://www.ietf.org/rfc/rfc2119.txt	IETF Requirement Level Indicators RFC2119	March 1997
5	http://www.wrsystems.com/whitepapers/managedes2k.pdf	MoSCoW Prioritisation Classification	n/a
6	NPFIT-FNT-TO-DPM-0911	LRA Artefacts Overview V2.01	latest
7	http://www.ihtsdo.org/fileadmin/user_upload/Docs_01/SNOMED_CT_Publications/SNOMED_CT_User_Guide_20080731.pdf	SNOMED CT User Guide	latest
8	http://www.ihtsdo.org/fileadmin/user_upload/Docs_01/Technical_Docs/abstract_models_and_representational_forms.pdf	SNOMED CT Abstract Models and Representational Forms	Latest
9	NPFIT-FNT-TO-DPM-0936	NHS Logical Record Architecture Data Types Abstract Specification	Latest

Templates for use in compliance with Analysis Authoring Guide:

Ref no	Doc Reference Number	Title	Version
1	To be completed as part of the LRA Transition work package. Expected completion June 2010	Analysis Scope and Objectives Definition Template	latest
2	To be completed as part of the LRA Transition work package. Expected completion June 2010	LRA Artefact Checklist Template (.xls)	latest
3	To be completed as part of the LRA Transition work package. Expected completion June 2010	Change History Template	latest
4	To be completed as part of the LRA Transition work package. Expected completion June 2010	NHS Connecting for Health Comments sheet	latest
5	To be completed as part of the LRA Transition work package. Expected completion June 2010	Analysis Artefact Project Template (.mpt)	
6	To be completed as part of the LRA Transition work package. Expected completion June 2010	Introduction to LRA Analysis Template	latest
7	To be completed as part of the LRA Transition work package. Expected completion June 2010	LRA Review Checklist Template	latest

Glossary of Terms:

List any new terms created in this document. Mail the NPO Quality Manager to have these included in the master glossary above [1].

Term	Acronym	Definition

Contents

1	ABOUT THIS DOCUMENT.....	10
1.1	PURPOSE.....	10
1.2	AUDIENCE	10
1.3	BACKGROUND.....	10
2	APPROACH SUMMARY	10
2.1	INTRODUCTION	10
2.2	UNIFIED MODELLING LANGUAGE	11
2.3	RELATIONSHIP WITH ISO 13606 AND SNOMEDCT.....	11
3	CONTROL	11
3.1	ROLES	11
3.2	DISTRIBUTED AUTHORING	12
3.3	MODEL LIBRARY	13
3.4	ANALYSIS COLLABORATION – SHAREPOINT	14
3.5	ELEMENT AND CONSTRUCT CONSISTENCY.....	14
3.6	VERSION CONTROL	14
3.6.1	<i>Version Control with SVN</i>	<i>15</i>
3.6.2	<i>Version Control without SVN</i>	<i>15</i>
3.7	POST ANALYSIS ARTEFACT RE-USE	15
3.7.1	<i>Control of Post-Analysis Artefact Development</i>	<i>15</i>
4	ROUTE MAP	16
4.1	INTRODUCTION	16
4.2	DOMAIN ANALYSIS & SCOPING	16
4.3	LOGICAL ANALYSIS	19
4.4	LRA ANALYSIS	19
4.5	ROUTE MAP REALISATION	21
5	GOVERNANCE	23
5.1	INTRODUCTION	23
5.1.1	<i>Categories of Deliverable</i>	<i>23</i>
5.1.2	<i>Categories of Project Phase.....</i>	<i>23</i>
5.1.3	<i>Categories of Review/Approval.....</i>	<i>23</i>
5.2	GOVERNANCE ADOPTION	24
5.3	GOVERNANCE LOGISTICS	27
5.3.1	<i>Review Checklist</i>	<i>27</i>
5.3.2	<i>EA Considerations.....</i>	<i>27</i>
6	PUBLICATION OF MODELS.....	30
6.1	INTRODUCTION	30
6.2	NAVIGATION PAGE – MODEL DEFAULT	31
6.2.1	<i>Introduction.....</i>	<i>31</i>
6.2.2	<i>General Use</i>	<i>31</i>
6.2.3	<i>EA Use</i>	<i>31</i>
6.3	FILECM	32
6.3.1	<i>Introduction.....</i>	<i>32</i>
6.3.2	<i>Requirements</i>	<i>32</i>
6.3.3	<i>Guidance</i>	<i>33</i>
7	ANALYSIS ARTEFACT CHECKLIST	34
7.1	INTRODUCTION	34
7.2	GENERAL CONSTRUCTION.....	34

8	ANALYSIS MODELS	35
8.1	INTRODUCTION	35
8.2	GENERAL CONSTRUCTION.....	35
8.2.1	<i>Element Identification</i>	<i>35</i>
8.2.2	<i>Project View Naming.....</i>	<i>35</i>
8.2.3	<i>Business Rules</i>	<i>35</i>
8.2.4	<i>Business Use Case.....</i>	<i>35</i>
8.3	ANALYSIS ARTEFACT STATUS	36
8.4	MODEL ANNOTATIONS AND ADDENDA	36
8.4.1	<i>Model Annotations.....</i>	<i>36</i>
8.4.2	<i>Constraints</i>	<i>41</i>
8.4.3	<i>Gap Analyses</i>	<i>44</i>
8.4.4	<i>Test Conditions</i>	<i>45</i>
8.4.5	<i>Embedded Documents & External Artefacts Usage within Models.....</i>	<i>45</i>
8.5	MODEL RISKS, ISSUES AND CHANGES.....	46
8.5.1	<i>Risks.....</i>	<i>47</i>
8.5.2	<i>Issues</i>	<i>47</i>
8.5.3	<i>Changes.....</i>	<i>47</i>
8.6	REFERENCING ANALYSIS MODEL ELEMENTS	48
8.6.1	<i>General Construction.....</i>	<i>48</i>
8.6.2	<i>EA Construction</i>	<i>49</i>
8.7	REFERENCING EXTERNALLY MANAGED ARTEFACTS	49
9	REQUIREMENTS	50
9.1	INTRODUCTION	50
9.2	REQUIREMENTS DEFINITIONS.....	50
9.2.1	<i>Requirement.....</i>	<i>50</i>
9.2.2	<i>Package</i>	<i>51</i>
9.2.3	<i>Requirements Diagram</i>	<i>51</i>
9.3	REQUIREMENTS NOTATION.....	51
9.3.1	<i>Core Notation</i>	<i>51</i>
9.3.2	<i>Toolset</i>	<i>53</i>
9.4	NOTATION ADOPTION	54
9.5	REQUIREMENTS CATALOGUE.....	56
9.5.1	<i>General Construction.....</i>	<i>56</i>
9.5.2	<i>EA Construction</i>	<i>58</i>
9.5.3	<i>Requirements Catalogue Quality Review</i>	<i>59</i>
10	USE CASES.....	60
10.1	INTRODUCTION.....	60
10.2	USE CASE DEFINITIONS	60
10.2.1	<i>Use Case</i>	<i>60</i>
10.2.2	<i>Actor</i>	<i>60</i>
10.2.3	<i>Actors Catalogue</i>	<i>60</i>
10.2.4	<i>Actor Hierarchy.....</i>	<i>60</i>
10.2.5	<i>Use Case Catalogue</i>	<i>60</i>
10.3	USE CASE NOTATION	60
10.3.1	<i>Core Notation</i>	<i>60</i>
10.3.2	<i>Toolset</i>	<i>65</i>
10.3.3	<i>Notation Adoption.....</i>	<i>65</i>
10.4	USE CASE TYPES	66
10.4.1	<i>Use Case Catalogue</i>	<i>66</i>
10.4.2	<i>Use Case Model</i>	<i>66</i>
10.4.3	<i>Use Case</i>	<i>67</i>
10.4.4	<i>Business Use Case.....</i>	<i>69</i>

11	CLASS MODELS.....	70
11.1	INTRODUCTION.....	70
11.2	CLASS MODEL DEFINITIONS	70
11.2.1	<i>Class</i>	70
11.2.2	<i>Association</i>	70
11.2.3	<i>Attribute</i>	71
11.2.4	<i>Operations</i>	71
11.2.5	<i>Constraints / Dependencies</i>	72
11.3	CLASS MODEL NOTATION.....	72
11.3.1	<i>Core Notation</i>	72
11.3.2	<i>Toolset</i>	79
11.3.3	<i>Notation Adoption</i>	80
11.4	CLASS MODEL TYPES	84
11.4.1	<i>Stakeholder Information View (SIV)</i>	84
11.4.2	<i>Information Analysis Model (IAM)</i>	85
11.4.3	<i>LRA Analysis Class Models</i>	86
11.4.4	<i>Domain Information Description</i>	87
11.4.5	<i>Candidate Data Element</i>	90
11.5	CLASS MODEL TYPE SUMMARY.....	91
11.6	CLASS MODEL EXAMPLES	93
11.7	CLASS MODEL QUALITY REVIEW	93
11.8	CLASS MODEL ISSUES	93
11.8.1	<i>Traceability</i>	93
12	ACTIVITY DIAGRAMS.....	94
12.1	INTRODUCTION.....	94
12.2	ACTIVITY DIAGRAM DEFINITIONS AND NOTATION.....	94
12.2.1	<i>Core Notation</i>	94
12.2.2	<i>Toolset</i>	104
12.2.3	<i>Notation Adoption</i>	105
12.3	ACTIVITY DIAGRAM TYPES	107
12.3.1	<i>Stakeholder Business Model</i>	108
12.3.2	<i>Business Process Model</i>	109
12.3.3	<i>Outline Query Diagram</i>	115
12.3.4	<i>Domain Query Functionality Activity Diagram</i>	116
12.3.5	<i>Documentation</i>	118
12.3.6	<i>Annotation</i>	119
12.4	DOMAIN QUERY FUNCTIONALITY ACTIVITY DIAGRAM EXAMPLE	119
12.5	ACTIVITY DIAGRAM EXAMPLES	119
12.6	ACTIVITY DIAGRAMS QUALITY REVIEW	119
12.7	ACTIVITY DIAGRAM ISSUES.....	120
12.7.1	<i>Alias</i>	120
13	APPENDIX 1: CASE TOOL USE	121
13.1	ENTERPRISE ARCHITECT	121
13.2	EA OPTIONS CONFIGURATION	121
13.2.1	<i>Increase Minimum Size of Diagram Images</i>	121
13.3	DOCUMENTING OR ANNOTATING MODELS/PROJECTS VIA EA.....	123
13.3.1	<i>EA Model Registers</i>	123
13.3.2	<i>Notes on or Attached to Elements</i>	123
13.3.3	<i>Publishing Control Information</i>	123
13.3.4	<i>Publishing to HTML</i>	123
13.4	EA COMMON ACTION INSTRUCTIONS	124
13.4.1	<i>Auto-Numbering of Elements</i>	124
13.4.2	<i>Drawing Right Angle Links (Associations, Flows etc)</i>	124

13.4.3	<i>Deleting Elements</i>	124
13.4.4	<i>Set as the model default</i>	125
13.4.5	<i>Create a package</i>	125
13.4.6	<i>Display/Hide Element Features</i>	125
13.4.7	<i>Drag and drop the elements into the correct package</i>	125
13.4.8	<i>Create a new diagram</i>	125
13.4.9	<i>Generate HTML set</i>	125
13.4.10	<i>Opening HTML Set</i>	126
13.4.11	<i>Generate RTF output</i>	126
13.4.12	<i>Import XML</i>	127
13.4.13	<i>Export to XML</i>	127
13.4.14	<i>Using the Search Engine within an EA Model</i>	127
13.4.15	<i>Caption Bar</i>	131
13.4.16	<i>Format Hyperlink to Diagram</i>	131
13.4.17	<i>Printing and Exporting for use in MS Word</i>	131
13.4.18	<i>Copying Elements</i>	132
13.4.19	<i>Creating Links Between Elements that are spatially separate</i>	132
13.4.20	<i>Moving and Resizing Elements</i>	132
13.4.21	<i>Moving Between Open Diagrams</i>	132
13.4.22	<i>Model Authors</i>	132
13.4.23	<i>F2 Hot-key (Edit element text on diagram)</i>	132
13.4.24	<i>Watermark on diagrams</i>	133
13.4.25	<i>Change Connector source/ target</i>	133
13.4.26	<i>Finding Diagrams in which an Element Exists</i>	133
13.4.27	<i>Understanding & Using the Relationship Matrix</i>	133
13.4.28	<i>Tracing Related Elements</i>	137
13.4.29	<i>Indicating Nominal Relationships</i>	138
13.4.30	<i>Linking Documents to the EA Model (native and HTML output)</i>	139
13.4.31	<i>Creating/Exporting a (Requirements) CSV File</i>	140
13.4.32	<i>Creating Relative Paths to External Documents</i>	141
13.4.33	<i>Creating Hyperlinks to External Documents</i>	143
13.4.34	<i>Creating an Association Class</i>	144
13.4.35	<i>Diff Facility</i>	144
13.4.36	<i>Compare Versions of a Package</i>	146
13.4.37	<i>Creating Stereotypes</i>	148
13.4.38	<i>Bookmarking Elements</i>	149
13.4.39	<i>Searching for Bookmarked Elements</i>	149
13.4.40	<i>Adding and Updating Glossary Entries</i>	150
13.4.41	<i>Setting 'Rectangle Notation'</i>	150
13.4.42	<i>Adding Tagged Values</i>	150
13.4.43	<i>Adding Risks</i>	151
13.4.44	<i>Adding Issues</i>	152
13.4.45	<i>Adding Changes</i>	153
14	APPENDIX 2: SUBVERSION AND EA	154
14.1	INITIAL SET UP OF SUBVERSION (SVN) AND EA	154
14.2	USING EA IN AN SVN ENVIRONMENT	156
14.2.1	<i>Updating you EA Model to the latest SVN copy</i>	156
14.2.2	<i>Checking out</i>	157
14.2.3	<i>Checking in</i>	157
14.2.4	<i>Multiple check in</i>	157
14.2.5	<i>Undo checkout</i>	158
14.2.6	<i>Version control</i>	158
14.2.7	<i>Checking out mismatches</i>	159

15	APPENDIX 3: METHODOLOGY FOR THE IDENTIFICATION AND MAINTENANCE OF RENAL CANDIDATE DATA ELEMENTS	161
15.1	BACKGROUND	161
15.2	INTRODUCTION.....	161
15.3	ANALYSIS OF DATA ITEMS IN DID	161
15.4	CREATION OF CANDIDATE DATA DEFINITIONS	163
15.5	MAINTENANCE OF THE DID AND DATA ELEMENTS DEFINITION PALETTE	165
15.5.1	<i>Candidate Data Elements</i>	165
15.5.2	<i>DID</i>	168
16	APPENDIX 4: SQL MODEL SEARCH SCRIPT.	170
16.1	INTRODUCTION.....	170
16.2	QUERY LOGIC.....	170
16.3	SQL SCRIPT	171
17	APPENDIX 5: PRODUCT DESCRIPTIONS	175
17.1	STAKEHOLDER INFORMATION VIEW	175
17.2	STAKEHOLDER BUSINESS MODEL.....	177
17.3	REQUIREMENTS CATALOGUE	179
17.4	BUSINESS PROCESS MODEL.....	183
17.5	USE CASE	185
17.6	USE CASE MODEL	187
17.7	USE CASE CATALOGUE	189
17.8	INFORMATION ANALYSIS MODEL	190
17.9	OUTLINE QUERY ACTIVITY DIAGRAM	192
17.10	DOMAIN INFORMATION DESCRIPTION.....	194
17.11	CANDIDATE DATA ELEMENT DESCRIPTION	196
17.12	DOMAIN QUERY FUNCTIONALITY ACTIVITY DIAGRAM	198
18	APPENDIX 6: ARTEFACT EXAMPLES	201
18.1	STAKEHOLDER INFORMATION VIEW	202
18.2	STAKEHOLDER BUSINESS MODEL.....	203
18.3	REQUIREMENTS CATALOGUE	204
18.4	INFORMATION ANALYSIS MODEL	205
18.5	BUSINESS PROCESS MODEL.....	206
18.6	USE CASE MODEL	207
18.7	OUTLINE QUERY DIAGRAM	208
18.8	DOMAIN INFORMATION DESCRIPTIONS	209
18.9	CANDIDATE DATE ELEMENTS.....	210
18.10	DOMAIN FUNCTIONALITY ACTIVITY DIAGRAM.....	211

1 About this Document

1.1 Purpose

The document aims to provide guidance on how to produce the analysis artefacts required during the lifecycle of a Logical Record Architecture (LRA) analysis project. It acts as a reference guide to the production of the LRA analysis artefacts giving detailed instructions on how to create and maintain each artefact. This document can be used with any toolset for the production of artefacts; however it also provides advice on the use of the CASE tool which is in widest use within NHS Connecting for Health. This is currently Enterprise Architect (EA) from Sparx Systems (see §13).

In addition to the description of individual artefacts, this document provides context to the production of these artefacts by describing their place in the overall lifecycle. Thus a more complete understanding of the underlying purpose and use of the analysis artefacts will be described here. This document also provides guidance on the creation and use of corporate artefacts i.e. analysis artefacts that may be re-useable across different domains which will be stored centrally and made available to any domain that needs to use them

1.2 Audience

This document is intended for Analysts and those needing to develop analysis models depicting requirements, functions, queries, processes and information. It provides specific instructions and guidance for analysis activities and their outputs; it is adopted as the Logical Record Architecture (LRA) analysis methodology of the Technology Office of NHS Connecting for Health and has been developed to facilitate adoption of LRA. Although there are many commonalities between the existing Analysis Authoring Guide (see Related Document 2) and this guide, it should be noted that this guide does not support communication and does not replace the existing Analysis Authoring Guide. Comprehensive supporting information is included to assist someone new to this methodology and UML as well as supporting the use of the currently approved CASE tool (see §13.1); however it does not purport to be an inclusive reference tool for teaching all aspects of UML.

1.3 Background

The LRA Analysis Authoring Guide has been developed to support the phase 1 deployment of the LRA approach. Phase 1 focuses on the adoption of Knowledge Artefacts. These provide a framework to capture and refine information requirements in a consistent way. It is expected that this guide will change as and when other elements of LRA are adopted in later phases

2 Approach Summary

2.1 Introduction

This document is designed to give the reader an understanding of the artefacts an Analyst will produce during the project development lifecycle and is applicable to

defining information requirements. It details the artefacts that are produced and gives guidance on when they are produced and the relationship between the different artefacts.

Readers should be able to use this guide as a reference manual – able to easily access information about a particular artefact as required. The routemap section (§4) gives an overview of the underlying process resulting in the production of the required artefact set; but more detailed information about how the artefacts interact and develop is found within the sections on the artefacts themselves

2.2 Unified Modelling Language

The modelling approach implemented by this methodology utilises the Unified Modelling Language (UML) notation. Specifically, UML version 2.1.1 has been (partially) implemented by this document (see Related Document 3). The principal UML artefacts produced are:

- Use Cases (see §10)
- Activity Diagrams (see §12)
- Class Models (see §11)

More detailed explanations of these UML artefacts and how they are produced and used within NHS Connecting for Health are documented in the sections below.

2.3 Relationship with ISO 13606 and SnomedCT

This methodology is designed with the intent to interface with Technical Modelling Artefacts. It is anticipated that later phases of LRA deployment will allow a direct transformation from Knowledge Artefacts to the relevant Technical Artefacts which will in turn handle the necessary relationships between ISO 13606 and SnomedCT. However for phase 1 this will remain a manual process conducted by the appropriate technical modelling teams.

In this respect the key artefacts of interest to the Technical Modelling teams will be the Domain Information Description (see §11.4.4) and the Candidate Data Element (see §11.4.5).

3 Control

This document has 2 levels of approval: Major increments are approved by the Director Responsible for Analysis within the NHS Connecting for Health Technology Office (Ken Lunn) and the NHS Connecting for Health Technology Office Head of Analysis (Ross Dixon); Minor increments are approved by the NHS Connecting for Health Technology Office Head of Analysis (Ross Dixon).

Models produced using the approach articulated within this document will be subject to a variety of controls as follows:

3.1 Roles

The generation and management of individual models will involve a number of roles:

- *Owner*: generally the Project Manager for the Domain under analysis; responsible for the validity of the model content and for setting review and approval requirements for artefacts and ensuring that these requirements are met. Has publication release control for the model (i.e. its release outside of NHS Connecting for Health). Has authority to accept/reject the comments of reviewers. Unlike other roles, the Owner is a valid role after the model is complete – the Owner is thus responsible for maintenance of the model after its completion.
- *Custodian*: allocated by or in conjunction with the Analysis Team; responsible for the production of an individual model including the allocation of authors (see § 3.2) and management of their involvement with model creation including:
 - Detailed scoping of the Analysis Work package.
 - conformance of models to the notation, rules and conventions described in this document
 - incorporation of author contribution into the models
 - Entry of model into the Model Library (see §3.3).
- *Author*: allocated by the Custodian; responsible for the parts of the model identified by the Custodian.
- *Reviewer*: allocated by the Owner in consultation with the Custodian; responsible for the review of models and completion return of appropriate comments (using provided template 4). It is expected that Reviewers will include domain, technical and model methodology experts.
- *Approver*: grants approval for artefacts to progress to their next stage, this can include the development of follow on artefacts or hand-over to artefact users.

Material and facilities which are not associated with individual models (such as this document and the Model Library) are managed by the Analysis Team

3.2 Distributed Authoring

It is expected that most models will be produced by multiple Authors and that these authors will belong to different parts of NHS Connecting for Health. To control this distributed approach to authoring, the activities undertaken by the Custodian and Authors needs to be clearly defined. The Custodian will divide the model and allocate parts to Authors.

Where the selected tool (see §13.1) is used, the partition of the model will make use of the 'package' facility such that each partition is exported/imported as a package to the tool by the Author who will modify/enrich the partition before returning to the Custodian as a package. The Custodian adds returned packages to the model (ensuring model consistency/integrity) leading to a new version of the model and its entry into the Model Library.

Where the Subversion (SVN) version control tool is used to manage distributed authoring, the version control aspect of the role of Custodian is largely replaced by SVN's functionality (see §3.6.1)

To prevent conflicts an Author must only amend the diagram/view/package for which he/she is responsible at that time; any comments/amendments on other aspects must be communicated to the relevant Author or Custodian.

3.3 Model Library

The main NHS Connecting for Health repository (FileCM see §6.3) which will be used as the Model Library, is available at <http://filecm-lds/filecm/IEmain.asp> and is managed by the Analysis Team. Models are to be registered within this library within their domain folders by the model Custodian according to the library structure within FileCM as shown in Figure 1.

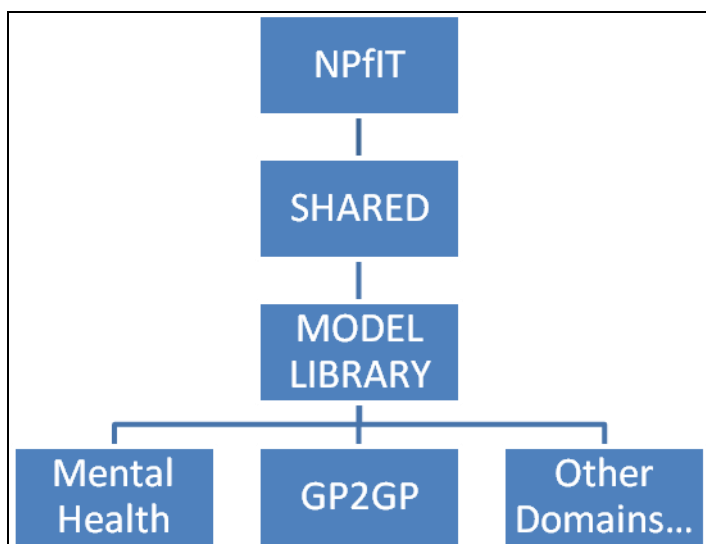


Figure 1: model library structure

The following principles of operation will be applied:

- Where artefacts are held within a different part of FileCM (e.g. Choose and Book Programme area) no content is expected within the Model Library, but entries should exist within the model Library which point to the appropriate location within FileCM.
- Where artefacts held within FileCM are also incorporated into a model (see §6.1), the version held in FileCM is considered to be the master and is subject to version control independently of the model.
- Within each domain folder individual models, and associated material, will be distinguished by index metadata (e.g. Document, Model etc).
- Templates for the generation of analysis artefacts are contained within the Templates folder.
- Non-domain specific artefacts for re-use within the domains will be held within the Components folder (containing components and fragments)
- Rules and guidance for publication of analysis models to the FileCM Model Library are given at §6.3

3.4 Analysis Collaboration – SharePoint

Contribution to methodology development and tooling is encouraged and SharePoint infrastructure has been made available specifically for this purpose. The main SharePoint site the 'Work in Progress' site for future versions located at (<http://shareit/sites/busarch/busreq/meth/default.aspx>)

NB. This is restricted to team members specifically tasked with methodology development

Several discussion sites are available on the Business Architecture SharePoint infrastructure, (<http://shareit/sites/busarch/default.aspx>) these are:

- Public (Available to all NHS Connecting For Health):
- Business Architecture: EA (CASE Tool) General Discussion
- Business Architecture: EA (CASE Tool) Hints & Tips
- Business Architecture: General Discussion Forum
- Private (Available to Analysis Team only):
- [General Discussion](#)
- [Methodology Discussion](#)
- [Suggest: Authoring Guide Changes](#)
- [EA \(CASE Tool\) Discussion](#)

3.5 Element and Construct Consistency

The methodology defined within this document requires that a variety of artefacts are produced and many of these are individually iterated or are developments from a predecessor artefact. Although this is a natural result of the requirement to analyse broad and complex areas, and is also consistent with analysis best practice, this leads to potential problems of inconsistency within the artefact set.

Individual product descriptions (see §17) identify artefact-specific requirements and constraints to ensure consistency; the route map (see §4) describes how the artefacts are related and how artefacts are used in the creation of other artefacts. However, the general principle is that the Analyst must ensure that consistency is maintained throughout the artefact set. Thus any artefacts constructed in the earlier phases of analysis must still be valid when the later analysis models are completed. This will involve working iteratively, revisiting the earlier models as the later models are produced to ensure the scope is still being addressed. Any changes must be fed back into the earlier models and these should be reworked if necessary. All the analysis models should support the scope originally identified and demonstrate how they are meeting this scope. If the analysis artefacts cause the scope to change this must be validated with the appropriate body and the scope updated.

3.6 Version Control

The Model Library (see §3.3) is built as an application of the FileCM product which provides the overarching version control and configuration management mechanism for artefacts. FileCM is used to allocate the NHS Connecting for Health identifier,

which is based upon the file structure created within FileCM. Each artefact will use a file structure as follows: NPFIT -- Shared -- ModelLibrary -- Domain -- DocNumber – SerialNumber (e.g. NPFIT-SHR-MODL-GUID-0014.05 – this documents reference).

All artefacts which are distributed for external review must be registered upon FileCM prior to distribution and must carry the resulting NHS Connecting for Health ID.

Mandatory metadata fields associated with domain artefacts include the EA version.

3.6.1 Version Control with SVN

The LRA project used a SVN repository to manage version control. The long-term intention is that modelling artefacts will be managed on a shared environment allowing shared working, reuse and version control. However it is not envisaged that this will be implemented for Phase 1 deployment of the LRA process which this guide covers.

It is therefore assumed that the analyst will adopt manual processes to facilitate version control. Should this not be the case, full details of how to set up and use the CASE Tool with a predefined SVN repository are described in §14

3.6.2 Version Control without SVN

For ease of development of artefacts, including facilitating the use of multiple authors and internal reviews, local and simple version control may also be applied in addition to the FileCM mechanism.

Version control will be manually applied using a 2 part identification of the form n1.n2 which will be identified within the filename, the artefact itself and FileCM metadata but is not reflected within the ID number generated by FileCM, such that:

- n1: Main version number that indicates significant changes in the models status or contents. Note that for initial drafts of a model, n1 is set as 0 (i.e. 0.1 to 0.n). n1 is incremented to 1 on model approval.
- n2: Minor version number that indicates small changes in the models status or contents.
- These numbers are not part of the number generated by FileCM but are added to the filename, FileCM Metadata and content of the document/artefact.

In addition to recording the version history in the Model Library, model authors must maintain a model change history document (using provided template 3), in which specific changes made to the model are recorded for every version increment. The model change history must be managed with the model to which it refers.

The same version control principles will be applied to complete models and also to corporate artefacts.

3.7 Post Analysis Artefact Re-Use

3.7.1 Control of Post-Analysis Artefact Development

During the analysis of any one domain there is a close and iterative relationship between all of the analysis artefacts being produced. During the project, it is expected that artefacts will be modified only by members of the project team to prevent erroneous development. After analysis is completed the further development of artefacts is encouraged. Such reuse takes two forms, the further development of a

domain within the team and the downstream use of analysis artefacts outside of the methodology. In the latter case:

- Version control needs to be maintained for the overall artefact set.
- Where the re-used artefact becomes a standalone new product this is subject to its own version control.

Analysis Artefact	Type of Re-Use
Requirements Catalogue	Testing by Suppliers, NIC, Model Communities and Users
Use Case Model including constituent Use Cases	Testing by suppliers, NIC and Model Communities Implementation Planning
Business Process Model	Technical Design Documentation Testing by suppliers, NIC and Model Communities Implementation Planning Business Process Re-engineering
Domain Information Descriptions	Data Modellers Data Dictionary Terminologists

Table 1: post-analysis use of artefacts

4 Route Map

4.1 Introduction

The Analysis process consists of three broad phases:

- Domain Analysis and Scoping (see §4.2)
- Logical Analysis (see §4.3)
- LRA Analysis (see §4.4)

Each of these phases has a corresponding package of deliverables that are produced during that phase of work. The following sections deal with each of these phases and outline the activities undertaken and the deliverables produced; the expected order of production is depicted in Figure 2. How these deliverables are constructed can be found in later sections of this document.

4.2 Domain Analysis & Scoping

The output from the **Domain Analysis & Scoping Phase** of work is the Domain Analysis & Scoping Package which comprises:

- Requirements Catalogue

- Stakeholder Information View
- Stakeholder Business Model

The project will be started by the receipt of 'approval to start' (approval requirements are identified in §5.2). This instruction may be a very general statement and an important part of this **DOMAIN ANALYSIS & SCOPING PHASE** will be concerned with getting agreement on what is expected of the project by the various stakeholders.

Analysis modelling for the project begins with the Analyst identifying and analysing existing domain artefacts describing information and process concepts. New artefacts will be produced including (no order implied):

- A Stakeholders Business Model (SBM) articulates the Stakeholders' process view of the domain and may be a direct reuse of Stakeholder artefact(s) if these exist. This is used mainly for information gathering, verification and traceability purposes. The stakeholder may use their own concepts and terms at this point, and these will be mapped to more formal UML artefacts at later phases in the analysis.
- The Requirements Catalogue articulates the domain requirements expressed by Stakeholders and subsequently derived by Analysts. This artefact will be used as the basis for much of the subsequent analysis activities and outputs. In order to focus the range of requirements that could potentially populate the Requirements Catalogue, National Requirement References have been identified as the primary requirement source for developing the LRA.
- A Stakeholder Information View (SIV) articulates the main conceptual entities identified at this early stage in analysis and provides a conceptual frame of reference for the domain. It is likely to be the result of initial brain-storming or information gathering activities, but may also be a direct re-use of stakeholder artefacts if these exist. Alternatively, existing business information such as data sets, clinical document structures, or headings within clinical document sections relevant to that domain (discharge reports, assessments, Patient held records for example) may be used as an initial starting point.



4.3 Logical Analysis

The **Logical Analysis Phase** builds on the artefacts produced in the previous phase to produce more detailed information and process concepts. During this phase the analysis should refer back to the artefacts produced during that previous phase to ensure the further analysis and previous artefacts are consistent and traceability is preserved.

New artefacts will be produced including (no order implied):

- Information Analysis Model (IAM)
- Use Case Model (UCM: comprising of Use Cases and the Use Case Catalogue)
- Business Process Model (BPM)
- Requirements Catalogue
- Query Catalogue: (containing outline queries, activity diagrams)

Note that this process may lead to the revision of artefacts from the previous phase.

- The Information Analysis Model provides the first detailed definition of information requirements within the domain and is likely to include all information concepts appearing within the Use Case Model.
- The Use Case Model and the Business Process Model define the detailed behavioural parts of the artefact set. Both of these artefacts are composed from Use Case Activity Diagrams (i.e. these are reused between the UCM and BPM) and no ordering between these artefacts is mandated.
- The Requirements Catalogue, although initiated in the Domain Analysis and Scoping phase, undergoes further iterations of analysis to provide a more precisely defined requirements in the form of Derived Requirements and Informatic Concepts.
- The Query Catalogue articulates the initial definition of data queries which begins at this point in the analysis. They are likely to be simple titles with brief descriptions outlining their business purpose at this stage. In this form they will be stored within the Query Catalogue. A brief outline of their behaviour may also be added.

4.4 LRA Analysis

The **LRA Analysis Phase** of work represents the first significant consideration of LRA technical artefact requirements and is framed by the understanding gained in the previous analysis phase. The analysis carried out during Logical Analysis Phase determines if any technical artefacts (Candidate Data Elements) are required and provides guidance as to what these artefacts should be.

Note: For the complete LRA production environment where Knowledge artefacts co-exist with and are inputs to LRA Technical Artefacts, the LRA Analysis Phase artefacts are designed to interface with ISO 13606 ENTRY and ELEMENT definitions (For further details see Related Document 6) and technical query artefacts. However, for Phase 1 deployment of LRA the LRA Analysis artefacts may be subject to adaptation to meet current Data Modelling Team working practices.

The output from the LRA Analysis Phase produces the following Knowledge Artefacts:

- LRA Domain Information Descriptions
- LRA Domain Query Functionality
- Candidate Data Elements

As the need for Candidate Data Elements or queries are identified, pre-existing LRA technical artefacts should be evaluated to assess whether they meet the newly-identified business requirement. Should new technical artefacts be required, the LRA analysis phase is initiated. The result of this phase is a set of artefacts that draws on and are inputs to physical LRA technical artefacts.

- LRA Domain Information Descriptions are represented in the form of class models derived from the pre-existing Information Analysis Model. However, the model differs significantly in that it directly references existing Data Elements. Where there is an absence of a suitable corresponding Data Element a new Candidate Data Element should be created. If a number of LRA Domain Information Descriptions are to be defined and these are related, care should be taken to account for overlapping content. It may be preferable that this communication 'super-set' is analysed and modelled as a whole.
- LRA Domain Query Functionality is the only dynamic artefact produced at this stage. It is an activity/action diagram defining the interactions between query logic, the data elements and their operations/methods. This artefact provides a strong link between dynamic and static models since the interactions identified in the dynamic artefact set are linked to their corresponding static models.
- Candidate Data Elements represent the new information requirement for the LRA. They are derived either from the gap established between the Information Analysis Model and the LRA Domain Information Description or alternatively the LRA Domain Query Functionality will highlight the absence of suitable of Data Element class attributes or operations required to process its logic.

The relationships between the artefacts at the point of completed analysis are shown in Figure 3.

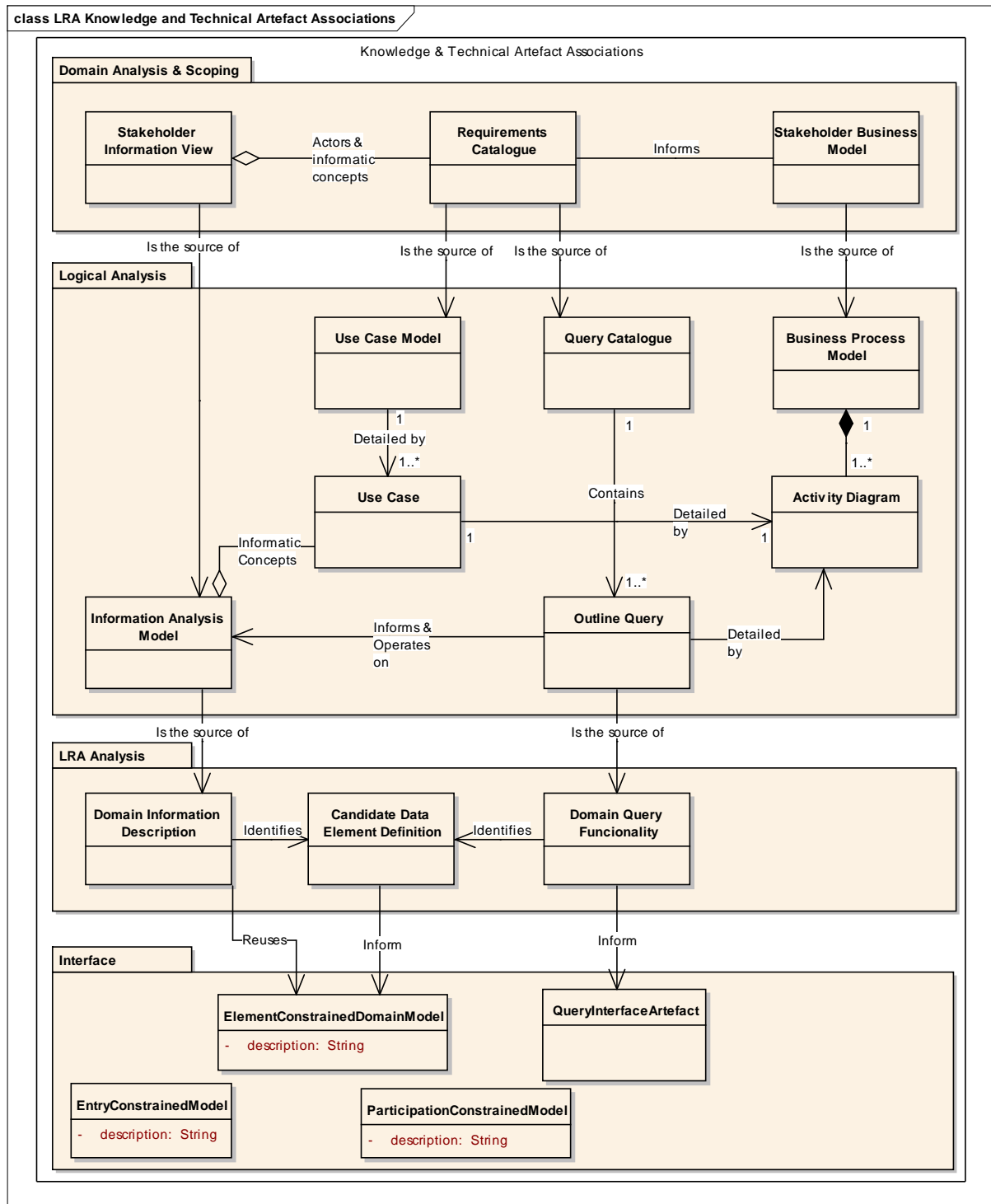


Figure 3 analysis product associations

4.5 Route Map Realisation

Each of the analysis artefacts can be considered to be started during one of the three phases described in §4. However, it will often be the case that the artefact is not completed during the same phase, as it will be revisited to ensure consistency with artefacts produced later.

A Gantt chart showing the expected order of production of artefacts is shown in **Figure 4**. A Microsoft Project template is available for use within analysis domains (see Template 5) which shows the relationships between artefacts. It is expected that this template will be tailored for each domain and other activities (such as review/approval, see §5.2) will be added, or may incorporated into a wider project plan.

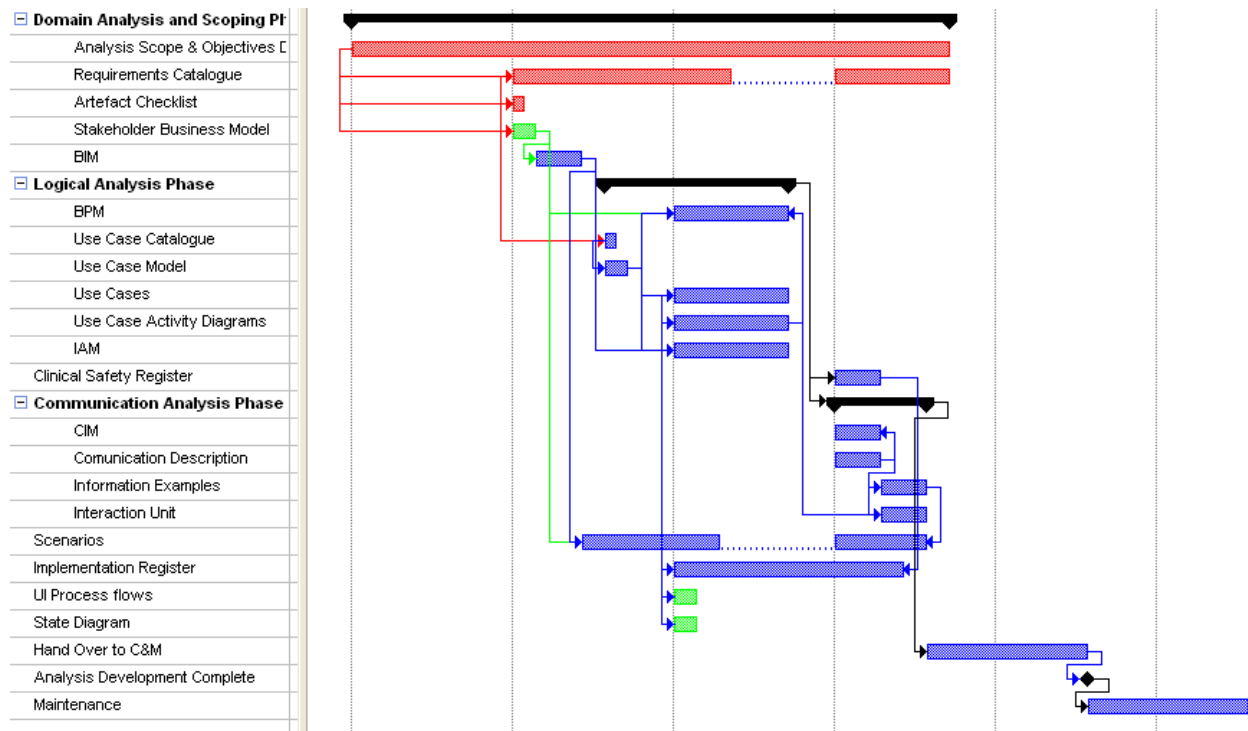


Figure 4 analysis artefact Gantt chart TO BE DONE

5 Governance

5.1 Introduction

The artefacts identified within this LRA Analysis Authoring Guide are used to inform and/or mandate subsequent development and implementation activities, and it is possible for some of these artefacts to be cited as compliance criteria. The importance of these artefacts requires that these are subject to the governance controls within this section.

To ensure rigorous governance without excessive overhead and delay requires that different levels of governance control are applied to different categories of artefact at different stages of the project lifecycle. These classifications are described below.

5.1.1 Categories of Deliverable

The artefacts are classified according to the following criteria:

- *transitional* (the artefact is a step towards another artefact) **or** terminating (the artefact is a final deliverable to be used by a consumer)?
- *reusable?* (i.e. is the artefact of value outside of the project)
- *direction setting?* (i.e. does the artefact guide/constrain subsequent artefacts)

The allocation of artefacts into these classifications is dependent on the nature of the project.

The general principles in respect of deliverable category are that:

- terminating artefacts require review/approval from recipients of the artefacts and those responsible for the artefacts
- reusable artefacts require review/approval from those responsible for reuse outside of the subject domain
- Direction setting artefacts require review/approval from accountable Senior Managers.

5.1.2 Categories of Project Phase

There are 3 groupings of artefacts which can be mapped to traditional project phases identified in §4, as follows:

- Domain Analysis and Scoping
- Logical Analysis
- LRA Analysis

5.1.3 Categories of Review/Approval

A number of different types of reviews and approvals are required through the lifecycle of analysis projects involving various participants, as follows:

- *Clinical/Domain Review*: Domain Clinical Lead, Domain Clinical Review Group
- *Project Review*: Project Team members
- *Peer Review*: fellow Analysts - not from the project team

- *Interim Assessment*: Project Manager, Lead Analyst, Analysis Team Manager and Head of Analysis
- *Approval to Start*: DS&P Senior Management Team, Programme Manager, Head of Analysis
- *Approval to Proceed*: Head of Analysis, Programme Manager, Analysis Team Manager
- Completion/Handover Approval

5.2 Governance Adoption

Following the definitions in §5.1 a governance matrix based upon categories of artefact progressing through project phases subject to review/approval controls is given in Table 2. This is used as the basis of the required reviews/approvals and gates underpinning the artefact governance approach (shown graphically in Figure 5). The order of review and approval activities (for individual artefacts) is shown in the left hand column.

As an example of using Table 2, the Stakeholder Information View is created during the **DOMAIN ANALYSIS AND SCOPING PHASE**, and must undergo Clinical/Domain Review, Project Review, Peer Review and Interim Assessment. When these reviews have been completed, the Stakeholder Information View can go forward to Approval to Proceed.

NB it is expected that artefacts will be batched together for approval submissions.

The formal Gates (shown in Figure 5) supporting this general model would occur at the following points:

- Gate 1:
 - when?: Start of the project
 - what?: Sanctioned Work Package
 - who?: 'Approval to Start'
- Gate 2:
 - when?: End of the Domain Analysis and Scoping Phase
 - what?: Requirements Catalogue, Stakeholder Business Model and Stakeholder Information View
 - who?: 'Approval to Proceed'
- Gate 3:
 - when?: End of the Logical Analysis Phase
 - what?: Logical Analysis artefact set and assurance of 'Clinical/Domain' / 'Project' / 'Peer' Reviews
 - who?: 'Interim Assessment'
- Gate 4:

- when?: Analysis project completion (NB this may coincide with Gate 3. In this case both Gates 3 & 4 are still required)
- what?: All unapproved artefacts and assurance of 'Clinical/Domain' / 'Project' / 'Peer' Reviews as well as Interim Assessment
- who?: 'Completion/Handover Approval'

	Domain Analysis and Scoping				Logical Analysis					Communication Analysis						
	Analysis Scope & Objectives Definition	Requirements Catalogue	Stakeholder Business Model	BIM	Use Case Model	Use Case Catalogue	Use Case Activity Diagrams	BPM	Information Analysis Model	Interaction Unit	CIM	Communic ation Description	Information Example	Scenario	Implementati on Register	Clinical Safety Register
Clinical / Domain Review																
Project Review																
Peer Review																
Interim Assessment																
Approval to Start																
Approval to Proceed																
Completion / Handover Approval																

Table 2 governance matrix **TO BE DONE**

5.3 Governance Logistics

To support the adoption identified in §5.2, the following low-level actions are to be enacted:

- All reviewers and approvers are identified within the Work package
- Any submission into a Gate **must** include evidence that the submitted artefact(s) has been previously reviewed in accordance with Table 2. The lack of this evidence at a Gate results in a rejection.
- Artefacts received at a Gate **must** have been checked for validity against their quality criteria (defined in their corresponding product descriptions in §17). This **must** take the form of a fully completed Review Checklist, see §5.3.1 .
- All reviews shall make use of the NHS Connecting for Health comments sheet (see Template 4) and a consolidated comments sheet for each review/approval round should be created, maintained and stored within the Model Library (see §3.3).
- Where artefacts to be submitted have been generated by the selected CASE tool (see §13.1), these must enact the instructions defined in §5.3.2

5.3.1 Review Checklist

To assist Analysts and Project Managers preparing artefacts for submission into Gates (see §5.2 & Figure 5), a set of checklists have been developed. Artefacts submitted into a Gate **must** be submitted with a completed checklist, as follows:

- the checklist document must be derived from provided template 7
- each artefact must have every entry (i.e. one for each criteria) completed
- a completed entry must have the 'Criteria Met?' column cell where a 'Criteria Met?' cell is completed with any value other than 'Fully', this must be explained within the corresponding 'Notes' cell

5.3.2 EA Considerations

5.3.2.1 Bookmarking

Where a submission for review into a Gate is a resubmission (for example to deal with comments from a previous submission, or a next version of a model) any elements which have been changed from the previous submission or are thought to be of particular interest **must** be bookmarked.

Guidance for the creation of bookmarks is given in §13.4.38

Guidance for searching for bookmarks within a model is given in §13.4.39

5.3.2.2 Element Linking

Significant models generally contain a large number of implicit relationships between contained elements. These relationships add to the richness of the models, aid users of the models and facilitate model maintenance. However, to be usable these relationships **must** be made explicit using the Relationship Matrix function see §13.4.27 (NB tracing element relationships using the relationship matrix is described in §13.4.28)

Models submitted into Gates **must** (as a minimum) contain the relationships identified in §8.6

5.3.2.3 Change History Linking

A Change History document is described in §3.6, this is mandatory for all submissions into Gates and **must** be linked to the submitted model from the Navigation Page (see §6.2) using the document incorporation technique identified in §13.4.30. NB. This document is required in addition to the recording of changes within the model (see §8.5.3).

5.3.2.4 General External Artefact Linking

A number of artefacts are required to be incorporated into the model (see §6.1) using the document incorporation technique identified in §13.4.30, where this occurs, the version of the incorporated artefact must be appropriate to the submitted model.

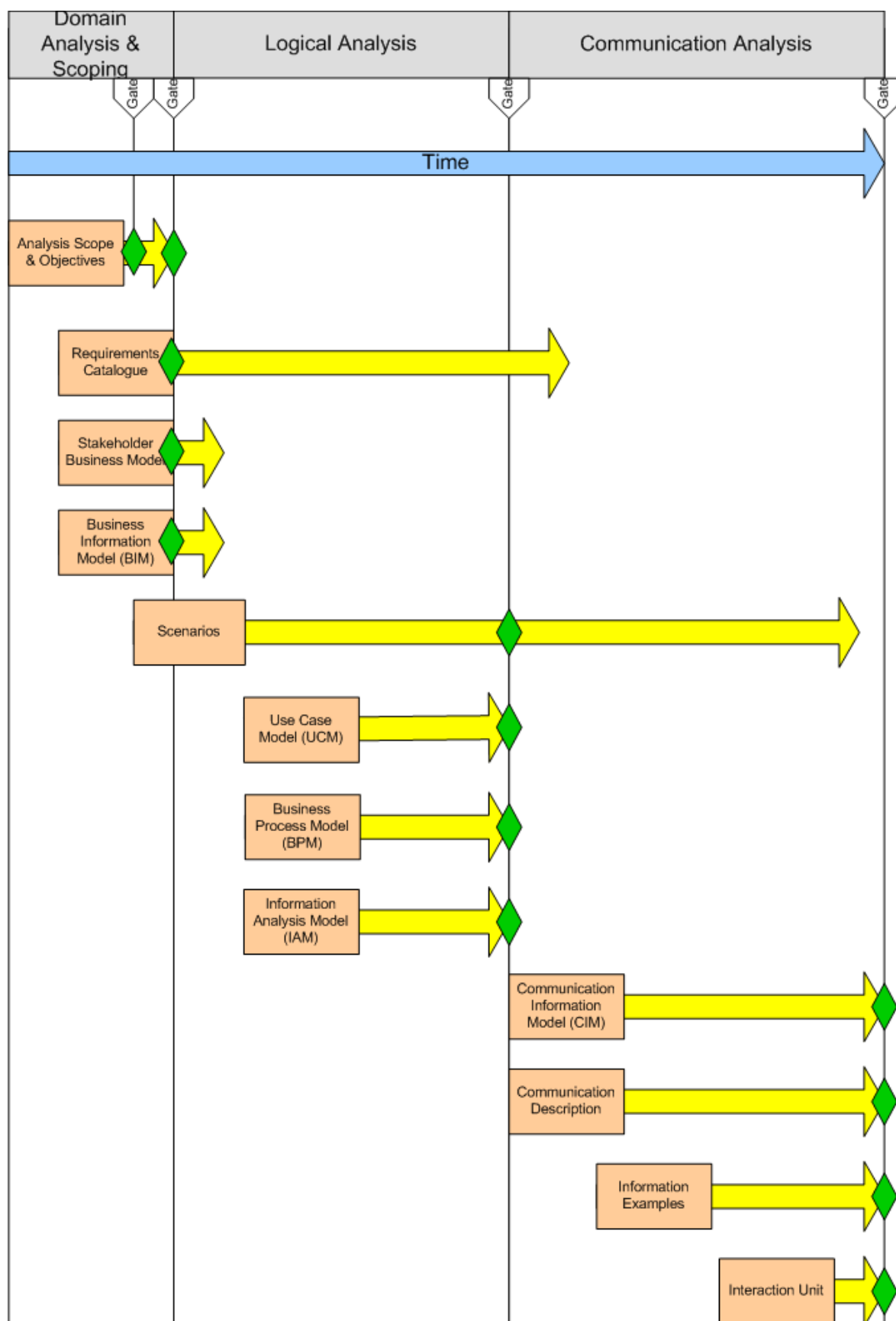


Figure 5: product phasing and approvals **TO BE DONE**

6 Publication of Models

6.1 Introduction

Models are to be published for external use as HTML reports, see §13.4.9. The method of distribution is the responsibility of the Owner and may include established mechanisms pertinent to the project. Models must be registered within the Model Library before distribution (see §6.3).

Where the model includes related artefacts (i.e. artefacts which are not produced using the modelling tool but are to be published with the model), these must be made available from the native and HTML forms of the model. The technique to be applied is to incorporate the related artefacts into the model as 'linked documents' which are directly accessible from the model navigation page (see §6.2), the method of document incorporation is described in §13.4.30. The artefacts to be linked in this way include the following:

- Requirements Catalogue (RTF Version) (see §9.5)
- Change History Register

Distribution of EA native files to suppliers and users (via normal communication channels and with appropriate guidance and support

NOTE: An EA native reader/viewer software is available from:

<http://www.sparxsystems.com/bin/EALite.exe>) is encouraged but at Owner discretion.

The process for publication of submitted and prepared models is:

- Sanity check of the requirements catalogue (including for consistency with the rest of the model)
- Sanity check of all model elements (completeness, logic, vocabulary conformant, English etc)
- Redundancy check of model including elements and links
- Notes check
- Incorporation of related artefacts have been added and resolve correctly
- Check top level Model Disclaimer and Copyright
- Generate HTML report
- Generate FileCM ID and attach or 'Up Issue' to the Model Library, (see §6.3) as appropriate (ensure that the FileCM ID is inserted within the top level note), for each of:
 - HTML reports should be added as a separate entity and within a ZIP file
 - EA native format
 - Related Documents

The formal publishing of a model will in future lead to freezing of the model for a period (to be determined by the Model Custodian). It is therefore usual for a model to have an internal review by the team concerned as well as by an analyst who is not a member of the models own development team prior to release for external review or approval.

6.2 Navigation Page – Model Default

6.2.1 Introduction

Analysis models can quickly become quite large and difficult to navigate. To improve this problem all analysis models must contain a navigation page. The navigation page is useful during the development of the model set, and after the final documentation has been produced, therefore the page should be created and maintained from the start of model development.

As the target audience of analysis artefacts may wish to access the artefacts via different routes - the navigation page displays artefacts grouped by both analysis phase and artefact type.

6.2.2 General Use

Links to groups of related diagrams will be displayed with an appropriate title/ supporting text.

Links to related documents will be displayed with an appropriate document artefact (see §13.4.30), this should be tailored to accurately reflect the analysis artefacts within the domain.

Acronyms are not to be used within the Navigation Page or the Package names within the Project Browser view.

Diagram notes required on the navigation page are described in §8.4.1.1

6.2.3 EA Use

The diagram type is a Package Diagram. New groups of links should be created using a diagram frame, given an appropriate name.

The navigation page is set to be the model default diagram, see §13.4.4.

It is possible to format the Hyperlink text and hide the icon see §13.4.16.

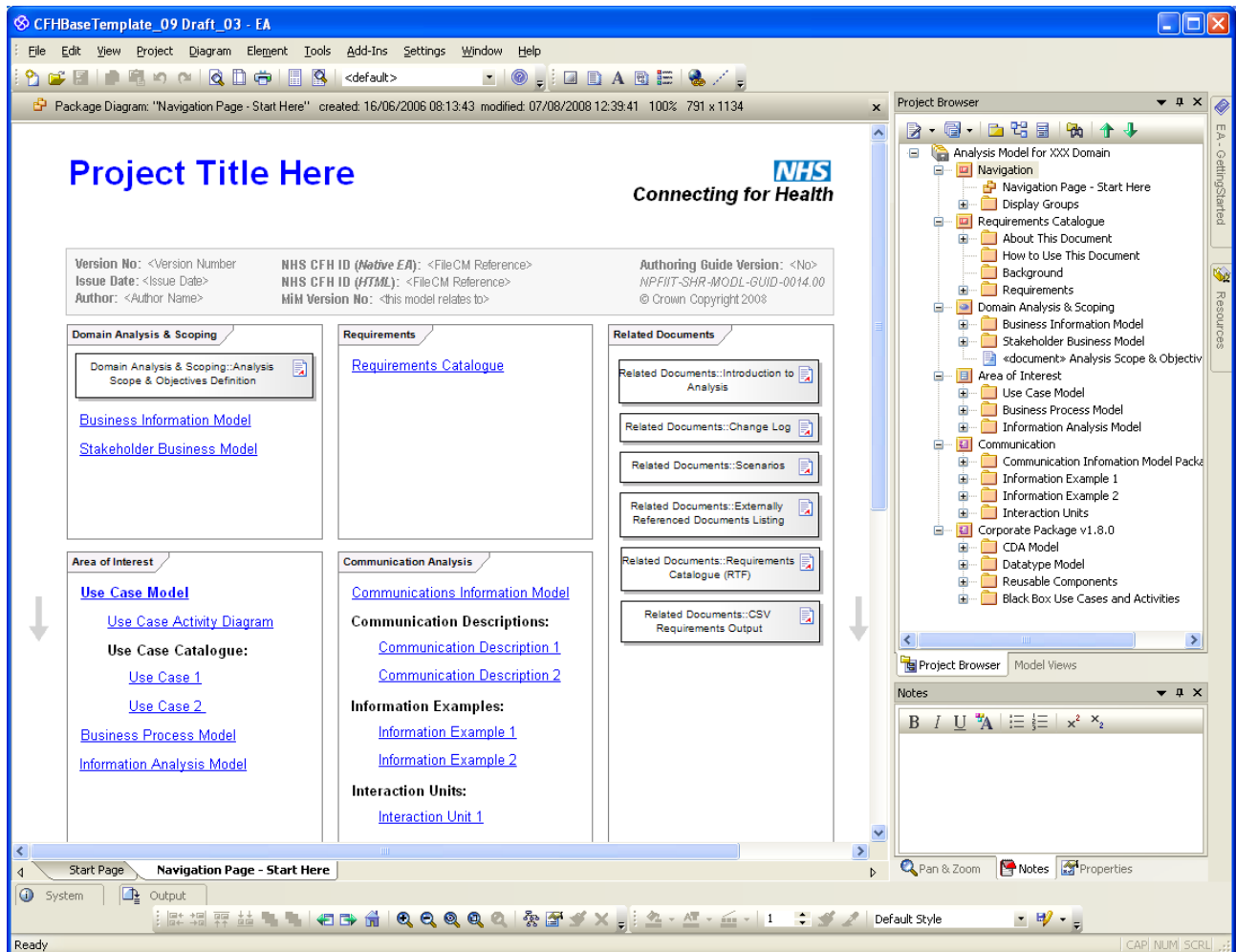


Figure 6: EA navigation page and project browser structure **TO BE DONE**

6.3 FileCM

6.3.1 Introduction

FileCM is the document management tool provided for use within the NHS Connecting for Health programme and is used as a model library by NHS Connecting for Health Business Requirements (see §3.3). The Analysis Team programmes have defined folder structures and metadata requirements within the product (see Model Library area for the folder structure) for current and historic analysis work; new domains can be added as required.

The following apply to analysis artefacts developed as part of the Analysis Team Programme.

6.3.2 Requirements

The following rules must be adhered to for all artefacts produced as a direct result of the analysis of the subject area, including discussion papers, diagrams, models and related documents.

Note that Analysis Artefact Checklists (see §7) are not subject to these rules.

Posting of the current version of the artefact to FileCM is required when one or more of the following occurs:

- On issue of the artefact for review to staff external to the immediate project team. For example, seeking review of an early draft from a group of domain experts
- On requesting formal review from the individuals nominated for that purpose.
- For an artefact that has previously been registered on FileCM, when a 'significant' change is made to the product. For example, incorporation of comments received from a formal review, changes made to a model as the result of a notable reduction or addition to scope etc
- On seeking appropriate Authority body approval for the artefact

Where an entry for the artefact already appears on FileCM, then the new attachment is to be 'up-issued'.

Where a model is to be entered into FileCM, if the model incorporates external artefacts (see §6.1), the version of these artefacts must be appropriate to the submitted model.

Note that where SVN is used to support the development of a model, the artefact for review should be extracted from the SVN repository and posted to FileCM. Any changes resulting from a formal review should be incorporated into the SVN Model. A new extract of the artefact should then be posted to FileCM.

6.3.3 Guidance

The following guidance is provided regarding the use of FileCM:

- Consider creating the FileCM entry and attaching the first draft of the artefact when it is first distributed for internal review to the project team.
- Discuss the FileCM requirements for other project documentation such as presentations, minutes etc with your Project Manager; generally these items should not be filed within the Model Library, it is recommended that the project SharePoint facility is used for this purpose.
- The metadata fields for Model Library areas of FileCM support the entry of the EA Version within which the artefact was developed and must be completed. Where no appropriate value exists its creation should be requested by email to 'ICT Service Desk'.
- If the artefact appears on the Analysis Artefact Checklist (see §7) this should be updated at the same time with the latest FileCM reference; in all cases the information on the checklist must not be more than 14 days out of date.
- Storage of Analysis Artefact Checklists within FileCM:
 - To update an existing checklist overwrite the previous file and do NOT use up-issue.
 - FileCM will not automatically change the meta-data field 'Date Received' to the date of the update so it must be entered manually.
 - Internet Explorer 6 or ThinApp Internet Explorer 6 (available on request from ICT to CfH users)

7 Analysis Artefact Checklist

7.1 Introduction

The Analysis Artefact Checklist is a spreadsheet which records the current state of analysis artefacts and storage locations (FileCM and SVN). It provides a high level snapshot of the artefact set and is made available to a wide internal audience so that downstream users of artefacts are able to see which artefacts are in production or planned.

It also provides a quick reference guide to the format, purpose and status of each artefact type.

7.2 General Construction

The template for this checklist must be used. See Template 2.

A checklist should be created for every active domain. Checklists should be updated at least every two weeks

The responsible Lead Analyst must review the checklist at least every 2 weeks and:

- If changes required:
 - make changes, update 'last changes made on' date and version number, reload (not up-issue) the checklist, change FileCM date and version number metadata to match spreadsheet
- If no changes required:
 - nothing to do on spreadsheet, update FileCM 'correspondence' field with today's date (to reflect the information has been checked, and is still current), no change to version number

Where multiple instances of one artefact type are created, these should be recorded on the checklist at the Analyst's discretion; suitable amendments should be made to the domain checklist, using the template as a base.

8 Analysis Models

8.1 Introduction

The following sections define in detail each of the various artefact types, and the specific artefacts within this methodology. Some general construction principles apply to all artefact types, and are detailed below.

8.2 General Construction

8.2.1 Element Identification

Each *major* element (see below) will be assigned an appropriate unique identifier, as the element alias see §13.4.1.

The identifier should be of a form, XXXnnnYYY where:

- XXX - Domain / Project Identifier (from Domain specific section of FileCM IDKey, N.B. this must be entered by the Analyst)
- nn – unique number
- YY - Artefact Identifier
- Applicable elements include: Activities (ACT), Use Cases (UC), Actors (A), Classes (CLA), Requirements (RQ) and Objects (OBJ)

8.2.2 Project View Naming

Acronyms are not to be used when naming packages within the project view

8.2.3 Business Rules

Business rules can be articulated in the following ways:

- As decision logic within a Query Activity/Action Diagram (see §12.3.4)
- As constraints applied against elements within static models (see §11.3.1.1.2.6).
- As Pre/ Post conditions and triggers within Use Cases (see §10.4.3.2).

8.2.4 Business Use Case

The term 'Business Use Case' is applied in many analysis methodologies and has been adopted by many IT organisations. This methodology does not formally adopt this term, however to accommodate interaction with suppliers it is necessary to generate a definition in the context of other artefacts. For the purpose of this methodology a Business Use Case is defined as a composition of the following 'required' artefacts ('required' is defined in §8.3):

- Use Case Catalogue (see §10.4.1)
- Use Case Models (see §10.4.2), including Use Cases (see §10.4.3)

Optionally, the Business Use Case may also include the following artefacts:

- Business Process Model (see §12.3.2)
- Information Analysis Model (see §11.4.2)

8.3 Analysis Artefact Status

Each analysis artefact falls into one of three categories; these are 'Mandatory', 'Required' and 'Optional', the category of the artefact is known as its *status* within the methodology. The status of each artefact is recorded on the artefact product description, see §17. A quick reference to this information is contained within the Artefact Checklist, see §7.

These states are defined in Table 3:

Table 3: artefact status definitions

Status	Definition
Mandatory	This artefact must exist for every domain under analysis
Required	This artefact must exist, unless the role of the artefact is fulfilled by an existing artefact. If a required artefact will not be created, the reasons for this and references to replacement artefact(s) must be stated in Work package
Optional	This artefact may be produced at the discretion of the project team, if it will support the existing artefact set and provide additional clarity of analysis.

8.4 Model Annotations and Addenda

Models have a wide range of information types that need to be captured and displayed (e.g. Notes, Descriptions, Requirements, Constraints, Author, Version information, Modification and Creation dates etc.).

This section provides details of how to handle, capture and display:

- Model Annotations
- Constraints
- Gap Analysis
- Embedded Documents and External Artefacts

NOTE: Element/artefact-specific requirements are also dealt with in the Notations sections for each element/artefact elsewhere within this document.

8.4.1 Model Annotations

Annotations within models have a number of uses. They can assist the understanding of model contents/elements, the business or information domain, model readability, and technical context as well as providing aide-mémoire/prompts during development. Essentially, annotations are used where a more structured approach is not practical. The general use of annotations within this methodology requires that:

- Annotations are either visible on diagrams or hidden within a models' documentation. They can be generalised (e.g. linked to a business or information context) or related to a specific model element.
 - The standard for visible annotations within UML 2.x diagrams is the folded-corner note box, free-floating (unattached) or attached to an element with a dotted line.
 - An object or class may also be used to display some types of annotated content in a delineated section of the element itself (processes dependent upon CASE tool used, see §13.1)
- Annotations on diagrams should be kept as brief as possible; if more detailed information is required then this should be provided by an incorporated or externally referenced document/artefact (see §8.4.5).
 - Too many annotations on a diagram may obscure its purpose; it is therefore advised that these are kept to a minimum, and that they should be short and to the point.
 - Adding a prefix/heading to define the reason for a displayed annotation to aid clarification and understanding is advised (see §8.4.1.1).
- All Annotations should be reviewed, and deleted as necessary prior to publication or release

8.4.1.1 EA Guidelines

It is required that all important annotations captured within a model's elements are recorded in the appropriate section of the properties dialogue, and displayed using an element-linked annotation. This ensures that the information retains a link to the element in RTF and HTML outputs, is recorded as part of the internal documentation for the model and reduces scope for element notes to conflict with notes displayed on diagrams.

- The content of annotations captured in the properties dialogue may be displayed on diagrams using an element-linked folded-corner note box (or optionally and only for general notes within the element using a separately delineated section of an appropriate element such as a requirement, class or object).
- Important annotations for a model should be captured either as a general note within the element (captured using the general tab of the properties dialogue) or as an additional constraints entry (see below and §8.4.1.2 for details of how to do this).
- In some situations the general notes within the element are specifically used to capture a Description of the element (e.g. in use cases and requirements). The addition of further information may mean that annotations cannot be easily displayed on a diagram because of the size or the mixed-nature of the dialogue content.
- In this instance additional annotations may be captured using the Constraint tab as follows:
 - The use of an additional entry using the Constraints tab of the element's properties dialogue to capture annotations is a work-around to enable the

- capture of additional information not suitable for capture within a general note within the element.
- Additional notes captured under the Constraints tab should have a type heading including the word 'Note' (e.g. Clinical Note or Working Note). This denotes to those using or reviewing the model that this entry is not a formal Constraint.
- The only elements that may not capture additional notes to the Constraints tab within the properties dialogue are Requirements and Diagrams – they do not have a constraints section to their properties dialogues. In these instances additional notes should be captured to the notes section of the properties dialogue. Analysts should consider using headings to delineate different content.
- When the general note within an element is displayed on a diagram, it will display the whole content of that section:
 - If the amount of additional information to be captured in the general note within the element is considerable, this may make the note unwieldy to display. The use of a linked document/artefact for non-displayed information is advised in these circumstances (see §8.4.5).
- The general note within an element section of the properties dialogue should not be used to capture formal Constraints which have their own tabbed locations within the dialogue (see §8.4.2, as well as artefact-specific notation sections throughout the document for further details).
- NB. Each diagram-visible instance of an element-linked note will be automatically updated whenever changes are made using the appropriate properties dialogue – they cannot be amended on the visible diagram annotation itself.
- Whilst other types of information captured in the properties dialogue may also be displayed in diagrams using an element-linked annotation, the only ones approved for use are: general notes within elements and constraints (including formal constraints, additional 'notes' and gap analyses).
- The content of general notes within elements may optionally be displayed using a delineated section of a Class or Object:
 - It is not recommended that this facility be used for Packages that are shown with their contained elements made visible, as these can obscure any additional annotation which may be displayed.
 - This facility should not be used to display Requirements, Scenarios or Constraints (This includes additional Notes or Gap Analyses created within the Constraints tab, as only basic headings are displayed and alternative representations are required)
 - Whilst it is possible within EA to display information of other types than just the general notes section within a class or object, this methodology currently prohibits the use of this facility for Constraints or Requirements (See §8.4.2 and §9).
- Annotations may also be created and populated directly on a diagram (see §8.4.1.2).

- This type of annotation should only be used for information having a low-level of importance that does not need to be related to that element or output in RTF or HTML formats (where this type of annotation is only represented as an anonymous item or as an unrelated general public note). This applies irrespective of whether a visible dotted line linking the annotation to an element exists.
- The Text Label function ('A' on tools menu bar) should not be used to capture diagram notes, in order to avoid confusion with swim-lane titles, decision element text etc., (This type of notation is also not linked to an element in a model's internal documentation and therefore should not be used for important annotations).
- It is recommended that annotations displayed on diagrams optionally use prefixes or headings to assist reviewers in understanding the reason why they have been placed there (e.g. Working Note or Clinical Note). A set of annotation heading titles has not been formally defined and is left to the discretion of the Analyst; however Analysts are advised to identify items such as: Clinical, Working, Business/Domain or Technical information.
- If an element is copied and pasted as a new instance (rather than a linked one) to another location then subsequent changes to annotations in the properties dialogue or any instances of its information (visible or not) will not be updated
- Each diagram has information about its title, author, version and creation/modification dates held internally within the model.
 - These should be displayed on diagrams prior to review or publication (except for the model navigation page which has its own display format see §6.2).
 - It is suggested that this information is placed in the top left corner of the diagram.
- Text formatting functions are available and may be useful for highlighting different types or priorities of information. The enhanced text functions are only available where they are displayed as options within the text creation dialogue:
 - The formatted text will retain its formatting in RTF and HTML outputs
- All annotations can be reviewed in the project Notes window (whether entered either directly onto a diagram or using elements properties dialogue) by clicking the element on the diagram and selecting the View | Notes item from the main top menu. This opens a project Notes window to the right of the EA workspace (where the project browser usually is). Clicking on other elements in the diagram will display their Notes content also.
- When an element is copied any folded-corner note attached to it on a diagram is not automatically copied; but if the note is subsequently copied and pasted to the same diagram as the copied element, the link is re-established.
- The Navigation Page (see §6.2) must contain the following information, see also §13.2.

IMPORTANT NOTE

This model has been developed in consultation with the design teams of the

Authority and its contracted suppliers. The behaviour and utilisation of CRS services has been modelled in accordance with information given by these teams and the models do not in themselves constitute design guidance or requirements.

And

Status MODEL Vn.n Date of Issue
 NHS CFH ID from FileCM for native model
 NHS CFH ID from FileCM for HTML representation of the model
 SVN Location (where applicable)
 Owner: <Project Manager name>
 Author: <Author name>
 © Crown Copyright YYYY

Where part of an analysis model is circulated, it is left to the Analyst's discretion to ensure correct version control information is available on the circulated models. If diagram notes appear anywhere other than the Navigation Page, these notes take precedence for the diagram on which they appear.

8.4.1.2 EA Construction

8.4.1.2.1 Folded Corner Annotation Boxes

Folded-corner annotation boxes are created and populated in different ways dependent upon how they are to be used (see §8.4.1.1). The approach to creation of folded-corner note boxes is as follows:

- Important information captured in sections of the properties dialogue can be displayed on diagrams within a folded-corner note box that is used as an element-linked item – to do this:
 - Drag a note from the common section of the tool palette onto the diagram, select *Cancel* on the displayed dialogue. Then create a note-link by dragging the arrow at the top right of the note box to the target element (unless you are linking it to the *Diagrams* own properties dialogue – see below). Right-click the link and select the item: *Link this note to an Element feature*. From the *Feature Type* drop-down list shown choose the type of information you are seeking to display (e.g. *Element Note*, *Constraint* etc.). If there are one or more items available you must click on the relevant one shown in the *Feature* list and click OK. The displayed note will now automatically derive its contents from the information stored against the target element.
 - To display the information for the diagram itself, recorded against the *general notes* section of its own properties dialogue: Create a note on the diagram from the Common section of the Toolbox and select *Cancel* on the displayed dialogue box; then right-click the blank note icon and select: *Advanced | Link to Diagram Note*.
 - Folded-corner note boxes with information entered in a section of the elements properties dialogue can only be amended by:

- Selecting the parent-element, right-click on the element (or double click the mouse on the element) opening the properties dialogue and then finding the appropriate section amend the displayed text
- NB: it is not possible using the F2 Hot – key facility to amend linked-element notes displayed on diagrams (see §13.4.23)
- Folded-corner notes used to display annotations having a low-importance to the final model can be created and populated directly on a diagram e.g. working notes or aide-mémoire. (NB: Annotations created and populated in this way are NOT linked to any specific element or the models internal documentation when represented within RTF/HTML outputs):
 - Drag a note from the Common palette onto the diagram; enter the content in the dialogue presented. To link it visually to an element drag the upward-pointing arrow (top right) to the element – a link will appear, Or
 - Right-click the element and select *Add / Note*; enter the content in the window presented
 - Annotations created in this way may be edited directly on the diagram by clicking the note and amending the text in the box shown or by selecting the content and clicking on the *Function F2* key (see §13.4.23)

8.4.1.2.2 Other Annotation Items

- To display detailed information about the diagram itself (author, date created etc.)
 - Select either: *Diagram / Insert Property Note* from the top main menu; Or
 - Drag a *New Diagram Note* item from the *UML Elements* toolbar at the top (only a limited subset of this information is shown if this is displayed using the *Diagram / Show Diagram Details* tab in the properties dialogue)
- To display the General | Notes section of the properties dialogue within a delineated section of a class or object (for conditions see §8.4.1.1)
 - Right-click the element and using the *Set Feature Visibility* menu item. Check the box against the item you wish to display (e.g. *Notes* to display the general notes section)

8.4.2 Constraints

Constraints define how the various elements within a model may operate individually and / or collectively. In addition to the guidance given here different elements and artefact types have their own specific requirements and rules regarding whether a constraint is applicable and how it should be managed; these are detailed against each elements / artefact type elsewhere within this document.

The most common types of constraints used in this methodology are:

- Constraints (general)
- Dependencies
- Datatypes
- Examples
- Representations

- Goals
- Triggers
- Pre and Post Conditions
- Goals
- Descriptions
- Cardinalities – although these are a form of constraint they have their own requirements and are not dealt with in this section (see §11.3.1.2.1)
- Gap Analyses – see §8.4.3
- Additional Annotations (see §8.4.1.1)

Constraints may be applied to the following model element types:

- Use Cases
- Actors
- Activities
- Flows
- Objects
- Classes
- Attributes
- Packages
- Traces

Constraints may *NOT* be applied to the following elements:

- Associations (although additional annotations with the word 'Note' in their type heading may be added)
- Requirements - these are treated separately within this methodology (see §9).

8.4.2.1 EA Guidelines

For the sake of consistency and ease of use it is important to be able to display and to locate constraints in the same place across models. This is especially important where a model is being revised from a previous version or has a new author.

- Constraints must only be recorded within the elements properties dialogue under the Constraints tab. This facilitates safe capture and documentation against a specific element; this is particularly important for RTF and HTML outputs (see §8.4.2.2). The addition of constraints directly onto diagrams or using general notes within elements is specifically prohibited.
- The content of a constraint may be displayed on a diagram using an element-linked folded-corner box for an object, class or package (See §8.4.1.2)
 - It is advised that as few formal constraints as possible are displayed on diagrams to avoid diagram clutter and confusion with other annotation types
 - It is not possible for a constraint captured against an attribute to be displayed on a diagram – these can only be viewed using the properties

- dialogue under the *Constraints* tab (however, an element-linked note can display the content of an attribute's general note).
- Display of constraints within a delineated section of an element (i.e. using the Set Feature Visibility facility to display constraints) is prohibited.
 - Additional annotations may also be captured to the Constraint tab and must have the word 'Note' in its Constraint Type, to clearly identify that it is not a formal constraint.
 - These may be displayed on a diagram using an element-linked annotation (see §8.4.1)
 - This annotation type may not be displayed within an element (class or object) itself – as the full text is not available. Display of a Constraint tab created additional annotation can be achieved using an element linked note created on the diagram and selecting the specific linked note.
 - Where constraints relate to another element of the model that element must be referenced, and:
 - Reference the other attribute using the full and exact attribute name
 - Reference the other class using the alias of the class.
 - Reference any attribute in another class using the alias of the class and the full and exact name of the attribute.

8.4.2.2 EA Construction

- To record a constraint against an element (see Figure 7)
 - Choose the Constraint tab in the element properties dialogue. A short and meaningful title should be entered under the heading *Constraint*. (NB. to display this constraint on a diagram using an element-linked folded-corner box it is advisable to describe the constraint type within the title so it can be more clearly understood – e.g. *Pre-Condition: then Your Title*)
- The addition of constraints directly within diagrams (by right-clicking and using the *Add* facility) is prohibited, in order to prevent this content from not being reflected in documentation.
 - The kind of constraint can be defined by using the Type drop-down menu and choosing an appropriate type – this may be left blank.
 - The current status of the constraint may be shown using the Status drop-down menu by choosing from the list displayed – this may be left blank; however if this is used then the displayed status will require updating prior to release of a model for review or approval.
 - A short narrative should be added in the open notes space to define the constraint - Constraints should be written in everyday readable English (at some point in the future the use of more structured 'machine-readable' constraints such as Structured English, OCL or RegEx may be adopted in addition to narrative constraints)
 - Further constraint type headings may be entered for use with the Constraints tab of the properties dialogue by selecting General Types from

the Settings menu. (N.B. New headings under this section should not seek to replicate information which is more appropriately carried elsewhere within the model e.g. Requirements)

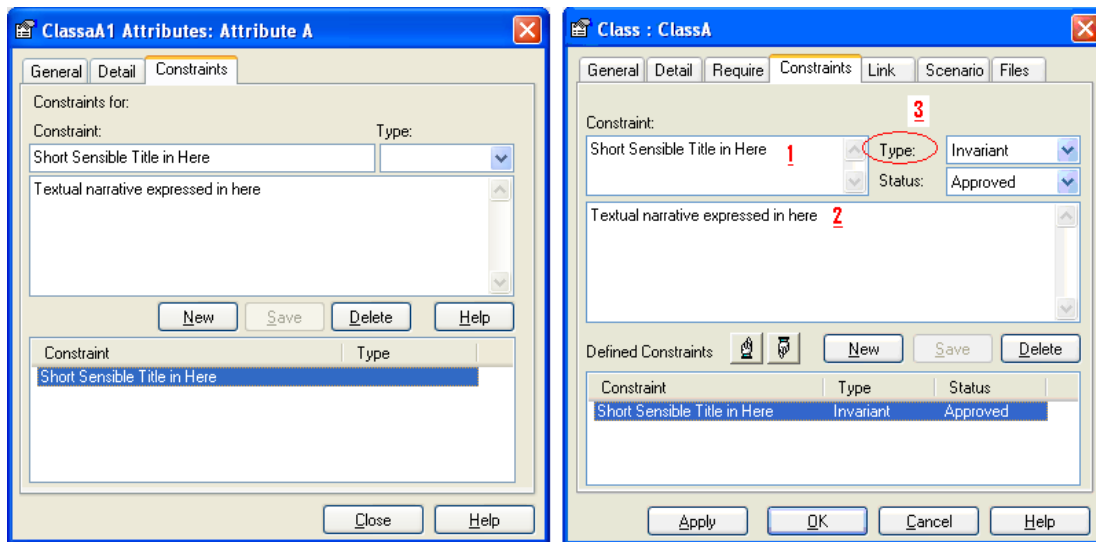


Figure 7: recording constraints

8.4.3 Gap Analyses

Gap Analysis aims to identify and quantify the 'gap' between the interim result of analysis and some form of control set (for example Completed Data Elements that may already exist). The aim is usually to clarify resource/timing assumptions and to direct further analysis work, including the updating of the control set.

Gap Analyses may need to be recorded within models against any type of element (e.g. Diagram, Object, Class or Attribute etc.), particularly following changes in methodology or other related technical processes. As these may have an influence upon further analysis or subsequent design work they are treated as Constraints with a Type of 'Gap Analysis'

8.4.3.1 EA Guidelines

- Gap Analyses recorded within models must be classified as a constraint of type 'Gap Analysis' using the properties dialogue of the element (see §8.4.2.2) to facilitate RTF HTML reporting and documentation.
 - The description of the *Gap* should describe what it is and/or how it is to be resolved (e.g. this is now realised by: Completed Data Element)
- If required, Gap Analyses may be displayed on diagrams using an element-linked folded-corner annotation box (see §8.4.1.2) in the same way as any other constraint
- A Gap Analysis entry should persist only for as long as it is necessary to define how a specific element is to be realised. Only when it is fully resolved and / or the element / structure has been removed from the model should the Gap Analysis be deleted
 - Where an issue highlighted by a *gap* has been resolved and it is no longer appropriate to show it, the resolution should be recorded in the *Related Documents: Change Log* on the models *Navigation* page.

8.4.4 Test Conditions

It is common during analysis activities to identify conditions which are likely to be required/of-interest to testers and can be used to inform the creation of test scripts and test data. An example is the value of an attribute constraining the value of other attributes.

Where such conditions are identified during analysis, they must be recorded within the analysis model as a constraint on the subject element (see §8.4.2). Furthermore the 'type' of constraint must be set to 'Constraint'.

8.4.5 Embedded Documents & External Artefacts Usage within Models

Non-UML diagrams or large amounts of textual information may need to be incorporated/ embedded, attached to or referenced by a model see §8.4.5.2 (exactly how this is achieved depends on the CASE tool being used). Information created or held externally to a model creates maintenance and versioning issues that need to be taken into account before being incorporated into or referenced by models.

8.4.5.1 EA Guidelines:

- An artefact or document can be incorporated or embedded into a model as an RTF formatted element which can then be accessed using a visible diagram Artefact/Document element (see §13.4.30).
- Externally sourced or referenced Artefacts or Documents:
 - Where possible an externally sourced artefact/document should be incorporated into a model to ensure that the specific context/version being referred to is clear (see §13.4.30).
 - If it is not possible to embed an external artefact/document within the model then a copy of it should be placed onto the NHS CFH document filing system (currently FileCM) and it should be referenced in the External References document held on the Navigation page. NB. It is also possible to create a relative link to a local copy §13.4.32).
 - A reference to any artefact/document held or created externally (see §13.4.33) should have its details recorded in the External References document held on the Navigation page, in addition to where it has been referenced within the model.
 - If for copyright reasons it is not possible to incorporate into the model or place a copy of an external artefact onto the NHS CFH document filing system (currently FileCM) then the fullest reference possible should be recorded on the External References document held on the Navigation page.
- Links to externally held documents or files from within a model:
 - Active hyperlinks or URL's (see §13.4.33) to external documents or files are not recommended (except where the link is to a copy which is distributed with the model or model output, see §13.4.32) and should not be used in preference to embedded documents.

- Whilst it is possible to embed other file types as OLE items (Object Linking and Embedding) within an *Artefact - Linked File* it is not currently recommended and should not be used if an alternative exists.
- Internally created Artefacts or Documents
 - These are items generated within a model during its development (e.g. Information Example RTF outputs) and held within *Artefacts* (see §13.4.30). They may be linked on diagrams to other elements by associations or left free-floating.

8.4.5.2 EA Construction

- Embedding an RTF document using a diagram element with a linked document:
 - Drag a Document element onto the diagram from the Common toolbox and enter a name for it in the properties dialogue. Embedded documents should be clearly labelled as a DOCUMENT or ARTEFACT along with a name of the analysts choosing in the Name field of the artefact elements properties dialogue (e.g. 'DOCUMENT – Information Example').
 - The blank document can be populated by double-clicking on the artefact element, which brings up a document window. Then the text can be entered directly into the window or by right-clicking on the document space and making a choice from the displayed menu options (Also see EA Help *Linked Document*).
- Embedding an RTF document within a diagram element
 - Right-clicking the element and selecting the *Create Linked Document* tab, the document can then be populated as for the previous item

8.5 Model Risks, Issues and Changes

During the construction of an Analysis Model it is likely that risks, issues and changes will occur. To ensure that these are recorded and articulated consistently during the model construction, these are to be entered into the model as described in this section. The underlying principles of this approach is that where possible this type of information should be articulated as metadata associated with model elements/structure; should be accessible from one place; and at the point of completion/publication should be deleted from the model as being either resolved or passed to the Owner of the model.

Adopting these principles means that features of the selected CASE tool (EA – see §13.1), determines how specifically risks/issues/changes are recorded and output, however where no appropriate facility or element/structure exists for recording this information it is possible to create an annotation (see §8.4.1) on the model Navigation Page (see §6.2) for this purpose.

NB: The facilities outlined in this section are internal administrative processes to support model development by the Analyst and within the project team. Whilst they are useful tools they do not replace the formal NHS CFH reporting processes. The aim should be that all risks, issues or changes items started during the analysis phase should either be resolved or transferred to the appropriate external reporting mechanism if a resolution is not possible before analysis is concluded.

8.5.1 Risks

Where a risk is identified, this must be associated with the relevant element/structure within the model. This is achieved in the selected CASE tool (see §13.1) as described in §13.4.43,

Where the risk is associated with a diagram/phase it is expected that the risk will be associated with the folder in the project browser structure (see §6.2.3) containing or representing the diagram/phase.

Outputting risks from the model involves the application of the RTF Reporting mechanism (see §13.4.11).

8.5.2 Issues

8.5.2.1 General

Where an issue is identified, this must be associated with the relevant element/structure within the model, or identified as a general risk. In the selected CASE tool (see §13.1) this is achieved as follows:

- Issues associated with elements/structures are created as described in §13.4.44.1, Where the issue is associated with a diagram/phase it is expected that the issue will be associated with the folder in the project browser structure (see §6.2.3) containing or representing the diagram/phase. Outputting issues (associated with elements/structures) from the model involves the application of the RTF Reporting mechanism (see §13.4.11).
- General issues are created and output as described in §13.4.44.2

8.5.2.2 Shared Environment

The LRA project adopted a shared environment (SVN) and a common CASE tool (see §13.1). This facilitated a cross team working environment through the use of a Collaborative Communication Log that utilised the 'Issue' functionality of the tool. This was created to enable members of the knowledge modelling, technical modelling and terminology teams to obtain clarification, share ideas and update the modelling status of each data element content requirement. Although it is unlikely that there will be such an environment for Phase 1 of LRA a similar style of working would be beneficial. Consequently, the approach taken for Release 3 of the project is documented in §14 for consideration.

8.5.3 Changes

This section exists in addition to the consideration of model Change History documents required for submission of models into review/approval gateways (see §5.2).

Where a change is raised/identified, this must be associated with the relevant element/structure within the model. This is achieved in the selected CASE tool (see §13.1) as described in §13.4.45

Where the change is associated with a diagram/phase it is expected that the change will be associated with the folder in the project browser structure (see §6.2.3) containing or representing the diagram/phase.

Outputting changes from the model involves the application of the RTF Reporting mechanism (see §13.4.11).

8.6 Referencing Analysis Model Elements

8.6.1 General Construction

This methodology requires close coupling between analysis artefacts and therefore a degree of traceability of elements throughout the artefact set. Recording the relationships between analysis elements is important for several reasons:

- Facilitates internal consistency across the artefact set.
- Enables compliance with artefact quality criteria.
- Domain experts/ users will be able to follow processes and stated requirements from overview to detailed level,
- Reviewers/ subsequent users of the artefacts will find each analysis artefact easier to relate to its predecessors and peers.
- Information gained during the analysis is captured and relationships between elements are clearly stated, and don't need to be inferred by the reader.
- Maintenance of the artefact set is easier, both during initial analysis and in future analysis (perhaps by another analyst).

Relationships that may exist between analysis elements are shown in Table 4:

Table 4: analysis element relationships

Source	Source Element Type	Target	Target Element Type	Notes
SIV	Class	IAM	Class	All classes in the IAM must be derived from a class in the SIV. Multiple classes in the IAM may be derived from a single class in the SIV
SIV	Class	Actor Package	Actor	Actors within the actor package may also be captured as classes in the SIV
IAM	Class	Domain Information Description (DID)	Class	A class in the DID package must be derived from classes in the IAM. Multiple classes in the DID package may be derived from a single class in the IAM.

BPM	Activity	Use Case Model	Use Case	Each Use Case must have equivalent activities on the BPM
-----	----------	----------------	----------	--

8.6.2 EA Construction

The relationships between analysis model elements must be created using the Relationship Matrix functionality. See §13.4.27

8.7 Referencing Externally Managed Artefacts

Where the Analyst is confident that an externally managed artefact meets the domain requirements, this artefact should be referenced within the analysis model, and should not be duplicated within the model contents. Typically this could be Stakeholder artefacts or documentation that is equivalent in whole or part to the SBM, Requirements Catalogue or SIV. NB where the artefact is managed within the project/domain the guidance in §6.1 applies.

9 Requirements

9.1 Introduction

A requirement is a single statement of need for what a product or service should be or do by identifying a necessary capability, characteristic, or quality of a system in order for it to have value and utility. Requirements provide the input for Analysis activities/artefacts and are articulated within a Requirements Catalogue (see §9.5).

9.2 Requirements Definitions

9.2.1 Requirement

A requirement is an element that defines a required capability, characteristic or quality. Requirements are minimally composed of a unique identifier and a description. The requirement may be classified as being:

- functional: describing what capabilities must be executed, or
- Non-functional: describing what constraints or quality characteristics apply.

However, for LRA it is not the intention of the analysis to design a system. The focus of the analysis is to identify the information set that systems will be required to support. Consequently, a further classification of requirements is also relevant for LRA, as follows:

- National Requirement References: A statement copied from a national source document that provides guidance with respect to what may reasonably be expected to be in the content scope of care records data (i.e. the scope of potential national data standards).

Note: Some National Requirement References may describe specific information objectives, e.g. from a secondary use analysis perspective.

For the LRA, appropriate national source documents include (listed in order of priority):

- Legal requirements
- Professional regulations
- Professional guidelines or standards
- National Output-Based Specifications for care record systems design
- National secondary use requirements
- Expressed requirements: are gathered at the beginning of the analysis work, are expressed by the business users and drive the analysis.
- Derived requirements: are derived from the analysis and need to be communicated to the recipients of the analysis model (typically suppliers/developers); these often include usage instructions and business rules.

- **Informatic Concepts:** are words or a phrase expressed in the stakeholder's natural language that describes an area or subject that may be important from an informatics perspective. More formal analysis and elaboration in successor artefacts will determine the extent of its importance.
- **Query Requirement:** Essentially a specialisation of a Derived or Expressed Requirement that indicates a requirement to extract information from systems to inform clinical, management and policy monitoring and decision making.

9.2.2 Package

A package is a container of elements such as requirements. An organising principle is imposed upon requirements by their grouping into different packages. The package will have a descriptive name, a requirements diagram (see §9.3.1) and contain related types of requirements.

9.2.3 Requirements Diagram

Each package can be depicted with a diagram. However visually this may not be helpful if there are a lot of requirements in a package so consideration could be given to use the 'view diagram as element list' in the Case Tool.

For diagrams that show traceability it is possible to also show the use cases (or other elements) that realise the requirements or elements that generated (are the reason for) the requirements.

9.3 Requirements Notation

Generic notation applicable to requirements is described in this section.

9.3.1 Core Notation

Figure 8 is an example requirements diagram:

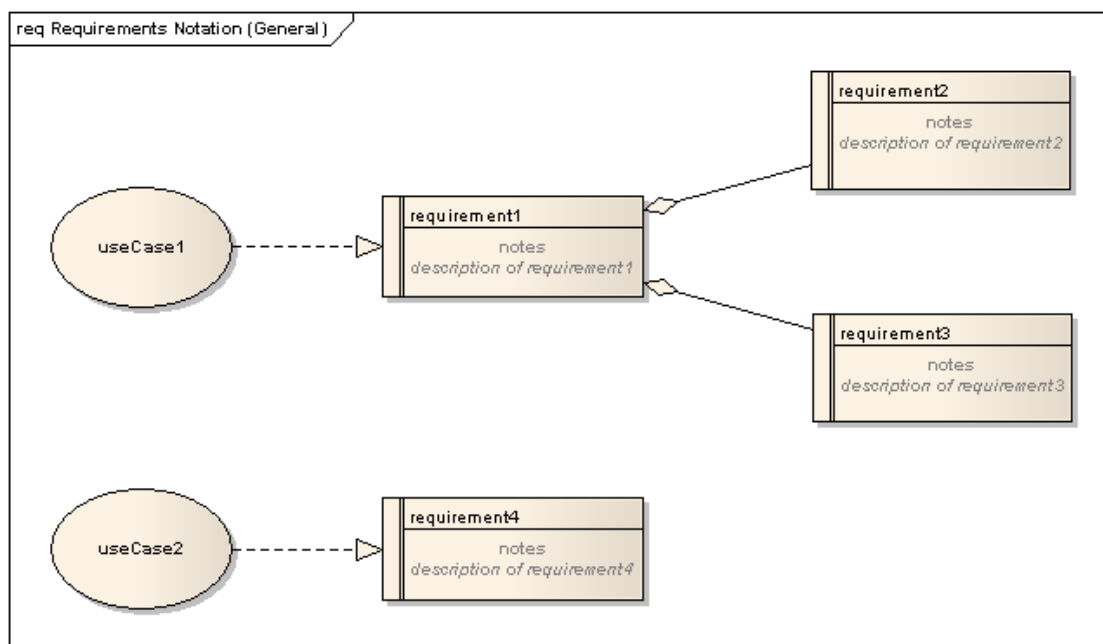


Figure 8: requirements diagram

Requirements are represented on requirements diagrams as boxes with a title bar and a vertical section on the left. The title bar contains the name of the requirements, which will be prefixed by a unique identifier (see §8.2.1). The main section of the box will contain notes describing the requirements.

Other elements are represented with their standard notation (e.g. use cases are represented by ellipses).

9.3.1.1 Requirements Aggregation Association

Where it is necessary to show that one requirement is made up of several other requirements an aggregation association will be used. This may not be needed when the requirements are first defined (e.g. they may start as high level requirements and be given more details at a later stage in the project).

An example of this may be where an Expressed Requirement has been further broken down into more explicit Derived or Query Requirements.

In **Error! Reference source not found.** requirement1 is made up of requirement2 and requirement3

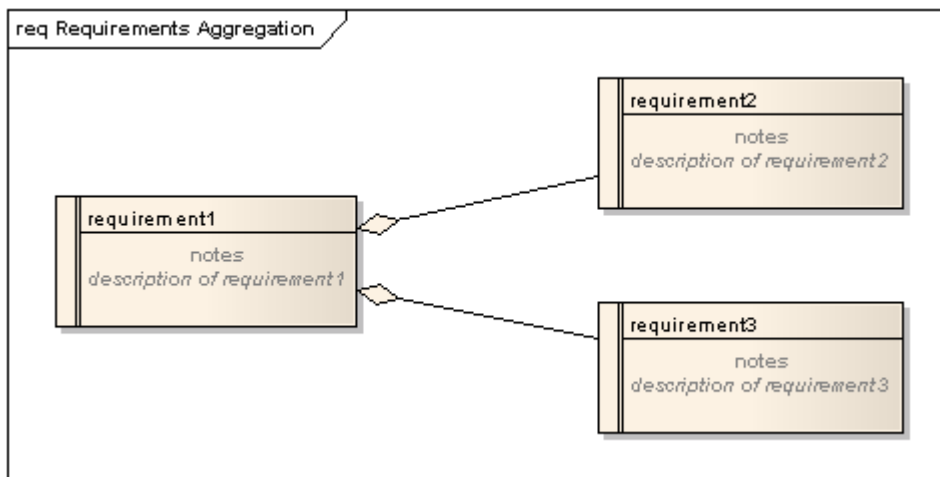


Figure 9: aggregation

9.3.1.2 Realisation Association

A realisation association is created from an element (typically a use case) to a requirement in order to show which elements realise the requirement; for example a requirement to display results to a user may be realised by an output results use case. An element may realise multiple requirements and a requirement may be realised by multiple elements.

In Figure 10 requirement1 is realised by useCase1.

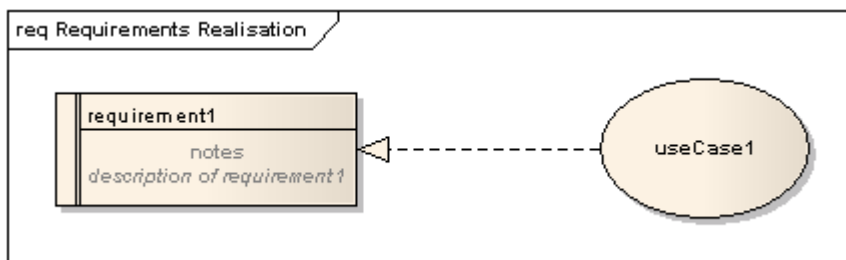


Figure 10: realisation

9.3.2 Toolset

Requirements shall be created by the preferred CASE tool (Enterprise Architect) see §13.1 and maintained within the native file of this tool and output as with other CASE tool artefacts see §6

9.3.2.1 Hints and Tips

Enterprise Architect (EA) provides a number of relevant features some of which are identified below:

- It is required to set the properties of requirements diagrams such that notes in elements are displayed (see §13.4.6).
- To make an association directional, select the Association Properties/ General tab and change the Direction field as appropriate.
- Displaying a Requirement already captured in the Requirements Catalogue on another diagram type can be achieved as follows:
 - Drag the Requirement from the Project Browser onto the intended diagram as a simple-link – it will be displayed as a Requirement element (a box with two vertical lines across it)
 - Selecting the arrow at the top right of the displayed element, drag it to the target element and select the Realization link. This link will also auto-populate an item under the Require tab in the properties dialogue for that element
 - To display the detailed content of the Requirement, right-click the element and then under the Set Feature Visibility item check the Notes box shown
 - The content can be amended at this location through its properties dialogue by double-clicking the displayed element (this will amend its internal documentation content and thus also its content wherever it is displayed).

9.4 Notation Adoption

Requirement elements are adopted as described in the core notation (see §9.3.1). In addition, the following conventions are adopted:

- the location of the elements within the project structure is denoted on the diagram as shown in Figure 11 and described in §9.5.1.1
- elements are identified with a unique alias as shown in Figure 11 and described in §8.2.1
- the following additional attributes should be recorded as part of a requirement (in addition to the identifier and description):
 - Short Description: the name and identifier of the requirement. The identifier should be the same as the alias (generated by the auto-name counter). If an existing notation is used for the project then this can be used. The format should be the identifier, followed by a colon and a space, followed by a short, descriptive name, as shown in Figure 11.
 - Alias: provides unique identification for the requirement as shown in Figure 11 and described in §8.2.1
 - Status: defined in the context of the project or domain. The specific use of this field is project/domain dependent, but as a minimum should reflect whether the requirement is in scope or not. If no specific usage is needed then 'Proposed', 'Approved' or 'Deprecated' should be used.
 - Type: The type field may be used to define the type of requirement as either a Functional or Non-Functional or Report requirement.
 - Phase: represents the project phase to which the requirement is associated and is thus project/domain dependent. If no specific usage is needed then '1.0' should be used.
 - Stereotype: Should be either 'National Requirement Reference', 'Expressed Requirement', 'Derived Requirement', 'Informatic Concept' or 'Query Requirement' depending on the type of requirement.
 - Notes:
 - National Requirements Reference: Should summarise the scope, purpose and version of the source document.
 - Expressed Requirements: Should contain the specific text that has been taken for the National Requirements Reference and a reference to the source document (e.g. NICE Guideline 39 version xx)
 - Derived Requirement and Query Requirement: fully describes the requirement using 'Must', 'Should' and 'May' terminology (described in §9.5.1.2)
 - Informatic Concept: key words or a phrases expressed in the stakeholders natural language taken for either the expressed, derived or query or requirements that describes an area or subject that may be important from an informatics perspective

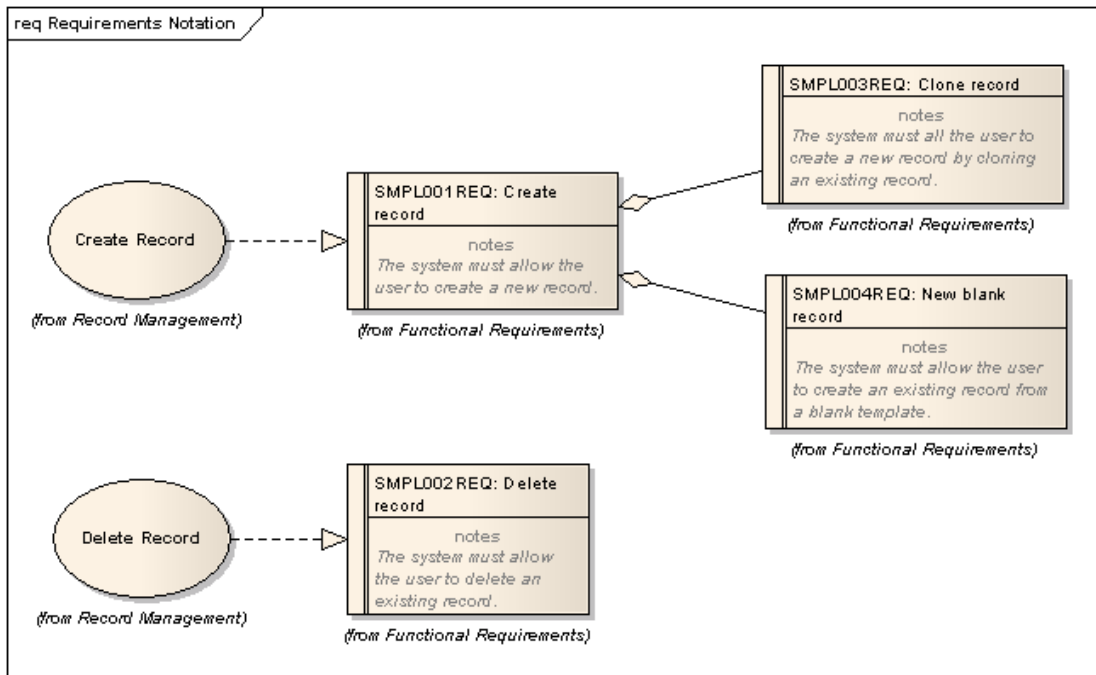


Figure 11: adopted requirements diagram notation

- Requirement elements (within a package) should be shown with notes visible (see §13.4.6) and all associations between them represented. Requirements from other packages may be included only to show associations between requirements.
- It is expected that all current and in-scope expressed, derived and query functional requirements will be realised by Use Case element(s). It is also expected that Informatic Concepts that are considered important will be realised by Class elements in the Information Analysis Model. Current and in-scope expressed non-functional requirements can be realised by various element-types including Use Cases (where the realisation is represented as use case constraint e.g. a pre-condition). These non-requirement elements will only be included at the relevant stage in the analysis process (i.e. once the element has been created within the model). Requirements marked as 'out of scope'/'deprecated' are not expected to be realised. Where a use case realises a requirement that has aggregate parts (see Figure 9) and the use case realises all of the parts then only the aggregate requirement needs to be associated with the use case. If separate use cases realise different aggregate parts then separate associations must be made.
- Derived and Query requirements may be realised in the same manner as expressed requirements.
- Requirements must not be deleted from the model; however they can be deprecated (made redundant without removal). The deprecated requirement should be removed from any diagram whilst also ensuring that any valid associations (or links) to the deprecated requirement are maintained.

9.5 Requirements Catalogue

The only instance of requirements within this methodology is as part of the Requirements Catalogue artefact, the recording of requirements using any other formalism is prohibited. A Requirements Catalogue is developed in order to contain the complete set of requirements for the project/domain in a form which is understandable to business users, an input to Analysis and as the basis for demonstrating the fulfilment of requirements.

The Requirements Catalogue has an internal structure of packages which group together individual requirements and each package contains one or more diagrams showing how requirements are associated with each other and with other model elements (for example Use Cases 'realising' a requirement).

The Requirements Catalogue is produced over two phases:

- Domain Analysis and Scoping Phase where:
 - National Requirements References are identified
 - Expressed requirements are taken from the National Requirements Reference source documentation
- Logical Analysis Phase where:
 - Derived and Query Requirements are elaborated from Expressed requirements (where required).
 - Informatic Requirements are derived from Derived Requirements.

9.5.1 General Construction

9.5.1.1 Requirements Structure

The requirements catalogue is held in a Domain Analysis and Scoping package named 'Requirements Catalogue'. It will comprise one or more requirements held in sub-packages, where these packages are used to group requirements of a similar type. Each package may contain one or more diagrams and a set of packages containing either requirements or other packages (but not both). Additional construction constraints include:

- where a requirements palette exists (depicting sub-packages), no other diagram must exist at this level in the hierarchy and the requirements palette must be the default diagram when the package element is 'drilled down'
- each package must include a brief description in the notes field of the package

9.5.1.2 Requirements Content

Expressed Requirements must contain the National Requirements Reference content to which they refer.

Derived and Query Requirements must be expressed using the words MUST, MAY, and SHOULD. These words are to be interpreted as described in RFC2119 (see Related Document 4):

- MUST: This word, or the terms 'REQUIRED' or 'SHALL', means that the definition is an absolute requirement of the specification (the term 'MUST' is expected to be used to define this priority level).

- **SHOULD:** This word, or the adjective 'RECOMMENDED', means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course (the term 'SHOULD' is expected to be used to define this priority level).
- **MAY:** This word, or the adjective 'OPTIONAL', means that an item is truly optional. One implementer may choose to include the item because a particular implementation requires it or because the implementer feels that it enhances the implementation while another implementer may omit the same item. An implementation which does not include a particular option **MUST** be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option **MUST** be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides (the term 'MAY' is expected to be used to define this priority level)).

The negation of **MUST** and **SHOULD** (i.e. **MUST NOT** and **SHOULD NOT**) are allowed, but the negation of **MAY** (i.e. **MAY NOT**) is prohibited.

The use of **MUST**, **SHOULD** or **MAY** within a textual requirement description must always reflect that used to specify the priority of a requirement in its tagged value e.g. if the tagged value priority says **SHOULD** the text cannot strengthen it to **MUST** or weaken it to **MAY**.

Derived and Query requirements must have the following characteristics:

- **Cohesive:** the requirement addresses one and only one thing.
- **Complete:** the requirement is fully stated with no missing information.
- **Unique:** the requirement is not (in whole or part) repeated in other requirements or referenced artefacts.
- **Consistent:** the requirement does not contradict any other requirement or referenced artefact.
- **Correct:** the requirement meets all or part of a need identified during analysis.
- **Unambiguous:** the requirement is concisely stated in language recognisable to stakeholders. It is subject to only one interpretation. Adjectives and adverbs are to be avoided. Compound statements are prohibited.
- **Verifiable:** the implementation of the requirement can be verified by inspection, analysis, demonstration, or test.

Informatic concepts contain a listing of Informatic concepts to be found in each of the Derived Requirements (or Expressed Requirements where there has been no further need for further elaboration). An Informatic Concept is a word or a phrase expressed in the stakeholders natural language that describes an area or subject that may be important from an informatics perspective. A single derived requirement may contain one or more Informatics Concepts. Similarly a single common Informatic Concept may be derived from many Derived Requirements.

The Analyst may capture this requirements information and construct the Requirements Catalogue in the users' work environment or in a workshop, but in all cases it is expected that this artefact is thoroughly reviewed by users.

See also Construction guidelines with Requirements Catalogue product description §17.3.

9.5.2 EA Construction

The element structure described in §9.5.1.1 is created within the Project Browser.

When requirements traceability diagrams are required, these are created by dragging the requirements and their 'realising' elements onto the diagram (and generating the various associations as appropriate).

If display of requirement notes is required – this involves setting the properties of requirements diagrams such that notes in elements are displayed (see §13.4.6).

Where elements realise requirements, the element should be dragged onto the requirements diagram (as a simple link) and a realisation association should be made from the element to the requirement.

Where requirements are derived from elements, the element should be dragged onto the requirements diagram (as a simple link) and a directional trace association should be made from the requirement to the element and the stereotype should be set to 'derived from'.

To accommodate stakeholders who require a document-centric view of requirements a Rich Text Format (RTF) version of the Requirements Catalogue can be generated. RTF output is generated from the model (see §13.4.11) and incorporated back into the model (see §13.4.30).

A CSV output detailing the requirements for a model can also be generated (see §13.4.31). NB: CSV outputs are only able to export the items available to the requirements element within the properties dialogue. It is advised therefore that analysts use an RTF output if they need to create an output showing this information for an element.

9.5.2.1 Model Search

Enterprise architect has a detail model search that can be used to populate an RTF report. Documentation on this can be found in the tool help files. To support this functionality an SQL script has been developed (See §16). This script allows the analyst to identify the 'parents' and 'children' of any chosen requirement. For example, from a derived requirement you can identify the parent Expressed Requirement(s) and spawned Informatic Concept(s).

9.5.2.2 Requirement Deprecation

The following outlines the process for deprecating a requirement:

- The requirement element's status within the requirements properties dialogue must be amended to 'Deprecated'
- The deprecated element must be moved from the original package) to one entitled 'Deprecated Requirements' within the 'Requirements Catalogue' package.

- The deprecated element's name as defined in the properties dialogue should be amended to add the prefix 'Deprecated:' at the beginning of its title.
 - A brief description of the reason for its deprecation and details of any other requirements that replace it or to which its parameters have been subsumed should be entered into the properties dialogue General Notes section
 - Where a requirement is deprecated any links or associations with other requirements should not be removed (removal from the diagram does not remove links to other elements) – this supports reinstatement of the requirement if it is needed at a later date as well as helping to understand its original context.

9.5.3 Requirements Catalogue Quality Review

See §17.3

10 Use Cases

10.1 Introduction

Use cases describe the functional aspects of the area / domain under investigation.

The following sections provide definitions and notations for use cases and identify the types of use case and supporting use case artefacts produced.

10.2 Use Case Definitions

10.2.1 Use Case

Use cases fulfil requirements by describing a discrete unit of interaction between a user (human or machine) and the system. This includes the steps of the process, any decisions points and alternate paths included within the process; and the process participants (Actors).

10.2.2 Actor

An Actor is a participant in a process/ function described by a use case. The actor may be a person or a system.

10.2.3 Actors Catalogue

The Actors Catalogue is used as a container for all the Actors used within a Use Case Model and any subsequent specific Use Cases.

10.2.4 Actor Hierarchy

Where relationships exist between actors, this is represented by the actor hierarchy. The Actor Hierarchy is shown on the Use Case Model (see §10.4.2.)

10.2.5 Use Case Catalogue

A Use Case Catalogue is used as a container for Use Cases. This includes candidate Use Cases which are either not progressed (and deprecated or deleted) or are constructed as described in §10.4.3

10.3 Use Case Notation

Generic notation applicable to use cases is described in this section; notation which is specific to individual use case types is described in §10.4.

10.3.1 Core Notation

Use cases are represented on use case diagrams as ellipses.

Actors are represented on use case diagrams as 'stick people'

The lines between Actors / Use Cases, Use Cases / Use Cases and Actors / Actors are associations. They show an interaction between the Actor / Use Case or Use Case / Use Case and a relationship between the Actors. These associations can be constrained.

A Boundary may be included on the diagram. The boundary encloses the use cases which will form the 'system' that will be developed. Note that system in this context

may refer to a business system, an automated (computer) system, or a combination of multiple instances of each/both types.

Figure 12 is an example use case diagram:

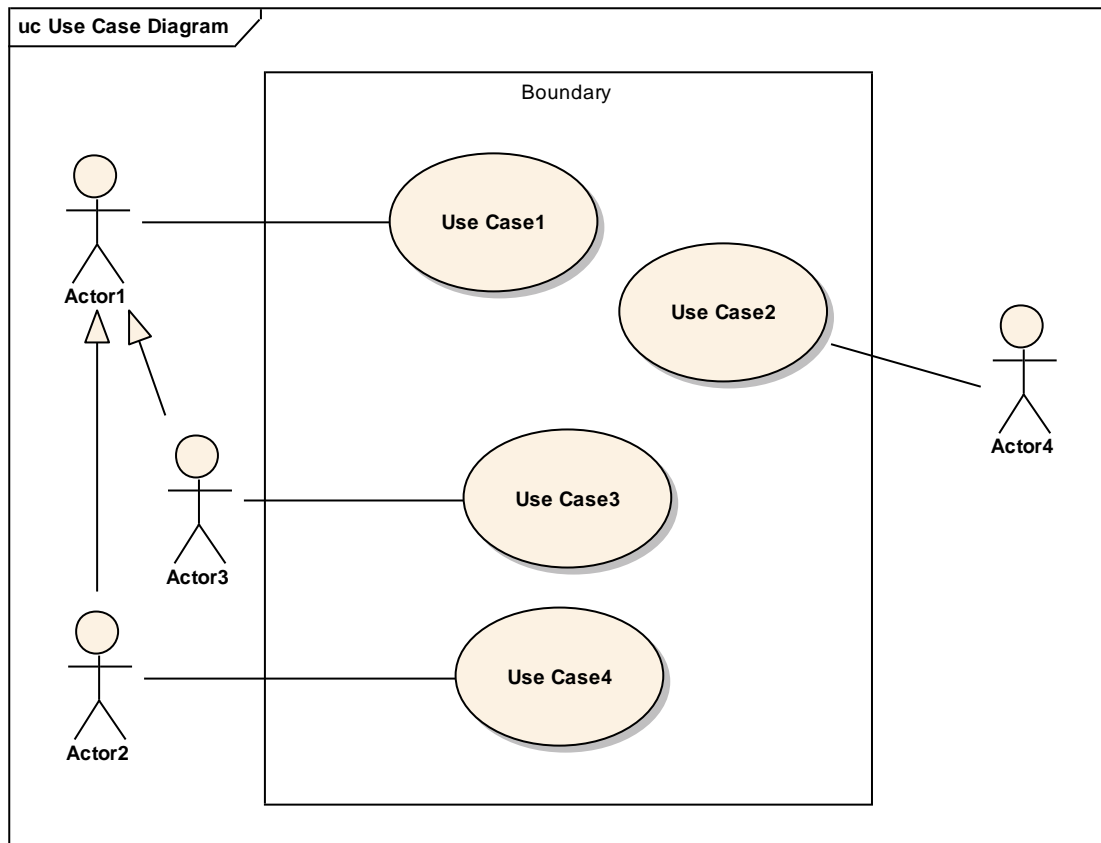


Figure 12: use case diagram

10.3.1.1 Use Case <<Includes>> Association

An include association is a relationship denoting the inclusion of the behaviour described by a Use Case within another Use Case.

In Figure 13, Use Case 1 is the including use case (i.e. the base use case.) It calls out to the included use case (Use Case 2.) This flow is mandatory, i.e. the base use case must call the included use case.

The control must return to the base use case when execution of the included use case's process is complete. The base use case must handle all end point conditions identified in the included use case.

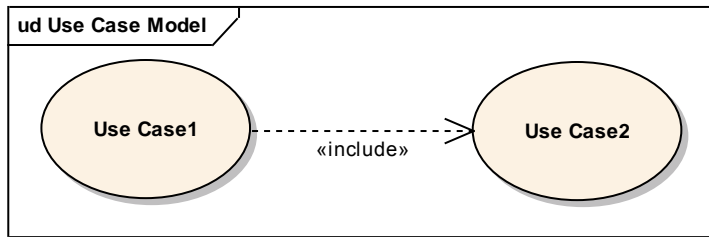


Figure 13: include use case example

10.3.1.2 Use Case <<Extends>> Association

An extend association is a generalisation association, where an extending use case continues the behaviour of a calling use case, at a specific point within the path of the calling use case.

In Figure 14 Use Case 4 is the extended or calling use case. Use Case 3 is the extending use case. Thus Use Case 3 'extends' Use Case 4, providing some additional, optional processing. The conditions under which this piece of processing will be triggered should be stated (potentially as a business rule see §8.2.3.)

The control must return to the calling use case when execution of the extending use case is complete. The calling use case must handle all end point conditions identified in the called use case (Use Case 3).

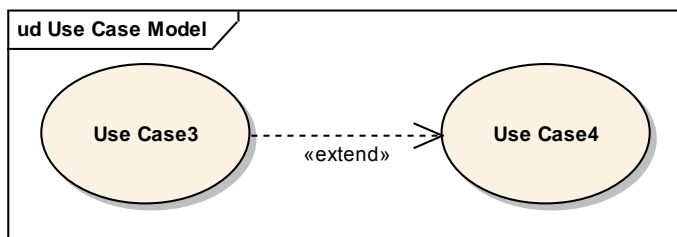


Figure 14: extends use case example

10.3.1.3 Associations

The relationship between a use case and an actor is represented by an association line connecting the use case and actor (see Figure 15). The association may have an open-headed arrowhead on one end of the line indicating the direction of the initial invocation of the relationship. The association indicates that the actor plays some role in performing the use case.

In the example shown in Figure 15, Actor 1 is associated with both Use Case 1 and Use Case 2, Actor 2 is only associated with Use Case 1.

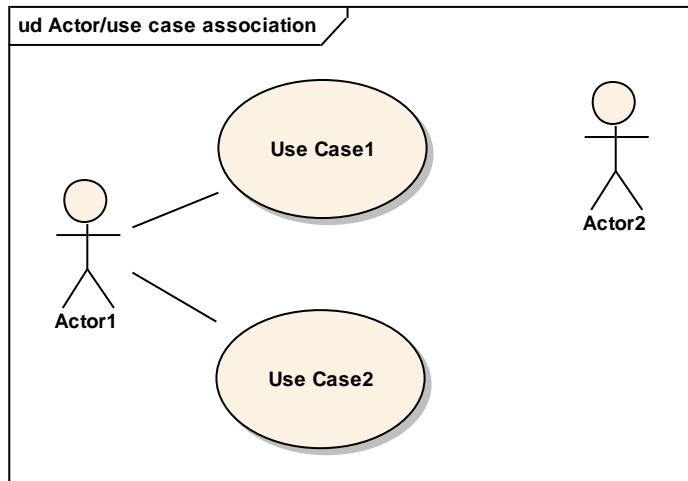


Figure 15: actor and use case association

10.3.1.4 Generalisation

Within a Use Case, generalisation relationships may exist between use cases and also between actors.

10.3.1.4.1 Generalisation of Use Cases

When a use case describes a variation on another use case, this can be described with a generalisation relationship between the use cases, see Figure 16.

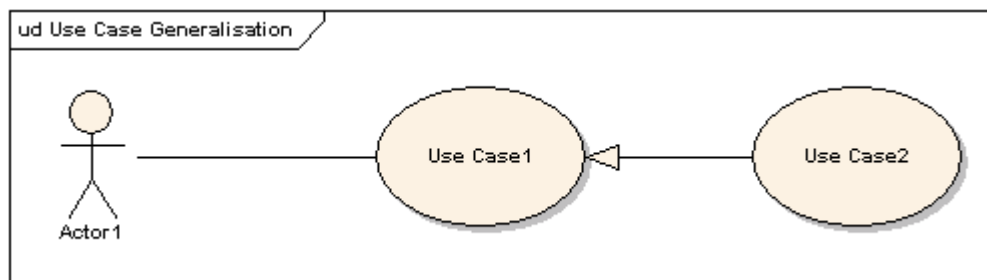


Figure 16: use case generalisation

Use cases that specialise another use case should only specify an alternative or exceptional situation to the general use case and the goals of the use cases should be the same. In Figure 16 the specialised use case (Use Case2) describes a situation in which the generalised use case (Use Case1) is not performed - the behaviour of the specialised use case replaces the behaviour of the generalised use case.

10.3.1.4.2 Generalisation of Actors

Generalisation relationships may exist between actors within a use case; this creates a hierarchy of actors. This is typically applied in one of the following situations:

- An abstract user role exists where responsibilities are shared across multiple concrete user roles. In Figure 17, Actor1 represents an abstract role which can be undertaken by Actor2 or Actor3.

- A user role is a special case of another user role

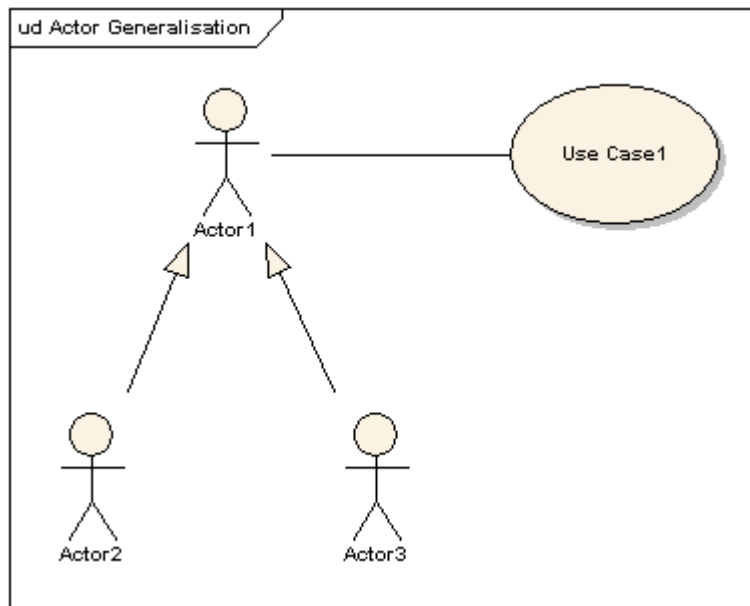


Figure 17: actor generalisation

The generalisation relationship is the only relationship allowed between actors.

10.3.1.5 Realisation by Use Cases

Use cases are often defined to show how requirements (in particular) functional requirements are 'realised'. This is described in §9.3.1.2.

10.3.1.6 Basic and Alternate Paths

Each use case describes a process in terms of a flow of events. These are documented as paths containing steps. Each step is a single discrete activity that is carried out to fulfil the purpose of the use case.

The basic path describes the route taken in the majority of cases. Thus, whenever this process is invoked these are the steps that will be enacted most of the time.

The alternate paths represent deviations from the basic path. They are triggered at points within the basic flow and should be shown as 'decisions' taken according to a defined logic. Alternate Flows should return control to the basic flow unless they cause an exception in which case the use case should end.

10.3.1.7 Trigger Events

A use case starts when it receives some form of stimulus – a trigger. The trigger is some event which causes the process described in the use case to begin.

A use case will normally have only one trigger event. If more than one trigger event is identified then care is required and several questions must be addressed (examples of which are shown here):

- Do all the trigger events need to have occurred before the use case can start?
- Does each trigger lead to exactly the same flows?

- Is there an order in which the trigger events need to occur?

A trigger may be identified as the result of an invocation of a business rule. Trigger events are to be described textually.

10.3.1.8 Pre-Conditions

A pre-condition is something which must have happened prior to this use case starting. The use case cannot start until the (set of) pre-conditions are true. Pre-conditions are to be described textually.

10.3.1.9 Post-Condition

Post conditions exist in two forms; a successful post-condition is an outcome of the use case when the processing runs as planned, a guaranteed post-condition is an outcome of the use case when the processing encounters an error. Post-conditions are to be described textually.

10.3.1.10 Goal

The goal describes the aim of the use case – when the processing completes successfully the goal must have been achieved. Goals are to be described textually.

10.3.1.11 Description

A textual description of the use case.

10.3.2 Toolset

Use Cases shall be created by the preferred CASE tool (Enterprise Architect) see §13 and maintained within the native file of this tool or an SVN repository if this is appropriate and output as with other CASE tool artefacts see §6.

10.3.2.1 Hints and Tips

Enterprise Architect (EA) provides a number of relevant features some of which are identified below:

- To make an association directional, select the Association Properties/ General tab and change the Direction field as appropriate.
- The Goal, Trigger, Pre-conditions and Post-conditions are entered under the use case Properties/constraints section. Select the appropriate 'Type' from the available list and enter the appropriate information.

10.3.3 Notation Adoption

10.3.3.1 Generalisation

An actor or use case may be a generalisation of another actor or use case. The generalised actor or use case contains the common properties, and the specialised actor or use case expands upon these, identifying additional properties that it possesses beyond those inherited from the generalised one. (N.B. Properties which are inherited are not restated in the specialised entity).

10.3.3.2 Use Case Activity Diagrams

Use Case Activity Diagrams are not a formal part of the Phase 1 LRA methodology. This is primarily because it is not the intention of the methodology to describe system functionality. However, the Analyst may find activity diagrams useful tool to validate the scope of a Use Case, in this case see referenced document 2.

10.3.3.3 System Boundaries

A Boundary may be included within a use case diagram to represent the 'system' (see §10.3.1). It is possible (although unconventional) for multiple boundaries to be displayed within a use case diagram to represent multiple systems.

10.4 Use Case Types

Analysis models typically contain numerous individual use cases and actors. The relationships amongst and between use cases and actors are represented within a use case model. The individual use cases and actors are managed within Catalogues to facilitate reuse.

10.4.1 Use Case Catalogue

10.4.1.1 Introduction

The Use Case Catalogue is produced in parallel with the Use Case Model and iterated through the life-cycle of the project. It is a grouping of Use Cases within Use Case Models.

The Use Case Catalogue is created as part of the **LOGICAL ANALYSIS PHASE**

The Use Case Catalogue has no graphical representation and is simply a directory in the model hierarchy. The Use Case Catalogue is part of the Use Case Model (see §10.4.2)

10.4.1.2 General Construction

The Use Case Catalogue is constructed using the following principles:

- Package structure within the Use Case Catalogue is usually based on the structure of the Use Case Model.
- The Use Case Catalogue must contain all known Use Cases.
- The selection and use of packages defines the organising principle of the use cases.
- The Use Case Catalogue does not describe associations between or within packages.

10.4.1.3 EA Construction

Within EA the Use Case Catalogue exists as a package held in the Logical Analysis package.

10.4.2 Use Case Model

10.4.2.1 Introduction

One or more Use Case Models are identified from the **LOGICAL ANALYSIS PHASE** artefacts. A single Use Case Model is a standard UML artefact which provides an organised view of the behaviours for a subset of the domain, and a possible point of entry to analysis/design artefacts. The collection of Use Case Models provides an organised view of the behaviours for the whole domain.

The Use Case Model is created during the **LOGICAL ANALYSIS PHASE**. The Analyst may capture this information and construct the Use Case Model in the users' work

environment or in a workshop, but in all cases it is expected that this artefact is thoroughly reviewed by users.

10.4.2.2 General Construction

The Use Case Model provides a way of organising the Use Cases. It creates a grouping mechanism for the Use Cases.

A Use Case Model is derived from **DOMAIN ANALYSIS & SCOPING PHASE** artefacts, notably the Requirements Catalogue (see §9.5) and the Stakeholder Business Model (see §12.3.1). It consists of a use case diagram and Use Cases, Actors and associations between them; a system boundary. Specific characteristics include:

- There may be multiple Use Case Models
- The Use Cases within a Use Case Model should all be written to a similar level of detail and using domain vocabulary. There should be no hierarchy of use cases within the model.
- All actors (including specialisations) identified in the Use Cases must be represented within the Use Case Model (it is insufficient to infer the existence of a specialised actor from a generalised actor) and held within a package (the Actors Catalogue) within the Use Case Model.
- An actor does not need to be human; it can also be an external system. Where an actor is a system avoid referencing particular systems unless they are design constraints. For example PDS. [N.B where a system exists that is not a design constraint it should be referred to in logical generic terms for example 'ordering system'].
- A domain/system boundary enclosing the Use Cases in the Model is recommended.

10.4.2.3 EA Construction

In EA each Use Case Model is created within a discrete package within the project browser structure (see § 6.2.3).

Actors from the Actors Catalogue are placed onto the Use Case Model diagram using drag and drop from the project browser.

Use Cases represented within the Use Case Model will have a hyperlink to the related corresponding Use Case Activity Diagram.

10.4.3 Use Case

10.4.3.1 Introduction

A Use Case describes how the information about how the organisation will function – it is a detailed description of the processes the organisation will carry out to produce an end result.

The Use Case comprises three parts – the diagram, the textual elements.

10.4.3.2 General Construction

A Use Case should include the elements listed in Table 5

Element	Description
---------	-------------

Use Case Name & Identifier.	Brief name and unique identifier.
Description	This should introduce the Use Case to the reader, placing it in context with the rest of the system and/ or domain and describing it and its purpose at a very high level. Any information necessary to understand subsequent sections of Use Case should be detailed in this section.
Actors	All actors referenced within a Use Case should be identified.
Trigger	A description of the event(s) that initiates the Use Case should be provided.
Goal	An explanation of what the Use Case is trying to achieve
Pre-Conditions	All conditions that must be satisfied if the Use Case is to be initiated
Post Conditions	Post conditions may be considered in two forms; Success and Guaranteed. The Success conditions show the state of the system after the process has run as expected. The Guaranteed condition shows the state we can expect the system to be in after an unsuccessful condition.

Table 5: use case elements

Use Cases may also include Business Rules to be observed within the Use Case (see §8.2.3).

Use Cases are constructed using the following principles:

- Use Cases are physically located within the Use Case Catalogue (see §10.4.1)
- Avoid referencing particular systems unless they are design constraints, for example PDS. [N.B where a system exists that is not a design constraint it should be referred to in logical generic terms for example 'ordering system']
- All actors (including specialisations) identified in the Use Case must be represented within the Use Case Model and are stored within the Actors Catalogue
- All paths must lead to an identified post condition.
- Pre and post-conditions identified in the Use Case must not conflict with process as stated in the BPM.
- Domain specific information concepts must be represented within the corresponding IAM.

10.4.3.3 EA Construction

The creation of an association between an actor and a Use Case on the Use Case Model will automatically populate the Actor section within the Use Case description.

10.4.4 Business Use Case

See §8.2.4

11 Class Models

11.1 Introduction

The purpose of a class model is to articulate structural characteristics, including:

- entities
- information requirements within entities
- relationships between entities
- methods available for the manipulation of information managed by entities

Class models allow this information to be ordered logically and can be used to identify common pieces of data that are used by different processes helping ensure its consistent application and update.

Class models, as other artefacts in the analysis toolkit, are developed over time and as the analysis progresses so the understanding of the data becomes more detailed allowing finer grained pieces of data to be identified, documented and organised.

The following sections first describe definitions and notations of the class models and then discuss the types of class models produced.

11.2 Class Model Definitions

11.2.1 Class

The purpose of a class is to specify a collection of related pieces of data. A class has a name, description, attributes, and relationships with other classes. Classes also have methods or operations which define how data managed by the class can be manipulated

11.2.1.1 *Abstract Class*

An abstract class follows the description given above, in addition these classes are never instantiated - abstract classes are designed only as parent classes from which child classes may be derived (see §11.3.3.2). This means that all artefacts using class diagram techniques may include abstract classes, but any object model must not include objects instantiated directly from an abstract class.

11.2.1.2 *Association Class*

An association class is a model element that has both association and class properties. An association class can be seen as an association that also has class properties, or as a class that also has association properties

11.2.2 Association

An association specifies a relationship that can occur between two entities. There are a range of possible relationships that can exist, which have different meanings and notation. Most commonly, an association will simply have a name, direction and cardinality. Where a greater degree of constraint is placed upon an association, three commonly used types exist:

11.2.2.1 Aggregation

Aggregation is a special form of association that specifies a whole-part relationship between the aggregate (whole) and a component part.

11.2.2.2 Composition

Composite aggregation is a strong form of aggregation that requires a part instance be included in only one composite at a time. If a composite is deleted, all of its parts are normally deleted with it.

11.2.2.3 Inheritance

Where one class (A) has an inheritance relationship with another class (B), Class A has all the attributes and relationships of Class B, plus any attributes and relationships of its own. In this case A is the inheriting class, and is known as the sub-class or *specialised* class; B is the super-class or *generalised* class.

11.2.3 Attribute

A piece of data managed within a class, typically the attribute will be a characteristic of the class.

11.2.3.1 Datatype

Attributes are datatyped, such that their values must conform to a classification definition. For example an attribute 'time' is likely to be datatyped such that completing the time attribute with '10.45' may be valid but 'orange' is invalid. This mechanism reduces scope for the incorrect use of attribute values.

Where an attribute has no value this is represented as 'null' which is distinct to a value of zero.

11.2.3.2 Example Data

These are example instances of attributes. Example data is used to help describe attributes whose meaning is otherwise unclear. The set of example data within a model need not represent a consistent/coherent set.

11.2.3.3 Default Values

The initial value allocated to the attribute when instantiated within an object is its default. This may be explicitly overwritten by another value unless identified as being 'constant'. See §11.3.2.1.

11.2.3.4 Coded Data

The value of an attribute may be constrained to being populated by one of a discrete collection of coded values, for example from a specific vocabulary. This approach requires that the set of values (i.e. the codeset/vocabulary/subset) is identified. A further requirement is to associate a value with a coded concept – a 'code value pair'. For example the coded concept may be 'Estimated Due Date' and the value to be associated may be '23/04/2007'.

11.2.4 Operations

Operations are features of a Class or other element that represent the behaviour or services an element supports. For a Customer Class, "UpdateCustomerName" and "GetCustomerAddress" can be operations. In this methodology Operations are

invoked as a behaviour call from Call Operation Actions located in the Domain Query Functionality Activity Diagram.

11.2.5 Constraints / Dependencies

Constraints or dependencies may exist between:

- Classes
- A class and attribute(s) within the class
- A class and attribute(s) within a different class
- Attributes in the same class
- Attributes in different classes

Constraints and dependencies should be written in everyday readable English.

All constraints recorded must have:

- A short, meaningful title
- Narrative defining the constraint
- Constraint type

Where constraints pertain to another element of the model that element must be referenced:

- Reference another attribute using the full and exact attribute name
- Reference another class using the alias of the class (N.B in EA the alias is the unique identifier generated from auto numbering)
- Reference an attribute in another class using the alias of the class and the full and exact name of the attribute.

11.3 Class Model Notation

This methodology uses UML class model notation; entities are identified, attributes and operations related to those entities are captured and relationships between entities are shown.

11.3.1 Core Notation

Class diagrams are composed from classes and associations, as follows:

11.3.1.1 Classes

The notation used for classes is a rectangle split into two parts; the top part contains the name of the class, and the lower part contains any attributes which make up the class.

Examples of classes and their notation are shown in Figure 18: class notation.

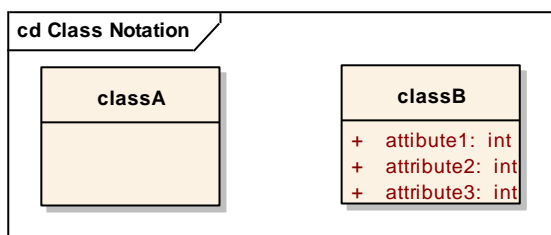


Figure 18: class notation**11.3.1.1.1 Description**

A class should have a brief description clearly explaining the purpose and nature of the class. This should promote a consistent understanding of the class by all users of the model.

11.3.1.1.2 Attributes

Attributes are individual items of information which have appropriate names, datatypes, cardinality and constraints.

11.3.1.1.2.1 Description

In most cases it will be appropriate to record a short description of an attribute. This should promote a consistent understanding of the attribute by all users of the model.

11.3.1.1.2.2 Datatypes

Datatypes must be recorded for attributes. They are shown after the attribute name, preceded by a colon. For example:

exampleAttributeName:Integer

If a complex type is not used from the datatype hierarchy this must be fully represented within every class diagram that it appears (see §11.3.3.3).

N.B where a constraint on a datatype exists (such as identification of a vocabulary) this will be held as a constraint on the attribute.

11.3.1.1.2.3 Default Values

Where an attribute has a default value (see §11.2.3.3), this is shown after the attribute name, preceded by an 'equals' sign (=). For example:

exampleAttributeName:Datatype = 1234

11.3.1.1.2.4 Attribute Hiding

An attribute can be classified as being Public (+), Private (-), Protected (#) or Package (~). The symbol denoting this classification is shown in front of the attribute name. For example:

+ exampleAttributeName:Datatype = 1234

In all analysis models, attributes must be classified as Public.

11.3.1.1.2.5 Coded Data

The most common clinical terminology used by this methodology is SnCT (SNOMED CT), background information is available in the form of a SnCT User Guide (see Related Document 7) and a SnCT Abstract Models and Representational Forms (see Related Document 8).

For Phase 1 LRA, the objective of the LRA analysis is to identify information requirements and present them in a form that can be modelled by DS&P terminologists and the Data Dictionary Team. Consequently, the identification of a set of values (i.e. the codeset/vocabulary/subset) is not required. However it is important that Terminologists are given sufficient supporting information to help them

understand the nature of the requirement. Therefore the datatype given to the attribute should indicate whether the attribute should be considered by the terminologist.

CardinalityThe cardinality of an attribute is shown in brackets after the datatype. For example: **exampleAttributeName: Datatype [0..1]**

11.3.1.1.2.6 Constraints / Dependencies

See §11.2.5

11.3.1.1.3 Operations / Methods

UML classes may have operations or methods associated with them. Within this methodology, they are used to add context to an attribute's usage. This essentially validates the usefulness of the information requirement when called from a query (from a call action) and supplies the context of usage by indicating where it was used.

For example,

Imagine a class named "BloodPressure" that has an attribute "Diastolic". In this form the information requirement is perhaps too generic. However if you apply the following operations to the attribute it becomes the information requirement becomes more apparent:

getPostHDSessionDiastolicBloodPressure()

getPreHDSessionDiastolicBloodPressure()

The operations show that this is a renal haemodialysis diastolic blood pressure that is recorded before and after a patient's haemodialysis session.

In this methodology, operations are only required where an Action element has a need to call or retrieve an attribute in the course of executing a query. The Action elements calling the operations can be viewed in the Domain Query Functionality Activity Diagram. An operation has to be defined before an Action element can call it. For an operation only following needs to be detailed:

11.3.1.1.3.1 Name

A name that describes the context of use of the operation and the attribute it operates with. Typically the operation name will start with the word 'get' to show that the intention of the operation to retrieve some information. The following words should describe the context of use which are followed in turn by the attribute name.

For example:

getContextAttribute()

11.3.1.1.3.2 Notes

In most cases it will be appropriate to record a short description of an operation. This should promote a consistent understanding of the operation by all users of the model

11.3.1.1.3.3 Return Type

This should be set to void

11.3.1.1.3.4 Scope

This should be set to public

11.3.1.1.4 Constraints / Dependencies

See §11.2.5

11.3.1.1.5 Class Types

11.3.1.1.5.1 Abstract Class

Abstract classes (see §11.2.1.1) are represented in UML by displaying the class name in italics.

Abstract classes are typically used for the following purposes:

- Representation of abstract characteristics (attributes and operations) which are inherited/extended by child classes. This is applied where common characteristics are shared between multiple classes and allows these characteristics to be defined in a single (abstract) class but instantiated from many (child) classes.
- Some forms of conditionality can be described using abstract classes. For example, the cardinality of an abstract class can be used to constrain the instantiation of child class. In Figure 19, when Class1 is instantiated, either Class2, Class3 or Class4 must also be instantiated. This also deals with the simple conditional case of if Class2 is instantiated then Class3 and Class4 cannot be instantiated.

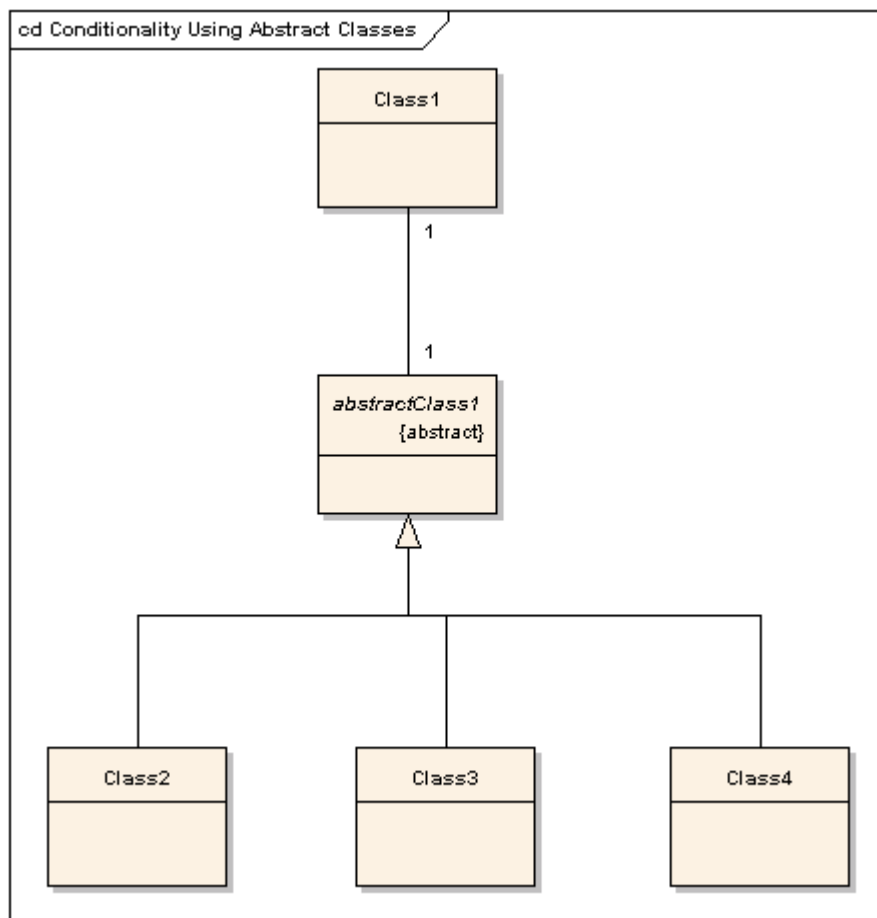


Figure 19: conditionality using an abstract class

11.3.1.1.5.2 Association Classes

An association class can only be instantiated if both of the classes it is associated to are being instantiated. It contains some data about that association. It offers us a clearer definition of complex associations than a simple textual constraint.

For example, in Figure 20, class3 is an association class. It allows us to show the following textual constraint diagrammatically:

If instances of class1 and class2 exist, the attribute `associationClassAttribute` from class3 will be returned (as part of an instance of class3):

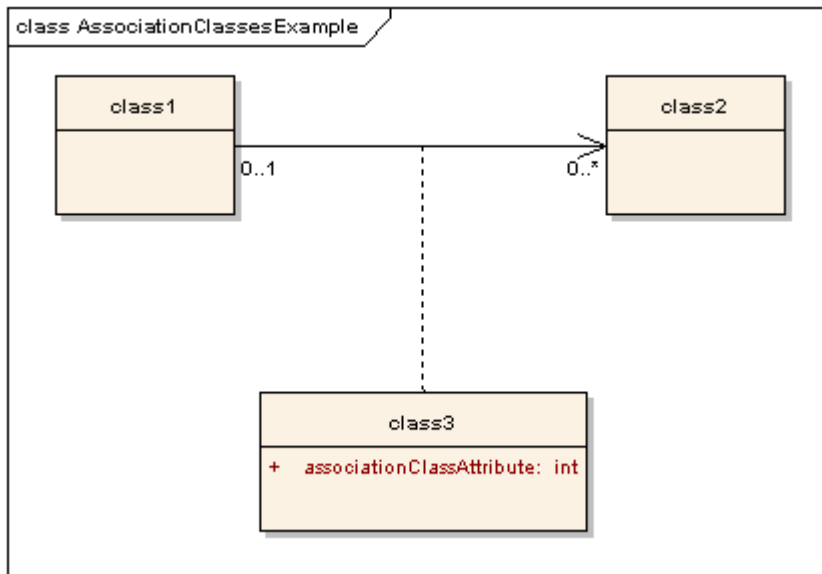


Figure 20: association class example

However, association classes should be used with caution:

- To aid understanding, a textual description of the constraint should also be expressed for clarity

11.3.1.2 Associations

Associations represent a relationship between two classes. A range of possible associations between two classes exist. These are:

1. Association – The least constrained type of relationship between two classes
2. Directed Association – Gives context for the cardinalities on the association (i.e. these only apply in the direction identified)
3. Inheritance – Generalisation/ specialisation – 'is a kind of'. Avoids duplication of information within the model (the attributes of the generalised class are inherited by the specialised class), and creates a hierarchy of related classes.
4. Aggregation – Whole/ part relationship – 'is a part of'. Parts can be moved from one whole to another, and may exist without the whole or be shared between wholes.

5. Composition – A more strict form of aggregation. Parts can only belong to one whole, cannot be moved from one whole to another and are destroyed when the whole is destroyed.

The different notations for these associations are shown in [Figure 21](#). Associations may have appropriate names, direction and cardinality.

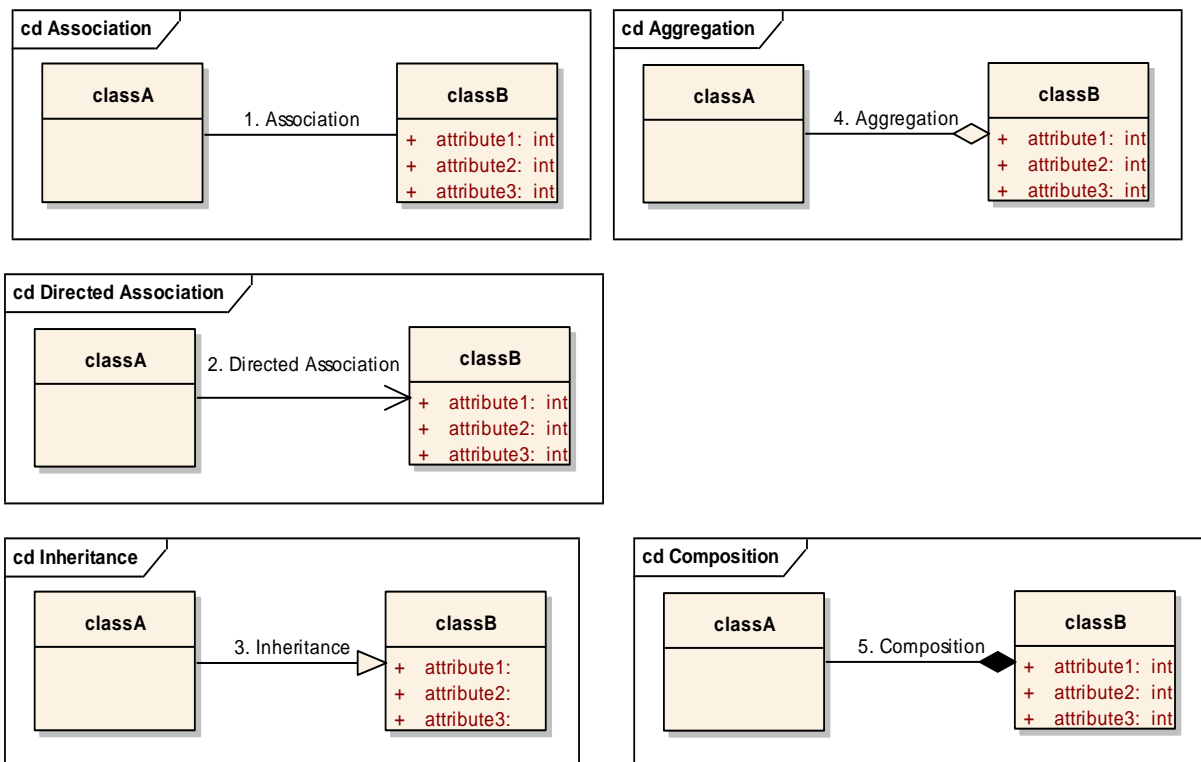


Figure 21: association notation

11.3.1.2.1 Cardinality

The cardinality of an association between two classes is shown by placing the relative number of possible occurrences at each end of the association, as shown in Figure 22. This figure shows that Class A has any number (including zero) of Class Bs associated with it; and that Class B can have one and only one Class A's associated.

N.B where an inheritance relationship exists between two classes, the concept of cardinality is invalid and hence not represented.

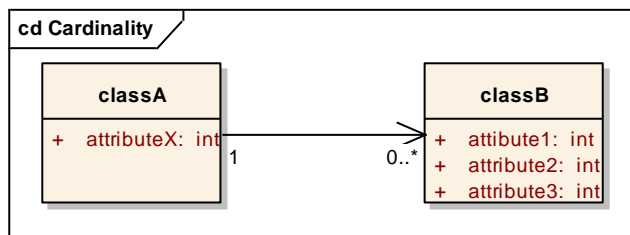


Figure 22: association cardinality

11.3.1.2.2 Constraints / Dependencies

Neither constraints nor dependencies should be recorded against associations. These should be recorded against classes or individual attributes as appropriate (see Figure 24).

11.3.2 Toolset

Class models shall be created by the preferred CASE tool (Enterprise Architect) see §13 and maintained within the native file of this tool and output as with other CASE tool artefacts see §6.

11.3.2.1 Hints and Tips

Enterprise Architect (EA) provides a number of relevant features some of which are identified below:

- The default attribute cardinality is (1..1), where this cardinality applies, it is not displayed within the class. The absence of visible cardinality on an attribute should be interpreted as (1..1).
- It is possible to record constraints against associations. The use of this functionality in this methodology is prohibited.
- It is possible to display the constraints recorded against classes on the class diagram. The use of this functionality in this methodology is prohibited.
- To re-use an attribute or operation which already exists, drag from the project browser onto the class which requires it. This creates a separate copy of the attribute which is not linked to the original.
- The short description of the class should be captured in the class properties dialogue box/General tab/ Notes field.
- Datatypes should be captured in the class properties dialogue box/Detail tab/ Attributes/ Type field.
- Default values (see §11.2.3.3) should be captured in the class properties dialogue box/Detail tab/ Attributes/ Initial field. Where an attribute is to be set and remain unchanged - use 'const' check box with initial value field. Note this will mark the attribute as 'READ ONLY' on the diagram (see Figure 23).

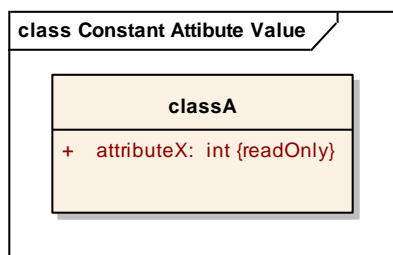


Figure 23: constant attribute value

- Where required, example data for attributes should be captured within the attribute properties dialogue box/constraints tab as a constraint type of 'Example'.
- Attributes are created by default as 'Private', this must be changed to public in each case to 'Public' in the class properties dialogue box/Detail tab/ Attributes/ Scope field.
- To clearly identify abstract classes: Tools/ Options select Objects, select 'Highlight {abstract} elements'.
- Constraints on attributes and classes are entered in slightly different ways. Both must have title (1) text (2) and Type (3), see Figure 24.

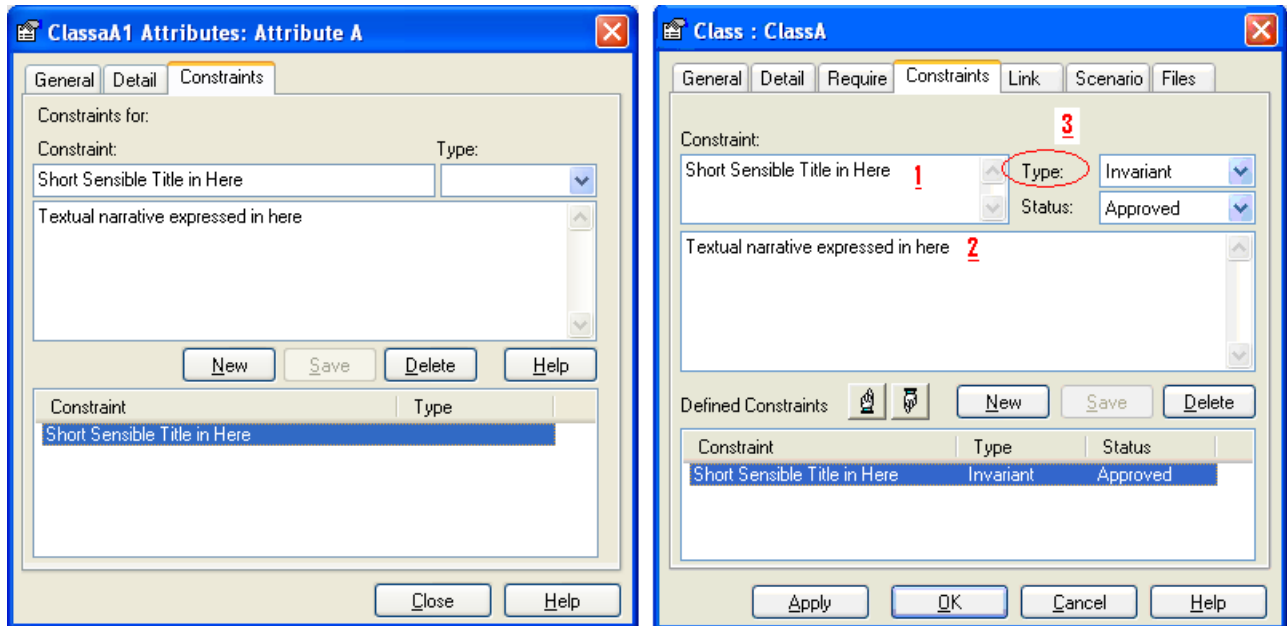


Figure 24: recording constraints

11.3.3 Notation Adoption

11.3.3.1 Naming Convention

All classes and attributes/operations should be named using camel case convention, e.g. investigationType

It expected that class and attribute/operations names will be nouns from the business area vocabulary.

11.3.3.2 Inheritance

The appearance of attributes/operations in classes linked by generalisation/specialisation relationships are as follows:

- Where attribute/operations are identically applicable (appear with identical cardinality, constraints, defaults etc) to all the specialised classes from a generalised class these attribute/operations must be displayed within the generalised class (see Attribute1 and Attribute2 in Figure 25).
- Where attribute/operations do not appear in all specialised classes or appear with differing cardinality/constraints/defaults, these attribute/operations must be displayed within the specialised class (see Attribute3 and Attribute 4 in Figure 25).

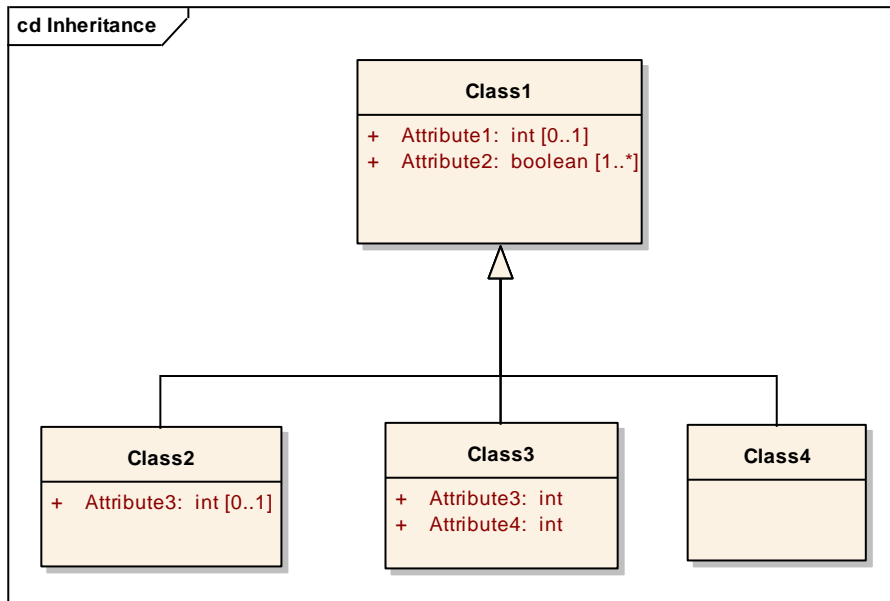


Figure 25: inheritance/display of attributes across a generalisation

11.3.3.3 Datatypes

All attributes must have specified data types.

11.3.3.4 Stereotyping Attributes and Operations

The typical representation of attributes within a class does not include any classification or organisation of the attributes or operations. In most cases this is sufficient and no precedence or classification of the attributes within a class is needed. However, in some circumstance the Analyst may wish to distinguish between categories of attribute operation (for example read-only and read-write attributes; or clinical and non-clinical operations). Where it is beneficial to apply such a grouping mechanism to attributes or operations within classes, this can be achieved by the application of stereotypes which leads to attributes or operations in classes being categorised as shown in Figure 26.

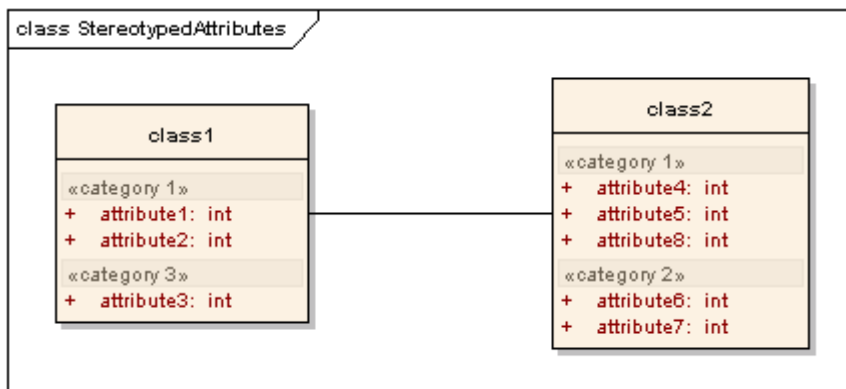


Figure 26: stereotyped attribute example

Stereotyping of attributes or operations should be used with care and the following should be noted:

- stereotyping only provides a single level of classification since only one stereotype can be applied to each attribute or operation
- Stereotyping must not be applied to express (or duplicate) a constraint.

The creation of stereotypes in EA is described in §13.4.37, the allocation of stereotypes simply involves the selection of the appropriate stereotype as part of the attribute or operation definition.

11.3.3.5 Attribute Validation

Attribute definitions can be used to validate attribute values (e.g. for input validation etc.) An example is given in §11.3.1.1.2.2 where the datatype of an attribute is specified this can be used to validate attribute values. Attribute characteristics (which can be used for validation) described elsewhere in this document include:

- constant values: see §11.3.2.1
- cardinality: see §0

Where attribute validation rules are to be defined and the above attribute characteristics are insufficient to articulate the rule, this will be articulated as attribute constraints (see §11.3.1.1.2.6). Standard validation rules are defined in Table 6 and can be used together to provide fine-grained validation rules.

Validation Constraint	Description	Type	Notes
Character Set (<i>name</i>)	See name	set	The name of the character set must be supplied e.g. UTF8
Alphabetic	See name	set	At the time of writing, no equivalent datatype exists in the hierarchy
Numeric	See name	set	At the time of writing, no equivalent datatype exists in the hierarchy
Alphanumeric	See name	set	At the time of writing, no equivalent datatype exists in the hierarchy
Lower Case Alphabetic	See name	set	At the time of writing, no equivalent datatype exists in the hierarchy
Illegal	Characters and sets	filter	

Validation Constraint	Description which are prohibited	Type	Notes
Extend to Include	Extension to a set	filter	
Pattern	An allowed data pattern	filter	Typically articulated as a Regular Expression
Space	Character space	character	
Consecutive Spaces	See name	characters	
Newline	New line character	character	
Overlap (target)	Allows/prohibits overlap with linked (target) values	dependency	
Existence (target)	Constraint invoked by the existence of a linked (target) entity	dependency	
Value (target)	Constraint invoked by the value of a linked (target) entity	dependency	
Update (update constraint)	Constraint on attribute 'updatability'	behaviour	'Remove', 'Add', 'Update' and their combinations used to describe constraint

Table 6: attribute validation

11.4 Class Model Types

There are four class model types, each described in detailed below. The order in which they appear below describes the progression of information concepts as analysis is undertaken. To ensure a consistent set of models, no new information concept may be introduced that cannot be traced back to the previous level of hierarchy. Specifically, the relationships between classes moving down though the hierarchy is 1..*, and tracing up though the hierarchy is 1..1.

11.4.1 Stakeholder Information View (SIV)

11.4.1.1 Introduction

The Stakeholder Information View is created as part of the **DOMAIN ANALYSIS AND SCOPING PHASE**. It is intended to be a basic overview of the information concepts and their relationships with the business area or domain. This can be represented diagrammatically or can also directly re-use existing stakeholder artefacts if they exist and specifically is used for:

- Identification of the business area(s) and system(s) with which the domain is concerned
- Identification of classes (information concepts) that are shared with other domains
- Identification of classes (information concepts) that are the source and destination of communications between business area(s) and between system(s)

The SIV is not intended to be a definitive document, so will not capture the detailed information requirement and their associated flows; however it is intended that the content should be understandable and recognisable to business users and describe at a high level the whole scope of the domain.

11.4.1.2 General Construction

The SIV shows:

Named classes – without attributes or operations, but with a brief description and an alias (see §8.2.1).

Named and directed associations – without cardinalities.

When the SIV is first created, the focus of attention may be on a particular part of the domain. This may lead to the initial model being better developed in that area than for the remainder of the domain. Subsequent work may shift the focus to a different area of the domain and lead to further development of that area of the model.

The SIV should be created using business terminology, capturing concepts that are important and recognisable to business users. This means that no constraints are applied to the granularity of SIV classes.

Where a class may ultimately represent a system, avoid referencing particular systems; i.e. use logical generic terms for example 'orderingService'.

Actors identified within the SIV should be represented as in the use case model. Representation as a generalised class is acceptable. The SIV may also show additional actors which are not represented elsewhere.

The Stakeholder Information View is expected to be published, and at the point of publication it should be correct and consistent with the rest of the artefact set.

11.4.1.3 EA Construction

The SIV classes should all be contained within the SIV package of the Domain Scoping and Analysis section.

All Actors represented on the SIV, which are contained in the Actors Catalogue, must be referenced using the Relationship Matrix (see §13.4.27).

11.4.2 Information Analysis Model (IAM)

11.4.2.1 Introduction

The Information Analysis Model (IAM) provides the domain information context for subsequent analysis and/ or development. It allows the exploration of the full information requirements for the area of the domain under analysis.

The IAM is created as part of the **LOGICAL ANALYSIS PHASE**. It is a refinement of (some of) the concepts captured in the SIV and an elaboration of the information requirements. All of the information required to fulfil the business processes captured in the Use Case Models (see §10.4.2) should be defined in the IAM, therefore these artefacts should be produced in parallel.

11.4.2.2 General Construction

The IAM shows:

- Named classes
 - Class description
 - Alias – unique identifier (see §8.2.1)
 - Attributes
 - Datatypes where determined
 - Cardinality
 - Constraints
 - Usually a brief description
 - Operations
 - Meaningful name
 - Usually a brief description
- Associations
 - Must have Cardinality
 - Direction should normally be shown
 - May have Name (where appropriate)
 - Must not have Constraints.

The Information Analysis Model uses the Stakeholder Information View as input. A class in the SIV that is within the area of interest must be represented within the IAM in some way (for example a SIV class being represented by multiple classes or

composite or generalised classes in the IAM); equally, every IAM class must be derived from a single class in the SIV.

Data concepts within the Use Case Model are candidates for representation within the IAM as a class or attribute.

Informatic Concepts identified in the Requirements Catalogue are candidates for representation within the IAM as a class or attribute

Action calls within the Domain Query Functionality Activity Diagrams may indicate candidates for representation within the IAM as an attribute

Depending upon the scope of the project, it may be necessary for multiple IAMs to be developed within a single project; however multiple IAMs may exist within a domain, which focus on separate areas of interest. Where there is overlap between the areas of interest, the separate models should be consolidated. Classes should be shared where possible between models where the concept is common to multiple models.

The use of a foreign key (i.e. identifier used to extract information from external systems such as PDS and SDS) in place of the full information requirement of a class is prohibited. For example, where full demographic information is the requirement, this shall not be substituted with NHS number; this is to validate any future-referenced system functionality fulfils the information requirements.

It is not expected that any systems will appear as classes in the IAM.

Where association lines cross on a diagram, the lines must be perpendicular.

Multiple associations between two classes are prohibited.

Only associations relevant to the area of interest need to be modelled.

11.4.2.3 EA Construction

- No movement of classes from IAM to Domain Information Description (DID, see §11.4.4) Package.
- Can create an independent copy of a class from IAM into DID (this requires that the new class is localised using 'local copy', the simple link mechanism should not be used.
- Every class in the IAM must be linked to a single SIV class using the Relationship Matrix (see §13.4.27).

11.4.3 LRA Analysis Class Models

NOTE: The following guidance is a generic form of the approach used for Release 3 15 Appendix 3) of the LRA project. At this point in time, the extent to which this approach will be adopted for Phase 1 will depend upon the outcome of discussions between the DS&P Analysis Team and the DS&P Data Modelling team. While it is likely the core principles will remain intact there is the possibility of significant changes to the process and artefacts once a way of working has been established.

11.4.3.1 Introduction

Following the production of the Information Analysis Model (IAM), as part of the Logical Analysis for a particular domain, the Logical Record Architecture analysis needs to be undertaken.

This analysis culminates in the production of the following artefacts:

- Domain Information Descriptions (DID)
- Data Element Definitions Palette comprising of
 - Candidate Data Elements
 - Completed Data Elements Definitions

During the review of each IAM there will be data items which may have already been reviewed and modelled by Terminologists and the Data Modelling Team but, in the initial stages, the majority will be new data items. These new data elements are Candidate Data Elements which will be assessed and, if appropriate, modelled by terminologists and the data modelling team

So the purpose of the LRA analysis is to produce a complete, coherent, traceable set of information requirements from an IAM, and ultimately a domain, which are linked to existing, defined items or identified as candidate items. As the candidate items are modelled by the terminologists and Data modelling team, the Candidate Data Elements identified during the LRA Analysis Phase model must be updated to correctly reference their ultimate representation (Completed Data Elements).

11.4.4 Domain Information Description

This section contains general guidelines regarding the construction of Domain Information Description classes and elements and specific guidelines in respect of the creation of the DID Package and DID Diagram.

11.4.4.1 Introduction

The DID, its related content and Candidate and Complete Data Elements are created during the **LRA ANALYSIS PHASE**.

The DID Package contains all the classes and their attributes within a model which are shown on the DID Diagram. The order of creation of DID and Candidate Data Elements is left to the discretion of the Analyst thus the development process is iterative and interdependent with all DID classes being derived from the IAM.

The DID Diagram shows all the information elements contained within the DID Package.

Candidate Data Elements are the main source for the development of the physical information requirements. It is expected that there will be a high level of consistency and traceability between a Candidate Data Element and corresponding physical information requirement. A Candidate Data Element is created to specify a granular piece of information content that can be defined and classified by Terminologists and the Data Modelling Team. The DID provides the context for the information requirement to support its definition. The Candidate Data Elements are contained within a separate Data Definitions Palette but are referenced as an Attribute Type from within the DID Class.

11.4.4.2 General Guidelines

- Classes
 - A class in a Domain Information Description must be derived from at least one class in the IAM

- A DID class will focus upon a core area of interest. It draws on the concepts initially identified in the SIV and subsequent derivations identified in the IAM. It is likely therefore that a class identified in the SIV or specialisations of the SIV class identified in the IAM will become the primary focus of a DID.
- Classes that are immediately associated with the primary class should then be considered for inclusion into the DID to ensure the DID sufficiently covers the scope of the area of interest. It is likely that the associated classes will be used in multiple DIDs, however this should not be regarded as duplicated effort as this enables the terminologists to establish the scope and usage of an information requirement that helps facilitate its definition.
 - The analyst should study the SIV and IAM to identify the core concept classes and the classes that have been subsequently derived from these concepts.
- A class's attributes in the IAM that represent information to be modelled in a DID must be represented attributes within the DID class
- The associated class attributes must also be represented within the DID class
 - Classes should be defined with:
 - Class name
 - Class description
 - Alias – unique identifier (see §8.2.1)
 - Constraints
 - Attributes; defined with
 - Datatypes
 - All attributes must have specified datatypes explicitly defined and represented as a model-specific complex datatype. (see §11.3.1.1.2.2)
 - Cardinality (see §0)
 - Constraints (see §11.2.5)
 - Description should be the same as the equivalent IAM attribute description
 - Where appropriate an example (as part of the description)
- Operations
 - IAM class operations do not need to be represented in the DID class although the analyst use them to help them provide supporting context for a DID
- Development Constraints
 - DID class models are likely to be used as the input to the development of physical information standards. It is possible that such standards are constrained by a specific formalism and this constraint can be extended to the DID class models. At this point in time for Phase 1 of LRA the particular constraints have yet to be determined. However the approach used for LRA Release 3 will form the basis of discussion and so have been included in 15 3 for reference.

- **Class Packages**

DID Class Packages containing related classes can be applied by the Analyst to group and manage the classes according to an organising principle; this is usually to make a large model more manageable:

- DID class models can represent content using classes only
 - All Class Packages must be stored within the DID Package and be present on the DID Diagram (where one is created).
 - Class Packages should represent a single business/information concept.
 - Each package:
- Should include a brief description as a general note and where appropriate further annotations (see §8.4.1)

11.4.4.3 EA Construction

It is expected that all classes and packages will have a description (see §11.4.4.2), this description will be recorded as a general note within the element properties dialogue (see §8.4.1).

11.4.4.4 Domain Information Description (DID) Package

The DID Package adheres to the general guidelines for the LRA Analysis class models (see §11.4.4.2). Additional guidelines for DID Package constructions include:

- The DID Package is a folder within the model hierarchy

11.4.4.4.1 EA (IAM/DID Package Relationship)

- No movement of classes should take place from the IAM to the DID Package.
- Every class in the DID Package must be linked to one or more IAM class using the Relationship Matrix (see §13.4.27).
- It is possible to create an independent copy of a class from IAM into DID Package (this requires that the new class is localised using 'local copy', the simple link mechanism should not be used).

11.4.4.5 Domain Information Description (DID) Diagram

The DID Diagram represents all the classes and class packages contained within the DID Package using a UML class diagram. The DID Diagram adheres to all the general guidelines for the LRA Analysis class models (see §11.4.4.2). Additional guidelines for DID Diagram constructions include:

- A DID Diagram is always required when there are multiple DIDs.
- Although it is unlikely, where there is only one DID within a model then a DID Diagram is not required; in this instance the individual DID takes the place of a DID Diagram.
- The DID Diagram should cover all the related Information Requirements within the scope of the model's Domain Query Functionality (see §12.3.4).
- Within the DID Diagram:
 - The DID Diagram is constructed of classes / class packages from the DID Package and which are present in one or more of the related DIDs.

11.4.4.5.1 EA Construction

- DID Diagrams must only be created within the DID Package
- If a DID Diagram is required (see §11.4.4.5) it should only use classes / class packages that have been created in the DID Package

11.4.5 Candidate Data Element

A Candidate Data Element (CDED) models a single information unit and is usually used as the basis for the development of defined and categorised information standards. CDEDs are visually represented using a UML class diagram. CDEDs adhere to all the general guidelines for the LRA Analysis class models (see §11.4.4.2). Additional guidelines for Data Element Definitions Palette Package constructions include:

- A CDED is required for each new information requirement identified by a DID attribute in the DID Class.
 - The CDED class will have the same description as the corresponding attribute in the DID from which it has been derived
 - A CDED class will consist of a single attribute of “value”

The CDED class attribute datatype will be constrained to the datatypes that have been created in the LRA Reference Models using ISO 21090 data types (see NHS Logical Record Architecture Data Types Abstract Specification EA Construction

- CDEDs should only be created within the Data Element Definitions Palette Package.
- NOTE: At this point in time for Phase 1 of LRA the particular method of CDED construction have yet to be determined. However the approach and construction methods used for LRA Release 3 will form the basis of discussion and so has been included in Appendix 3: Methodology for the Identification and Maintenance of Renal Candidate Data Elements for reference.

11.5 Class Model Type Summary

	Area	SIV	IAM	DID Package	DID Diagram	CDED
Class	Naming	camelCase	camelCase	camelCase	camelCase	camelCase
	Description	Essential	Essential	Essential	Essential	Essential
	Unique ID (Alias)	From AutoCounter	From AutoCounter	From AutoCounter	From AutoCounter	From AutoCounter
	Constraints	Prohibited		Not Expected	Not Expected	Not Expected
	Contained in Packages			Where Appropriate	Where Appropriate	Where Appropriate
Attributes	Naming	No Attributes	camelCase	camelCase	camelCase	camelCase
	Constraints		Where Determined	Where Determined	Where Determined	Where Determined
	Datatypes		Where Determined	Essential	Essential:	Essential:
	Cardinality		Essential	Essential	Essential	Essential
	Description		Expected	Expected	Expected	Expected
	Foreign Key Only		Prohibited	Expected	Expected	Expected
	Example Value		Not Expected	Not Expected	Not Expected	Not Expected
	Default Value		Not Expected	Not Expected	Not Expected	Not Expected
Operations	Naming	No Operations	Where Determined	Not Expected	Not Expected	Not Expected
	Description		Where Determined	Not Expected	Not Expected	Not Expected

Associations	Naming	Expected	Where Appropriate	Not Expected	Not Expected	Prohibited
	Direction	Expected	Expected	Where Appropriate	Where Appropriate	Prohibited
	Cardinality	Prohibited	Essential	Essential	Essential	Prohibited
	Constraints	Prohibited	Prohibited	Prohibited	Prohibited	Prohibited

11.6 Class Model Examples

See example diagrams §18.1, §18.4, §18.8 & §18.9

11.7 Class Model Quality Review

See product descriptions (§17.1(SIV), §17.8 (IAM), §17.10 (DID) & §17.11 (CDED))

11.8 Class Model Issues

11.8.1 Traceability

There is a maintenance overhead involved in creating multiple versions of the same class within different analysis artefacts; for example a change of class name must be performed in a variety of places. However – this must be the case for the following reasons:

- A SIV does not contain attributes
- An IAM must contain full business information requirement
- A DID and Candidate Data Element may contain information requirement specific information, and must show only the information which will be presented as a requirement

To preserve these separate artefact requirements, each artefact package must ‘own’ all of its constituent classes – no automated links between classes in separate artefacts. As this is the case – a manual method must be employed to show relationships and traceability through the entire artefact set. In EA the Relationship Matrix functionality (see §13.4.27) will be used to capture the relationships between classes in different artefacts.

12 Activity Diagrams

12.1 Introduction

Activity Diagrams are a UML diagramming technique used to analyse aspects of dynamic behaviour. They describe the control and object flows between business activities undertaken by different actors, including any decision logic. The format and nature of activity diagrams, particularly their similarity to flow charts, is seen as providing the best combination of fitness for purpose (amongst the diversity of potential users), familiarity and consistency with existing approaches.

12.2 Activity Diagram Definitions and Notation

12.2.1 Core Notation

Activity diagrams are composed from activities, actions and flows, as follows:

12.2.1.1 Activities

Activities represent behaviour following a received stimulus (flow). The role may be a user or a system. Examples of activities and their notation are shown Figure 27.

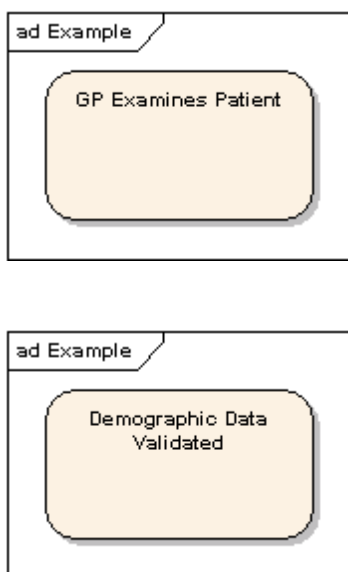


Figure 27: simple activities

An activity can have the following incoming flows (see §12.2.1.7). Multiple incoming flows should either:

- be 'joined' if all incoming flows need to occur prior to the following activity (see §12.2.1.10); or
- 'Merged' if only 1 incoming flow needs to occur prior to the following activity (see §12.2.1.11). This joining/merging is optional for final or flow final activities (see §12.2.1.4).
- Alternately, where an activity has multiple entry parameter nodes, each node can behave as a connection point for an object flow.

Similarly, activities can only have single outgoing flows (which may be split/forked see §12.2.1.10)

12.2.1.2 Structured Activities

A Structured activity can be considered to be the parent to a child Activity diagram. Uses of structured activities include:

- To hide detail on otherwise cluttered diagrams.
- to logically group activities
- to aid navigation through large models by drill-down

The toolset-specific (see §12.2.2) notation for a structured activity is shown in Figure 28.

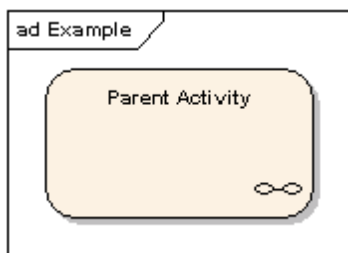


Figure 28: structured activity

12.2.1.3 Activity Parameter Nodes

An Activity Parameter Node is an object node that accepts input to an Activity or provides output from an Activity. It represents specific information passed to or from an activity. In this methodology its use is constrained to the Outline Query Artefact (see §12.3.3) and Domain Query Functionality (see §12.3.4) Activity Diagrams.

(Figure 29) depicts two entry parameters and one output parameter defined for the Activity.

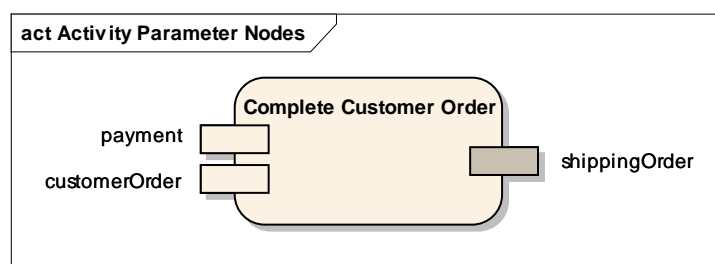


Figure 29: activity parameter nodes

12.2.1.4 Initial/Final Activities

Special cases of activities include the initial and final states represented within the diagram, as shown (with an example simple activity) in Figure 30.

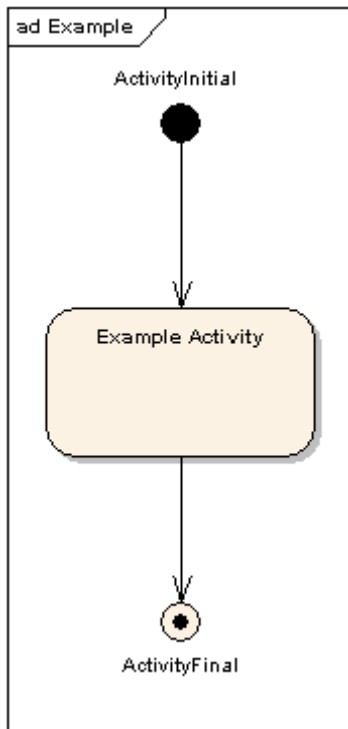


Figure 30: initial and final activities

The final state has two forms. The final activity (shown in Figure 30 and Figure 31) when 'fired' terminates all processing on the diagram and control is passed to the parent activity. The flow final (shown in Figure 31) when 'fired' terminates only the flow leading to the flow final i.e. other flows continue to execute.

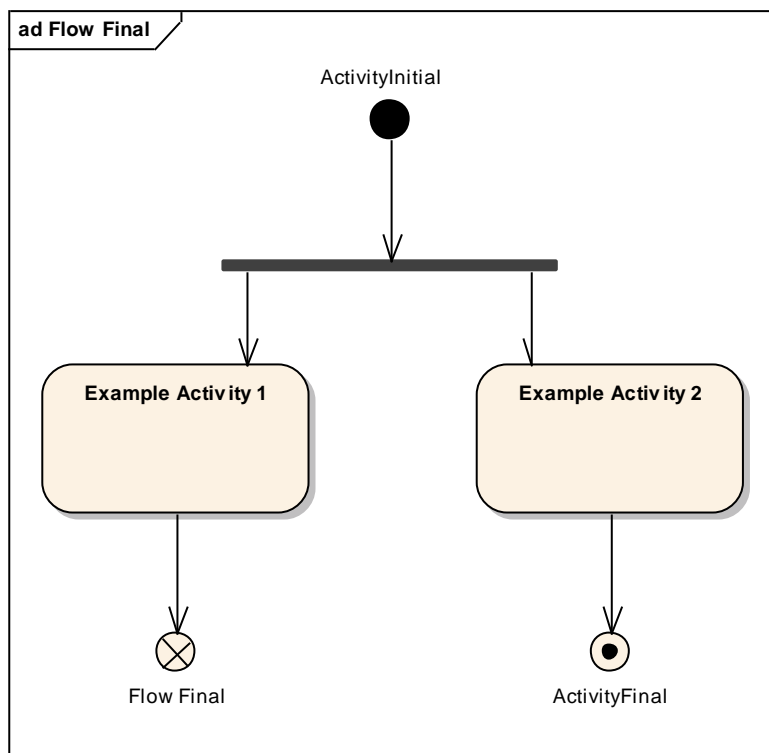


Figure 31 flow and final activities

12.2.1.5 Actions

An Action element describes a basic process or transformation that occurs within a system. It is the basic functional unit within an Activity diagram. Actions can be thought of as children of Activities. Both represent processes, but Activities can contain multiple steps or decomposable processes, each of which can be embodied in an Action. An Action cannot be further broken down or decomposed.

An Action can be further defined with pre-condition and post-condition notes (see §8.4.1), and certain properties can be graphically depicted on the Action. The data values passed out of and into an Action can be represented by Action Pins (see §12.2.1.5.1).

12.2.1.5.1 Action Pins

An Action Pin is used to define the data values passed out of and into an Action. An input pin provides values to the Action, whereas an output pin contains the results from that Action.

Action Pins are used (see Figure 32) to connect two Actions:

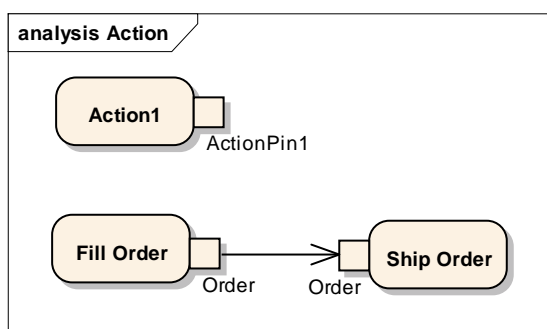


Figure 32: actions and action pins

12.2.1.5.2 Call Operation

Some properties can be graphically depicted on an Action element. For this methodology, the only role of action elements is within the context of a Domain Query Functionality Diagram (see §12.3.4). This artefact aims to demonstrate where and which information requirements are used to satisfy a query or identify the absence of an information requirement. This requirement would then need to be defined in order for the query's functionality to be satisfied.

Operations from Classes can be displayed on Activity Diagrams as Actions. When an operation is shown as an Action, the notation of the Action displays the name of the Class that features the operation, see Figure 33.

Note: the Action can only call a class operation that has already been detailed in the class.

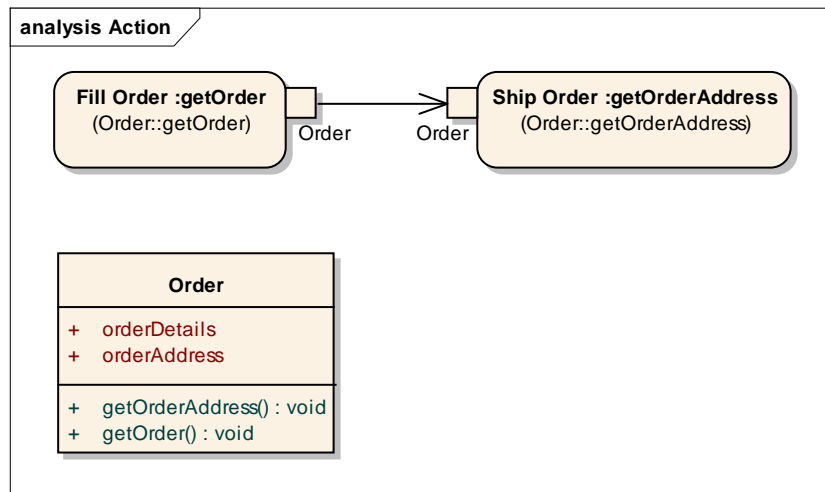


Figure 33: call operation action

Activities represent behaviour following a received stimulus (flow). The role may be a user or a system. Examples of activities and their notation are shown Figure 27.

12.2.1.6 Combining Activities, Actions and Call Operation Actions

In this methodology this approach is constrained solely for Domain Query Functionality Activity Diagrams. Actions can be viewed as the individual decomposed steps that fulfil the activity's purpose. Actions are therefore depicted within the activity element. Information is passed to and from Activity Parameter Nodes from the corresponding Action Pins. In the example (see Figure 34), the 'Complete Customer Order' activity requires 'payment' & 'customerOrder' information inputs; outputs 'shippingOrder' information; and comprises the actions 'Process Payment', 'Fill Order' & 'Ship Order'.

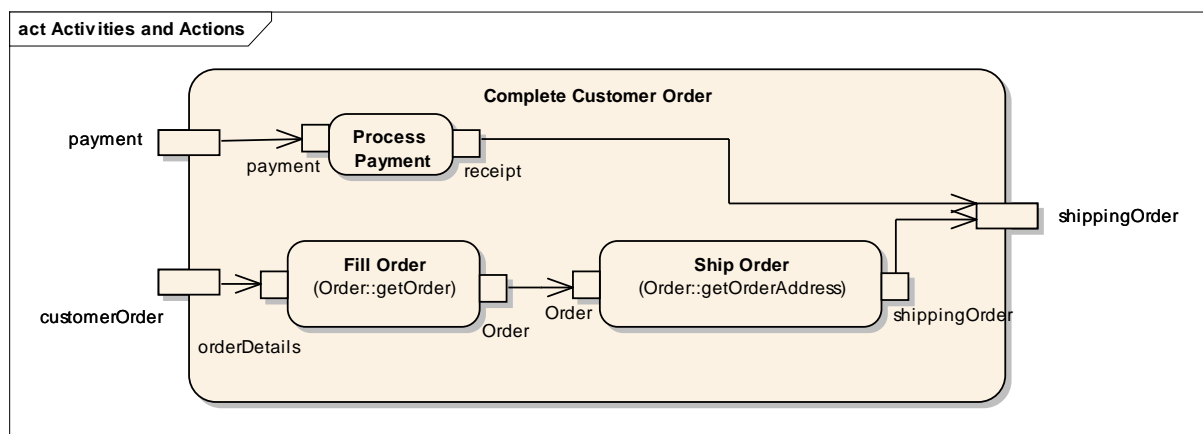


Figure 34: activities and actions

12.2.1.7 Flows

Flows represent the relationships between activities or actions – specifically their ordering and trigger dependencies. The basic flow is called a *control flow* which represents the order of activities such that once a source activity is completed a target activity is initiated as shown in Figure 35.

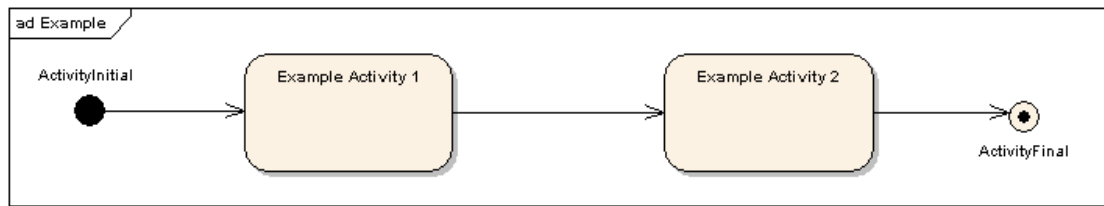


Figure 35: control flow

A special case of control flow is an object flow which identifies that an activity or action outputs an object (or has an input from an object) such as an information requirement. Object flows describe flow of control as well as the object being transmitted/received which is explicitly denoted as shown in Figure 36. Identified object flows are of particular importance in identifying information requirement related artefacts, notably Information Analysis Model. Object flows connect to Activity Parameter Nodes for activities (see §12.2.1.3) and Action Pins for actions (see §12.2.1.5.1)

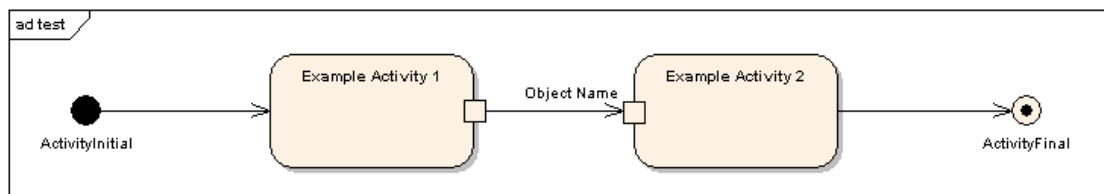


Figure 36: object flow

12.2.1.8 Decisions

Where multiple control flows are possible, decisions represent the selection of a single control flow. A single incoming control flow triggers the decision and the choice of outgoing flows is determined by the guard conditions defined for these flows which must be satisfied for the flow to be fired, as denoted in Figure 37. The combination of guard conditions from a single decision should represent the full set of possible conditions from the decision.

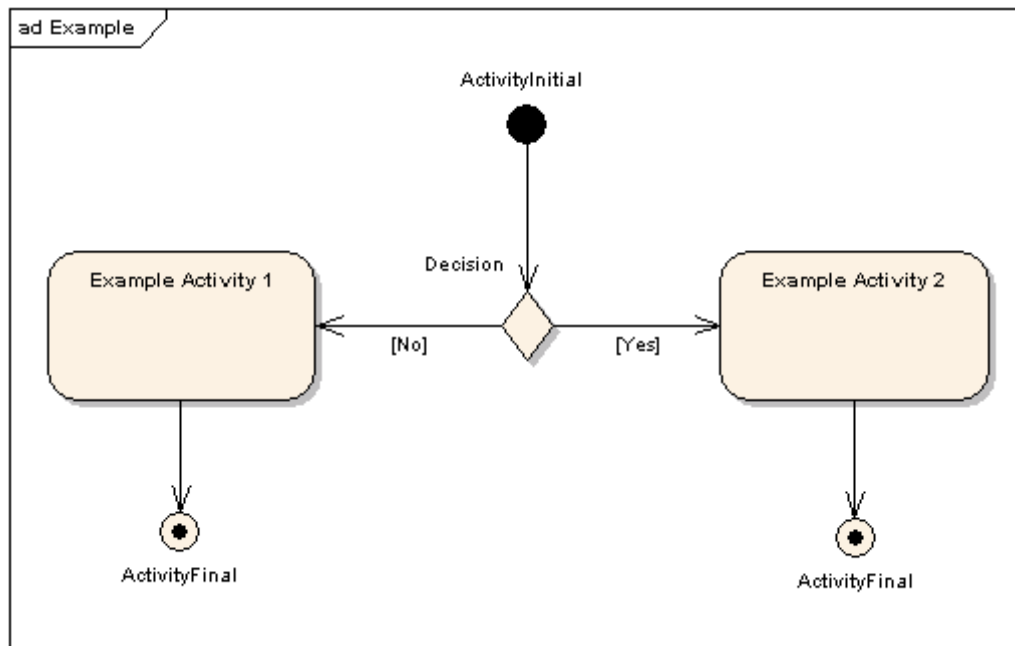


Figure 37: decision

A decision can have only a single incoming flow (see §12.2.1.7). Multiple incoming flows should be 'joined' if all incoming flows need to occur prior to the following decision (see §12.2.1.10); or 'merged' if only one incoming flow needs to occur prior to the following decision (see §12.2.1.11).

12.2.1.9 Events

Events can be applied within activity diagrams to represent occurrences which have no explicit trigger. Such events may be considered to be 'magic' – the event shall not have incoming flows. The only types of events to be applied for activity diagram purposes are:

- Receipt event which can occur in any context and which is not specific to an activity or a part of the process flow. This is represented by an event structure which triggers a flow, as shown in Figure 38;

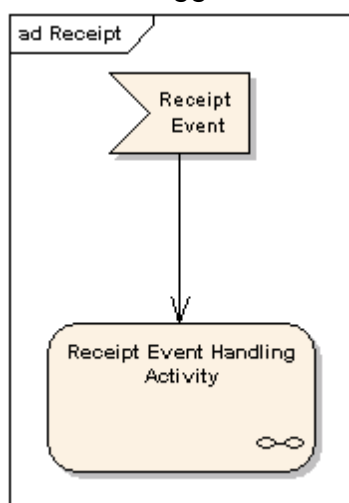


Figure 38: receipt event and exception activity

- Interrupt event where a predictable event may interrupt a specific activity or part of the 'normal' process. The association with the event or part of the process is represented by enclosing the interruptible events/process within a region along with the interrupt event. The interrupt handler is outside of this region and is connected to the interrupt event by an interrupt flow as shown in Figure 39. An example of an interrupt event is an un-received message resulting in a time-out exception handling interrupt.

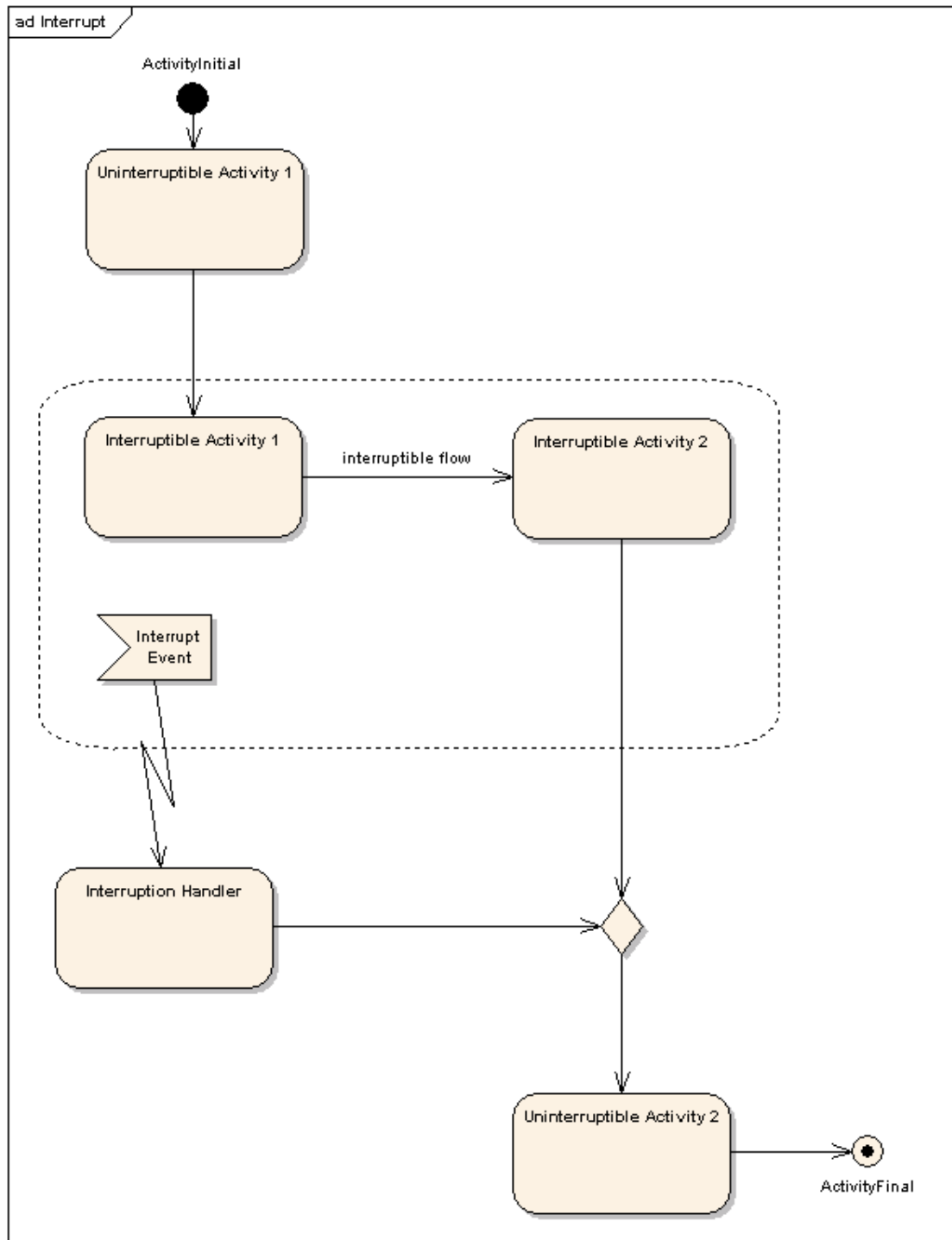


Figure 39: interrupt event associated with activity

12.2.1.10 Forks / Joins

Where flows occur concurrently this is represented with a fork such that an incoming flow triggers **all** outgoing flows. Multiple flows can be synchronised with a join such that the completion of **all** incoming flows trigger the single outgoing flow, as denoted in Figure 40.

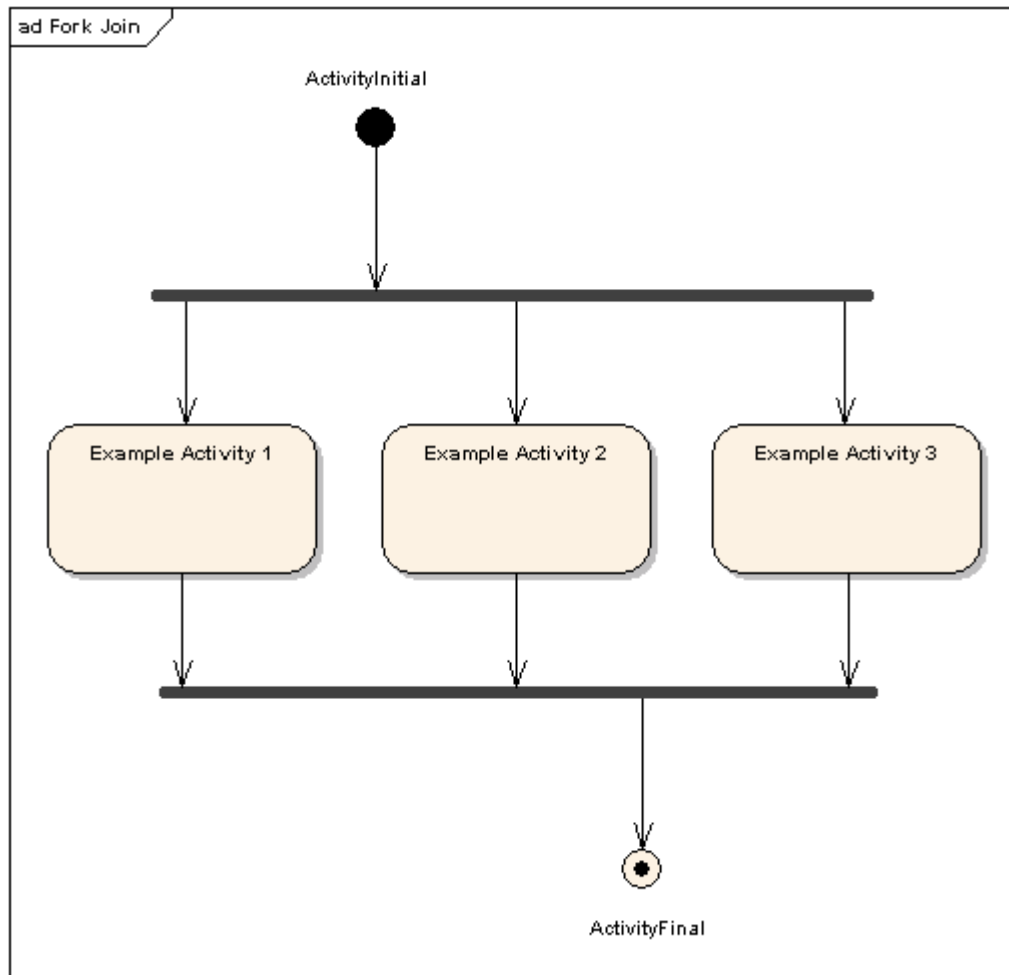


Figure 40: fork and join

12.2.1.11 Exclusive Or

Within a diagram there may be an activity which is reached by more than one route in certain circumstances but can only have one possible incoming flow in any given scenario, and to illustrate this, a merge decision box is used to combine the multiple possible flows into an exclusive or, as shown in Figure 41. Where an activity (see §12.2.1.1) or decision (see §12.2.1.8) has more than one incoming flow these should be 'merged' using this merge diamond (unless the incoming flows are to be joined (see §12.2.1.10)). An exception to this rule is the case of multiple flows into final activities or flow final activities (see §12.2.1.4), use of the merge diamond here is optional. The merge diamond is distinguished from a normal decision box as it is apparently un-named on the diagram. However to allow for efficient house-keeping, and identification of redundant elements, the merge box is identified via auto-

numbering (see §13.4.1) but the resultant label may either be hidden or, at the authors discretion, displayed as greyed out so as to downplay any significance to this ID.

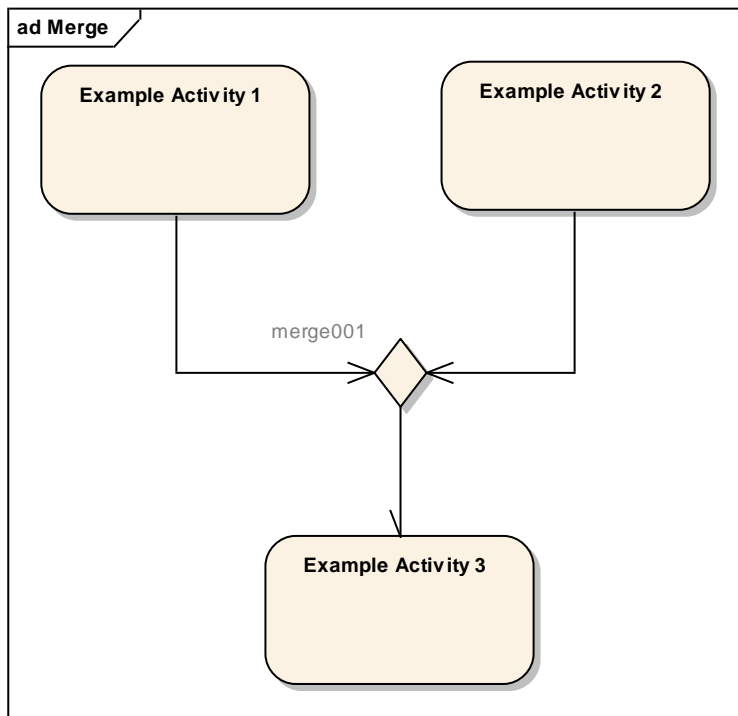


Figure 41: Exclusive Or (merge)

12.2.1.12 Swimlanes

Swimlanes are regions within an activity diagram which represent the role which has responsibility for activities within the region, as shown in Figure 42.

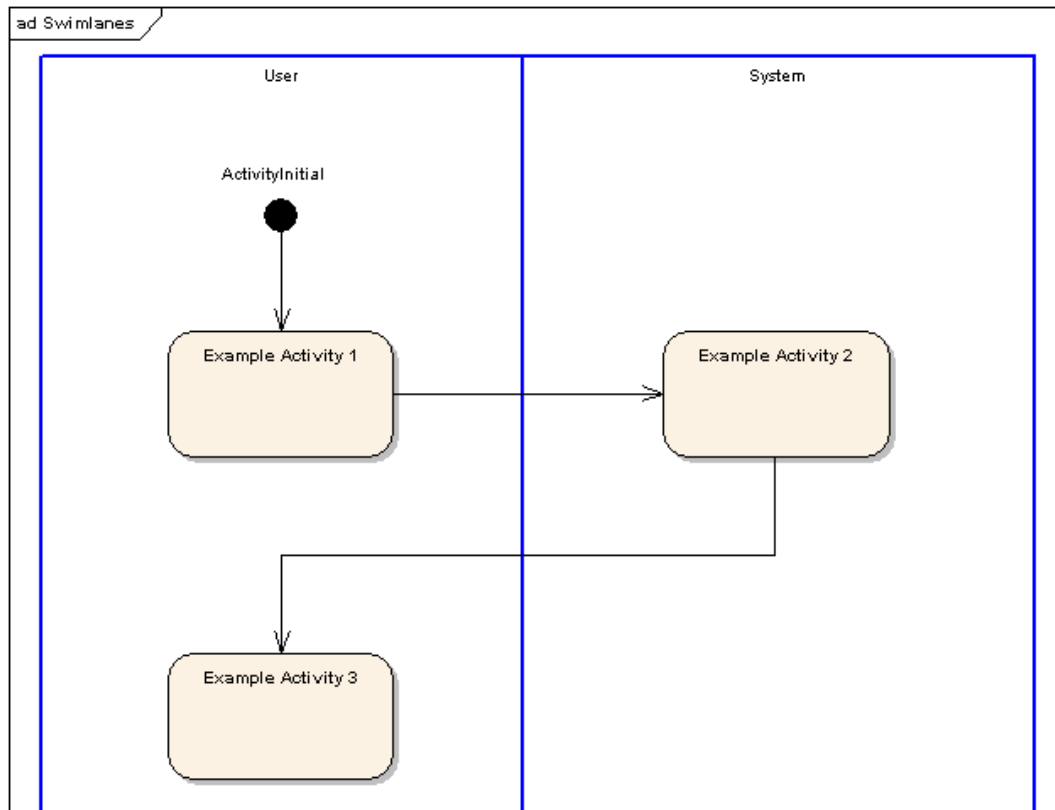


Figure 42: swimlanes

12.2.2 Toolset

Activity Diagrams shall be created by the preferred CASE tool (Enterprise Architect) see §13 and maintained within the native file of this tool and output as with other CASE tool artefacts see §6.

12.2.2.1 Hints and tips

Enterprise Architect (EA) provides a number of relevant features some of which are identified below:

- It is possible to set flow style to 'auto routing' but care should be taken to ensure that the meaning of the flow is not modified.
- If a structured activity name is amended the relevant child (diagram) name is not automatically amended and must be manually amended in the project view.
- The easiest way to re-name an activity is to double click the relevant activity which displays the properties box, but on a sub activity this action opens the diagram. To display the properties box of a sub activity, highlight it by clicking once and press Return
- The use of regions within an activity diagram can cause EA to prevent some legitimate actions to be undertaken. In particular the addition of (interrupt) flows within/across the region boundary may be identified as being in conflict with UML. This is a bug in Enterprise Architect and can be worked round by connecting the flow in the opposite direction and then reversing the flow direction. Reversing flow direction can be achieved by right-clicking the flow, selecting Connection Detail and selecting Reverse Direction.

- Create swimlanes via diagram menu (Diagram | Configure Swimlanes) or *Right click, Swimlanes*.
- To prevent the swimlane from being accidentally moved, check the *Locked* box in the Diagram Swimlanes dialogue box.
- To enable the swimlane names to be visible when scrolling to lower parts of the Activity Diagram, check the *Show Names in Title Bar* box in the Diagram Swimlanes dialogue box.
- The auto-generated alias for Activities populates the 'partition' field in the activity properties dialogue box.

12.2.3 Notation Adoption

The notation described in §12.2.1 does not prescribe how it should be applied. The intended use of activity diagrams within NHS Connecting for Health involves multiple users authoring/viewing individual activity diagrams. To provide the necessary consistency across the models it is required that a number of constraints are applied by authors when creating models. These constraints can be classified as either for mandatory adoption (rules) or recommended for adoption (conventions) and are identified in §12.2.3.1 and §12.2.3.2 respectively below.

Note that in addition to the core notation rules and conventions, use of both the notation and EA are subject to further, artefact specific constraints as described within §12.3.

12.2.3.1 Rules

12.2.3.1.1 EA Functions Supported by NHS Connecting for Health Activity Diagrams

Any function of the EA product that is not specifically mentioned within the authoring guidelines is not to be used within the production of an NHS Connecting for Health Activity Diagram. Functions to be considered for use within NHS Connecting for Health models should be highlighted to the Analysis Team and where accepted a further version of this document will be issued.

12.2.3.1.2 Structured Activities

Must contain activities or actions, and not just be used as a drawing convention.

12.2.3.1.3 Object Flows

Object flows are to be named to identify the object being transmitted. The name must be compatible with the receiving Action Pin and or Activity Parameter Node names.

12.2.3.1.4 Activities

- Every diagram must have one and only one initial activity. However two exceptions may apply which are:
 - Outline Query
 - Domain Query Functionality Activity Diagram
 - In both cases the Entry activities must have defined input Activity Parameter Nodes
- Every diagram must have one or more final activities. However two exceptions may apply which are:

- Outline Query which must have defined output Activity Parameter Nodes instead of a final activity.
- Domain Query Functionality Activity Diagram which may have defined output Activity Parameter Nodes instead of a final activity.
- Where a diagram is a child, the initial/final activities must match the incoming/outgoing flows of the parent activity.

12.2.3.1.5 Guards and Flows

- Activities must have an incoming and an outgoing flow (an exception here is the event and error handling activities described in §12.2.1.9). Two additional exceptions are
 - Outline Query
 - Domain Query Functionality Activity Diagram
 - In both cases the Entry activities must have defined input and output Activity Parameter Nodes
- All activities must only have one outgoing flow except where Activity Parameter Nodes have been defined.
- All outgoing flows from a decision must have guard conditions stated.
- All activities (except activity finals and flow finals) and decisions can only have a single incoming flow. The exceptions being activities with Activity Parameter Nodes where used in Outline Queries or Domain Query Functionality Activity Diagrams
- Guards for outgoing control flows from a decision must cater for all possible decision conditions and the guard conditions must not overlap.
- Forks and Joins to have single entry/exit flows respectively.
- Outgoing flows from a fork or join have no triggers or guards.
- All flow optionality is represented using decisions.
- Redundant use of control flows is to be avoided – the existence of an object flow implies control flow and does **not** require an additional control flow structure within the diagram.

12.2.3.1.6 Diagrams

- Naming of diagram elements avoids redundancy:
 - Where a role is identified within a swimlane, the activity name should not refer to the role name.
 - Where an object flow identifies the object name, the activity name should not refer to the full object name.
- Swimlanes have vertical orientation and are highlighted by: line colour blue, width 2.

12.2.3.1.7 Redundant Elements

Use of redundant elements is only allowed to provide context to viewers of the model. For example, a component may include elements from other (incoming or

outgoing) components which are defined elsewhere. Where this occurs the redundant elements must be identified by the use of a note (see §12.3.6).

12.2.3.1.8 Exception / Error Handling

Representation of errors and exceptions occurs in activity diagrams; however different approaches are used in each of these cases as follows:

- Error Handling: an error condition is definable and predictable. Where errors are to be modelled these are represented as alternate paths (see §10.3.1.6 and §12.3.4.2) within a Use Case.
- Exception Handling: exceptions are unpredictable events which are represented as an event (see §12.2.1.9) in an activity diagram. Where the exception is localised to an activity or set of activities the event will be an interrupt event (see Figure 39).

12.2.3.2 Conventions

To ensure that the meanings of the diagrams are clear, a number of conventions are recommended to be adopted by authors as follows:

12.2.3.2.1 Orientation of Flows from Decisions

Within any one diagram, all decision flows should have consistent orientation e.g. all vertical down arrows are guarded as 'No'.

12.2.3.2.2 Guards of Flows from Decisions

Where possible a decision point should be evaluated as a 'Yes' or 'No' response.

12.2.3.2.3 Time Delays

Where a time delay event is to be represented (see §12.2.1.9) and the time delay is known, this should be represented on an incoming flow for the receipt event as a guard (if the incoming flow is represented).

12.2.3.2.4 Redundant Swimlanes

Swimlanes which are redundant (i.e. empty) should not be used.

12.2.3.2.5 Coupled Flows

Where flows are chronologically associated between the same activities (for example a requesting flow and its response flow) the earlier flow should be positioned above the later flow.

12.2.3.2.6 Crossing flows

Flows should not cross if at all possible but if it is unavoidable all flows should cross at right angles.

12.3 Activity Diagram Types

Within the methodology Activity Diagrams are used as the format for two different analysis artefacts:

- SBM: required to establish the process scope of the domain analysis (see §12.3.1).

- BPM: required analysis artefact that describes the sequencing and process logic between Use Cases (see §12.3.2).
- Outline Query: A structured activity representing a high level view of a Domain Query (see §12.3.3).
- Domain Query Functionality Activity Diagram: defining the basic steps and detail of an Outline Query (see §12.3.4).

It should be noted that Use Case Activity Diagrams are not a formal part of the Phase 1 LRA methodology and so are excluded from further consideration. However, the Analyst may find activity diagrams useful tool to validate the scope of a Use Case, in this case see referenced document 2.

12.3.1 Stakeholder Business Model

The Stakeholder Business Model (SBM) is an early product in the analysis process for a project and is developed during **DOMAIN ANALYSIS AND SCOPING PHASE**. Crucially, the Stakeholder Business Model provides traceability of Analysis Models (see §8) to the domain(s) under analysis. A Stakeholders Business Model articulates the Stakeholders' process view of the domain and may be a direct reuse of Stakeholder artefact(s) if these exist. The purpose of the SBM is to define the potential scope of the domain entire scope of the domain. Later artefacts may focus on particular areas of interest and the SBM will help identify the boundaries for this analysis at a high level. Specifically the SBM should aim to:

- Gain an initial understanding of the domain.
- Clarify scope of the analysis required.
- Ensure stakeholder engagement.
- Analyse and agree outline business processes for the domain.
- Inform planning and development of downstream dynamic analysis artefacts.

Meeting this purpose requires that the Stakeholder Business Model is a model of (aspects of) the domain which is recognisable to domain stakeholders. The stakeholder may use their own concepts and terms at this point and these will be mapped to more formal UML artefacts at later phases in the analysis. Within a clinical context, the Stakeholder Business Model is likely to include elements such as:

Care Pathway

Business definition

An evidence based outline of anticipated care, placed in an appropriate timeframe, to help a patient with a specific condition or set of symptoms move progressively through a clinical experience to positive outcomes

Business definition source: NHS CFH Pathways Forum

Care Event

Business definition

A Care Event is any occasion during which a health care professional, and/or the patient, and/or their carer, makes a material contribution to the health care of the patient, resulting in a change to a patient's NHS Care Record

Business definition source: NHS CFH Pathways Forum

Where business models of the domain pre-exist, these can be reused in this methodology as the Stakeholder Business Model. Where such models do not exist or are inadequate, the Stakeholder Business Model can be created using the Analysts toolset. The unconstrained and optional nature of the Stakeholder Business Model allows the Analyst large flexibility in respect of its construction and nature, however it is expected that the use of a free form activity diagram (see §12) will be a typical approach.

12.3.1.1 Introduction

Where business models of the domain pre-exist these can be reused in this methodology as the Stakeholder Business Model. Where such models do not exist (or are inadequate), the Stakeholder Business Model can be created using the Analysts toolset. The unconstrained and optional nature of the Stakeholder Business Model allows the Analyst large flexibility in respect of its construction and nature, however it is expected that the use of a free form activity diagram (see §12) will be a typical approach.

The development process is expected to be via a mixture of workshop, interview and Analyst background reading/knowledge; wherever appropriate previous analysis on related or corporate areas should be taken into account, offering the opportunity to educate and inform stakeholders who have not previously been involved in such detailed work.

12.3.1.2 General Construction

Where the Stakeholder Business Models are to be produced in the form of Activity Diagrams, several Stakeholder Business Models may be required for a domain where the domain deals with multiple business areas. Alternative the use of Swim lanes may aid the demarcation of business areas.

12.3.1.3 EA Construction

The Stakeholder Business Models are to be contained within a discrete package within the Domain Analysis and Scoping Package (see Figure 6).

Reusing a pre-existing business model must involve any of the linking/importing mechanisms identified (see §13.4.30, §13.4.32 and §13.4.33).

12.3.2 Business Process Model

12.3.2.1 Introduction

The Business Process Model (BPM) provides a representation of the process relationships between Use Cases identified in an area of interest; where non-Use Case activities are excluded the BPM does not provide the complete definition of process semantic for the area of interest. Produced during the **LOGICAL ANALYSIS PHASE**, the BPM shows the Use Cases from the Use Case Catalogue and also represents any sequencing and decision logic between the Use Cases. The product is presented as part of the Logical Analysis Package.

Some business process models follow a linear path such that process control flows from one activity to another (see §12.3.2.2.2); however, it is possible that models

involve processes which are not linear, or vary in their order of presentation, such models are referred to as 'discontinuous' (see §12.3.2.2.3).

Development of process logic between identified Use Cases is based upon Analyst understanding gained during development of the Use Case Catalogue and refined by later development of Use Cases; note that the BPM activities also provide an access mechanism into the Use Cases. Work with stakeholders is therefore likely to cross all of these products and to be iterative in nature.

12.3.2.2 General Construction

12.3.2.2.1 Generic Construction

The BPM is primarily constructed of activities corresponding to the Use Cases which appear within the Use Case Catalogue. In some cases it may be necessary to introduce activities which do not represent a use case, to give context to the BPM. These activities are denoted as 'for information only' and will not be the subject of further analysis work. Such activities are candidate pre-conditions for subsequent use cases.

It is not expected that swimlanes will appear at the BPM level.

At this level all flows will be represented as control flows.

Only Use Cases which are triggered directly by actors (i.e. not called by another Use Case) will be shown on the BPM.

It is not expected that basic and alternate paths will be labelled on the BPM.

Terminology of content must be recognisable to domain participants.

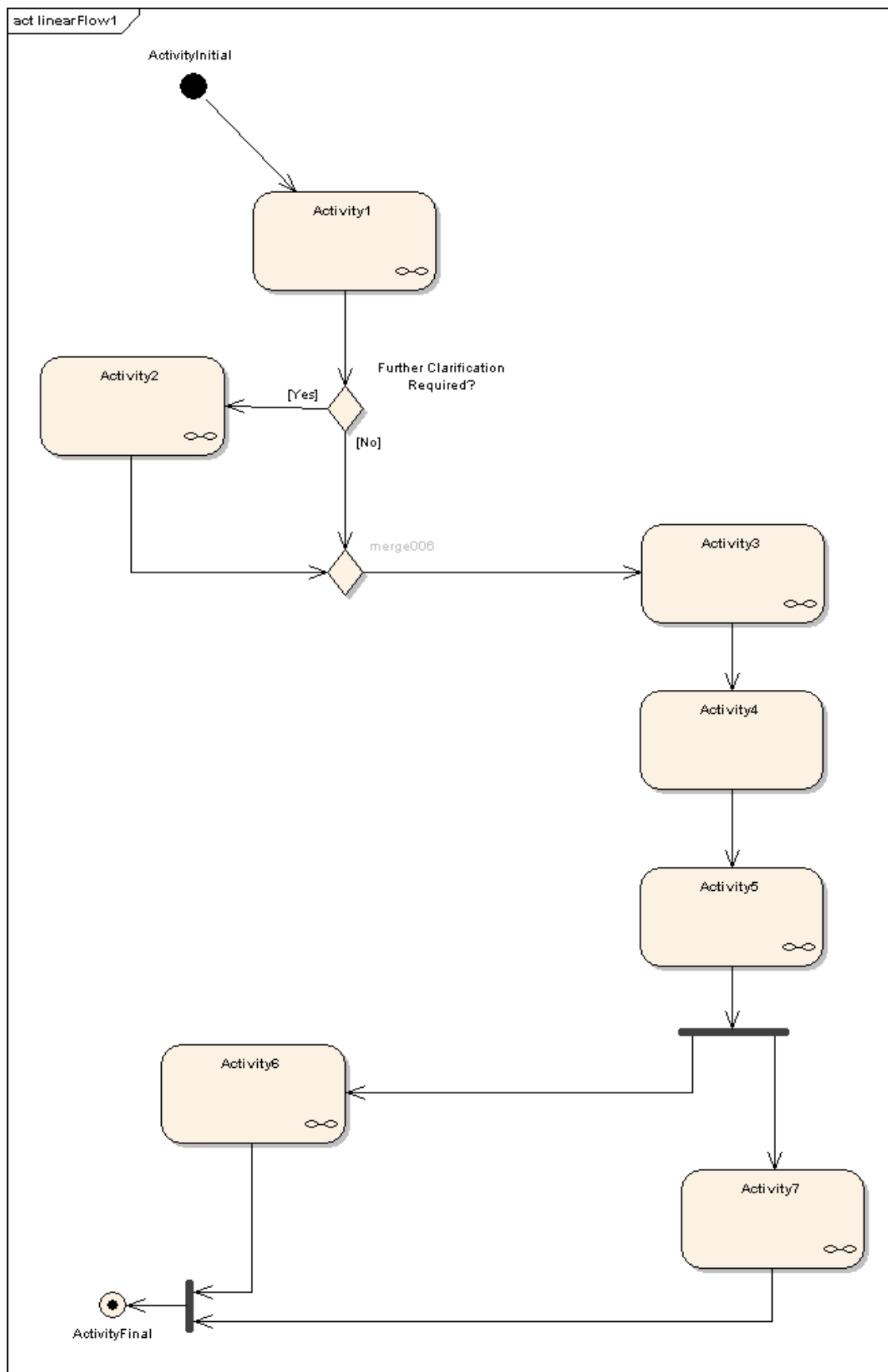
Where Stakeholder Business Model content is not represented within the BPM, this should be identified as out of scope in the SBM.

Domain Specific information concepts to be represented within the SIV.

Pre and post-conditions identified in the Use Case must not conflict with process as stated in the BPM.

12.3.2.2.2 Linear BPM

A linear BPM consists of a single 'level' model held within the BPM package. For this type of BPM, the process logic connecting the Use Cases can be described within a single configuration of Use Cases, an example is shown in Figure 43

**Figure 43: example of a linear BPM**

12.3.2.2.3 Discontinuous BPM

Most simple business process models follow a linear path where process control flows from one activity to another to the end of the encounter (Patient registration, Clinician–Patient interaction, Report publish, for example), such BPMs are described in §12.3.2.2.2. However, where processes are more complex this often involves non-linearity, or processes which vary in their order of presentation. In these cases the processes may be sufficiently discrete to be contained within one model but have a number of distinctly separate sections and the order of the sections cannot be easily represented, or dictated by, for example, decision points. Even where the process is apparently linear if significant time breaks may exist between some actions the simple approach to process description may not be appropriate. In these cases it is not appropriate to depict the process logic connecting the Use Cases within a single configuration of Use Cases and the BPM is considered to be ‘discontinuous’. Discontinuous BPMs are produced consisting of an overview of discontinuous activities (Use Cases) with supporting example configurations.

Discontinuous flows are discrete process flows (i.e. sequences of Use Cases) which are typically used to generate example configurations where each configuration is a specific assembly of some of the discontinuous flows which are composed by the Analyst to illustrate points of interest. Thus the configurations add the missing continuity between the elements of the discontinuous flow and are typically used to represent flows which are meaningful to service consumers (i.e. patients) rather than service producers (i.e. NHS staff), where the latter are more likely to recognise the discontinuous flows. An example of this is shown in Figure 44 where a number of discrete processes are represented as discontinuous flows which have been composed into example configurations (see Figure 45). A practical result of the creation of configurations is that the discontinuous elements are repeated across configurations, this leads to the following considerations:

- *use of colours*: it is possible to use colour schemes to identify to which discontinuous element the components of a configuration ‘belong’. In this approach each service provider of the overview has its own colour which is inherited by its child elements. Where administrative business processes are represented as discontinuous elements in blue and the clinical business processes are represented as discontinuous elements in pink.
- *avoiding redundancy across configurations*: Replicating elements within example configurations is allowed given the nature of the BPM.

12.3.2.3 EA Construction

This product is contained within the LRA Analysis package within the Project View, held within a BPM package. See Figure 6. The Use Cases are contained within the Use Case Model package under The Logical Analysis package within the Project View.

Construction is as follows:

- The activities representing the Use Cases are dragged onto the BPM as a simple link.
- The activities are constructed as structured activities. which drill down taking the user into the relevant BPM Diagrams.

- Additional elements required upon the BPM diagram to represent process logic (decisions, merges and any contextual activities) are to be held within the BPM package itself.
- Where an activity represents a Use Case this must be cross referenced using the relationship matrix functionality (see §13.4.27).

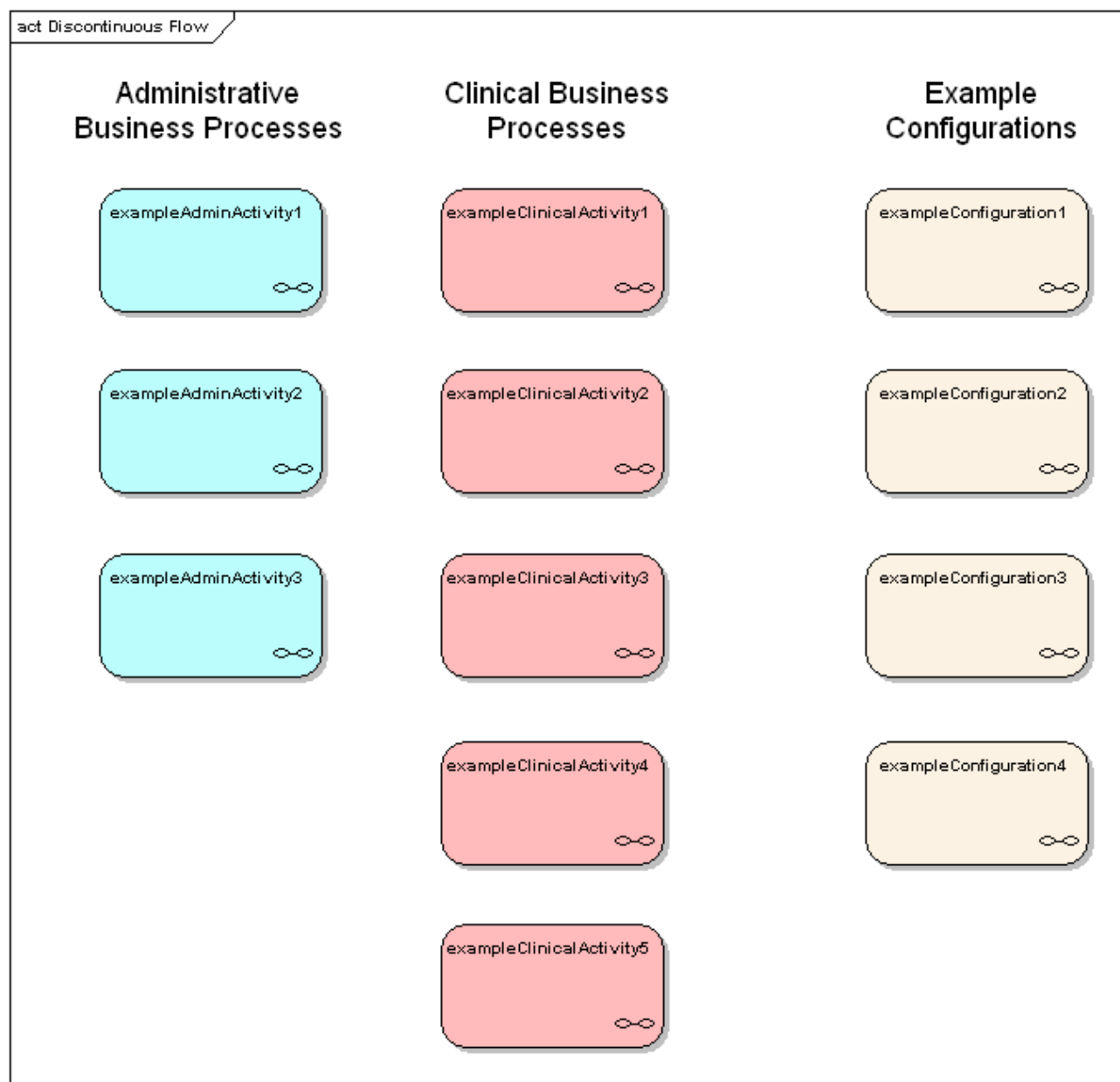


Figure 44: overview showing discontinuous flows and example configurations

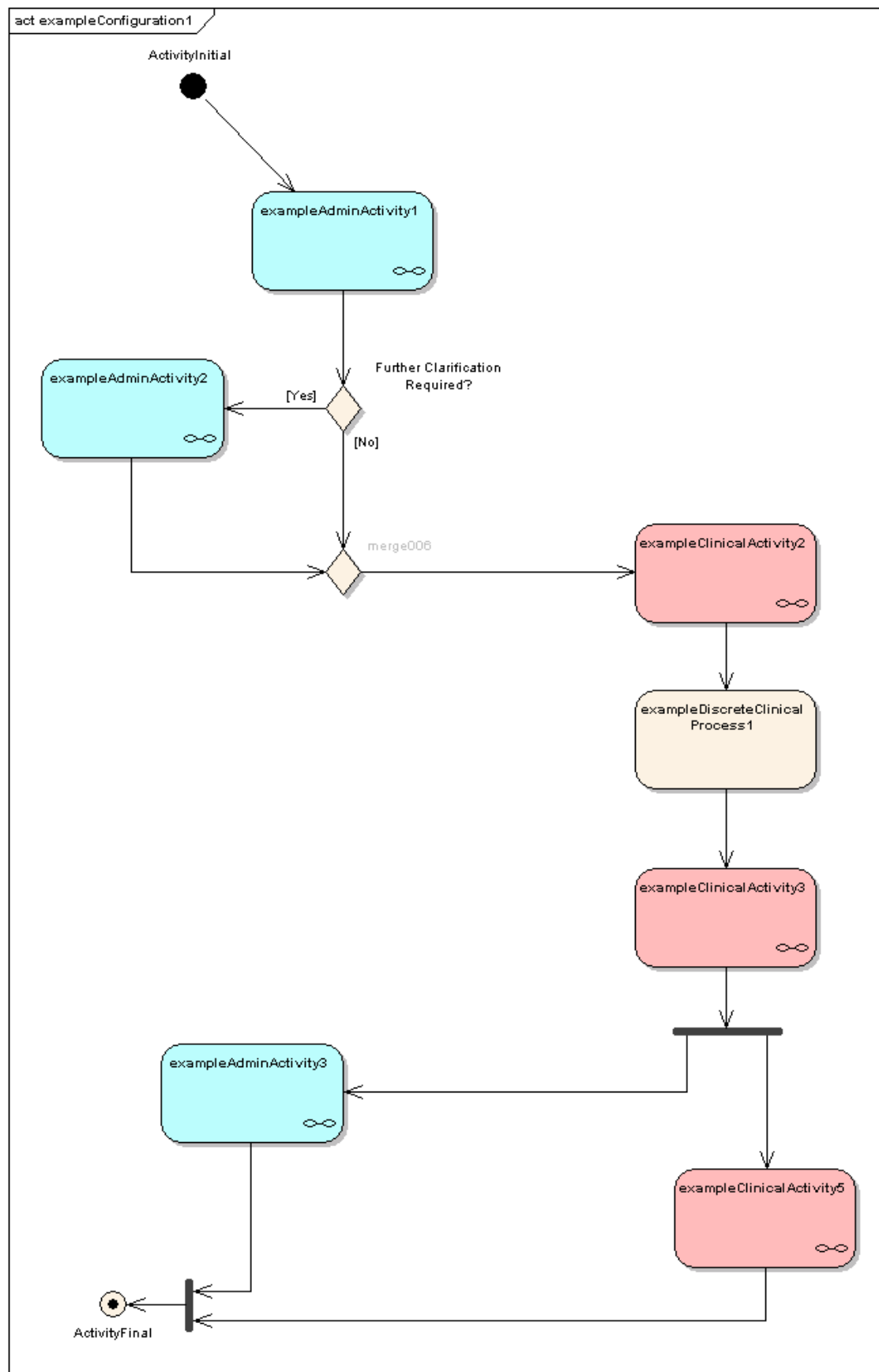


Figure 45: example configuration (a configuration of discontinuous flows)

12.3.3 Outline Query Diagram

12.3.3.1 Introduction

The Outline Query diagram provides a representation of the discrete queries held within the model. Produced during the **LOGICAL ANALYSIS PHASE**, the Outline Query diagram shows the Outline Queries held in the Outline Query package.

12.3.3.2 General Construction

Outline Queries are structured activities that represent a high level the logical definition of a query. They are further elaborated by the Domain Query Functionality Activity Diagram. Outline Queries are discreet with known inputs and outputs but can form part of other queries. Consequently, they can be reused in multiple contexts and cannot represent a specific linear process. They are therefore 'discontinuous'. In this regard they should be treated in the same manner as a discontinuous BPM.

The key exceptions are:

- Outline Query discontinuous flows represent the construction and process of a query from more granular Outline Query building blocks.
- The broader Outline Query itself should also be regarded as an Outline Query in its own right.
- The structured activity representing an Outline Query must have named input and output Activity Parameter Nodes (see §12.2.1.3)
- The Structured Activities (where decomposed) will reference a Domain Query Functionality Activity Diagram (see §12.3.4).

In these cases the processes may be sufficiently discrete to be contained within one model but have a number of distinctly separate sections and the order of the sections cannot be easily represented, or dictated by, for example, decision points. Even where the process is apparently linear if significant time breaks may exist between some actions the simple approach to process description may not be appropriate. In these cases it is not appropriate to depict the process logic connecting the Use Cases within a single configuration of Outline Query activities and the BPM is considered to be 'discontinuous'. Discontinuous BPMs are produced consisting of an overview of discontinuous activities.

12.3.3.3 EA Construction

The product is held under the Outline Query package under the Logical Analysis package (see Figure 6).

Construction is as follows:

- Where an Outline Query has been elaborated (Domain Query Functionality Diagram, see §12.3.4), the elaborated content specific to the Outline Query will be held under each Domain Query Functionality package in the LRA Analysis package.
- The activities are constructed as structured activities (see §12.2.1.2), which drill down taking the user into the relevant Domain Query Functionality Diagram.

- Additional elements required upon the Outline Query Diagram to represent process logic (decisions, merges and any contextual activities) are to be held within the Outline Query package itself.
- Outline Query structured activities must be stereotyped as 'outline query'

An example Outline Query diagram is shown in Figure 46.

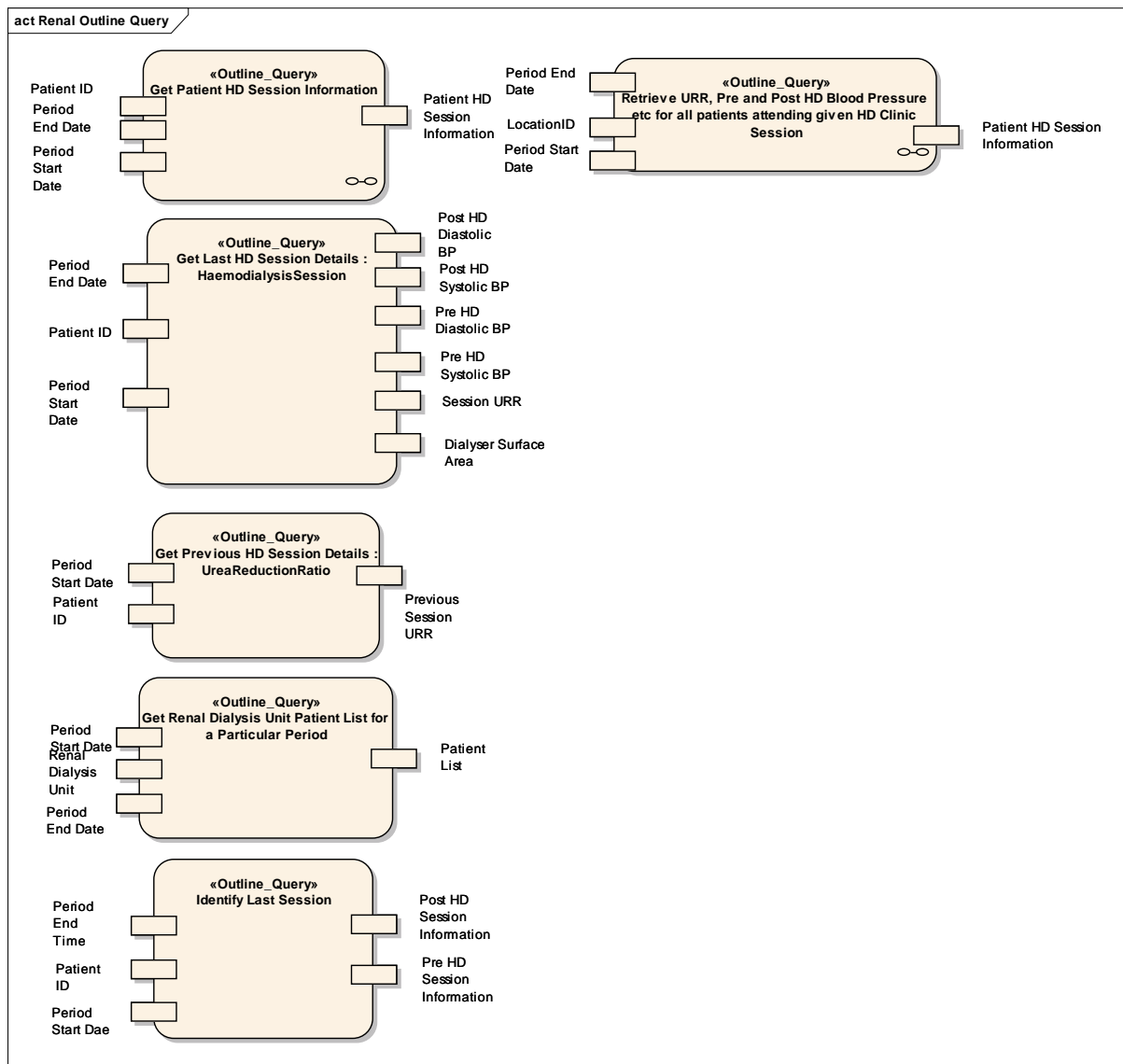


Figure 46: outline query diagram

12.3.4 Domain Query Functionality Activity Diagram

12.3.4.1 Introduction

A Domain Query Functionality Activity Diagram (DQF) is developed to analyse and document the basic and alternate paths of each Outline Query (see §12.3.3), which also provides the detailed representation of individual activities within the Outline Query Diagram.

The Domain Query Functionality Activity Diagram is bounded by its Outline Query.

12.3.4.2 General Construction

It is expected that the activity diagrams will represent both the basic logic of the query and any alternative logic. The alternate logic paths are initiated by decision points on the Domain Query Functionality Activity Diagram.

The name of the parent activity within the Outline Query diagram should be used as the name of the Domain Query Functionality Activity Diagram.

Domain Query Functionality Activity Diagrams can only have sub-activities if they use another Outline Query structured activity as part of their process logic. In all other cases Actions (see §12.2.1.5) should be used to describe the internal tasks undertaken by an activity.

The name of a single activity within the Domain Query Functionality Activity Diagram must not replicate the name of the Outline Query.

Where a known information requirement is required this is represented by an object flow

Communication object flows must be connected to or from Action Pins (see §12.2.1.5.1) or Activity Parameter Nodes (see §12.2.1.3).

All final activities should have matching post-conditions within the Outline Query.

Where a Domain Query Functionality Activity Diagram calls out to another Outline Query, the call from the invoking activity/action must match the trigger of the invoked Outline Query.

Where a note is recorded against an activity, it must not affect the meaning or logic of the activity.

From a decision point there must be at least one alternate path. Alternate paths will be denoted as [Alt_n – *Guard condition*], where n is an integer, incremented for every decision point. The incoming flow must be repeated as one and only one of the out-going flows (e.g. if Alt_1 in, then one of the out-going flows must be Alt_1). See example in Figure 47.

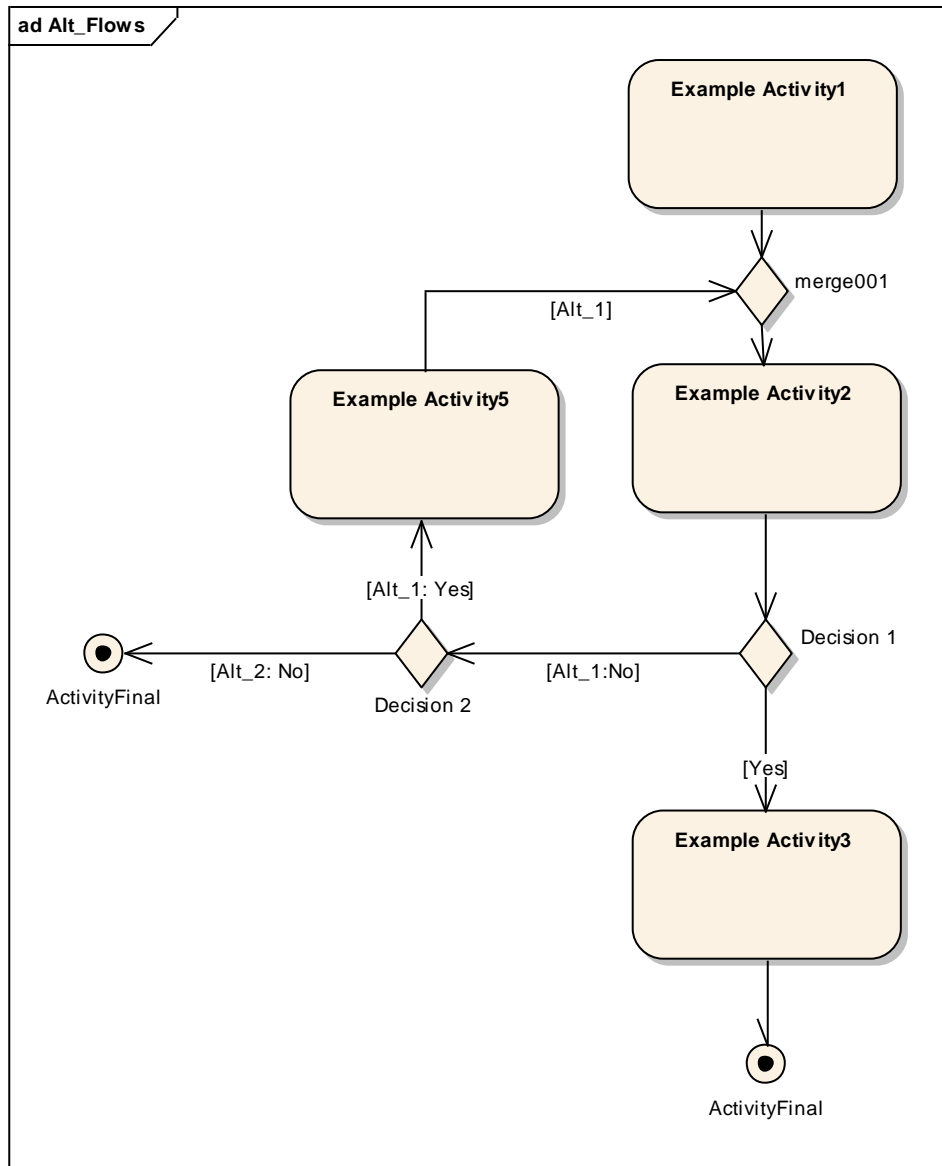


Figure 47: alternate flows in domain query functionality activity

12.3.4.3 EA Construction

When invoking an Outline Query - drag on invoked Outline Query from the Outline Query package as a simple link. It will appear as an activity with a composite link to invoked Domain Query Functionality Activity Diagram. Embedded Activity Parameter nodes will need to be made visible.

Naming of alternate logic paths – entered in guard condition field on flows.

Individual activities or actions within a Domain Query Functionality Activity Diagram should not be re-used in any other Domain Query Functionality Activity Diagrams.

12.3.5 Documentation

Text entries can be associated with specific diagram elements and composed as model documentation automatically. Note these are not displayed as part of the model and are thus distinct from annotations

12.3.6 Annotation

Whilst annotations can be added directly on a diagram to any element and will be displayed as part of the model – they will not form part of its internal documentation or be linked to the specific element in RTF or HTML outputs. It is therefore required that significant annotations be added using the element's properties dialogue (see §8.4.1).

12.4 Domain Query Functionality Activity Diagram Example

See Below:

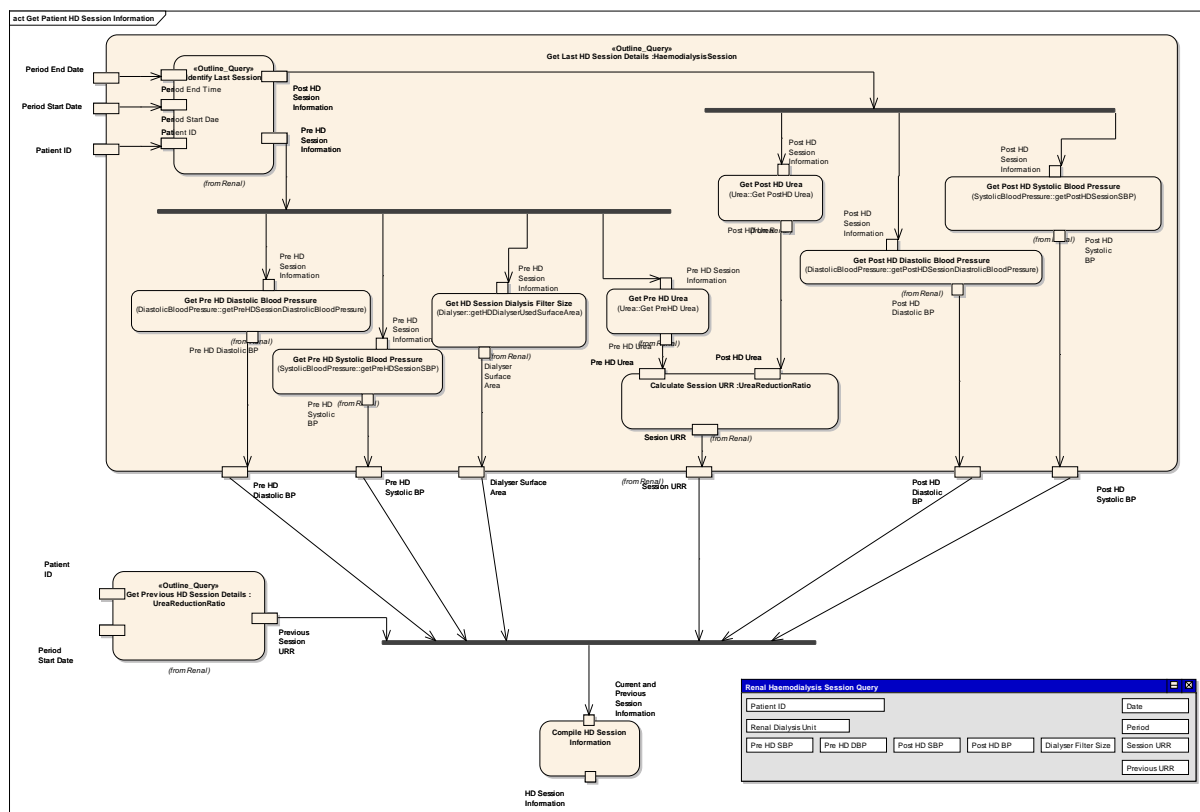


Figure 48: domain query functionality activity diagram example

12.5 Activity Diagram Examples

See example diagrams §18.2 (Stakeholder Business Model), §18.5 (Business Process Model), § 18.7 (Outline Query Diagram) and §18.10 (Domain Query Functionality Activity Diagram)

12.6 Activity Diagrams Quality Review

See product descriptions §17.2 (Stakeholder Business Model), §17.4 (Business Process Model), §17.9 (Outline Query Diagram) and §17.12 (Domain Query Functionality Activity Diagram)

12.7 Activity Diagram Issues

12.7.1 Alias

When auto-numbering (see §13.4.1) is activated on Activities in EA, the auto-number populates the 'partition' field' in the activity dialogue box. The auto-generated number is always visible on the activity.

13 Appendix 1: CASE Tool Use

13.1 Enterprise Architect

To control the various components and views of a process model to ensure the integrity of the overall model and minimise its internal redundancy requires an intelligent modelling tool rather than a drawing tool. A number of tools have been analysed based upon their fitness for purpose and ease of use (given the potentially diverse user base). This assessment activity has concluded that the most appropriate tool is Enterprise Architect from Sparx Systems

(<http://www.sparxsystems.com.au/products/ea/>) and this tool will be used as part of the methodology described within this document. The example process models and components within this document were produced using Enterprise Architect.

13.2 EA Options Configuration

EA has a number of customisable global settings which affect the way that the application behaves. In order for the users to have a uniform and optimal architecture environment, it is required that the following settings are configured after EA has been installed.

NB: These settings have global application effects and are not specific to a certain project or model.

13.2.1 Increase Minimum Size of Diagram Images

There are diagrams within the EA Business Architecture toolset which are so large that the default setting of 48 megabytes is not sufficient to support some of the larger diagrams. By increasing the Image Memory Limit, the larger diagrams can be accommodated. It is important when you have very large diagrams, as it affects the point at which Enterprise Architect starts to scale down the image; a low memory setting means it scales the image sooner. Select from the main menu:

- *Tools / Options*, (or Ctrl F9) – opens the option menu (see Figure 49)

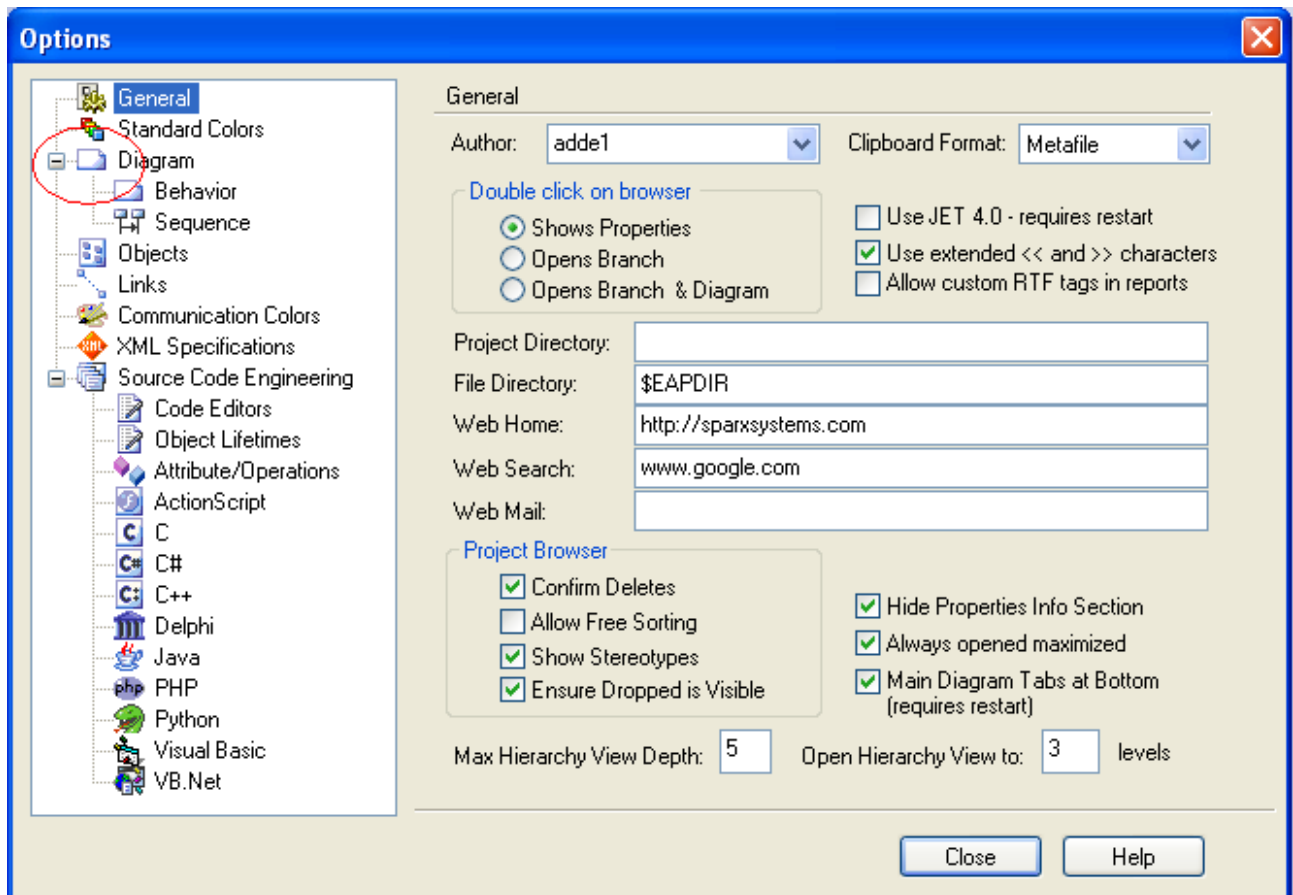


Figure 49: option menu

- Select the diagram option as indicated and that will open the diagram configuration (see Figure 50).
- Select the dropdown for *Image Memory Limit* and select 512 megabytes.
- Select the *Close* button

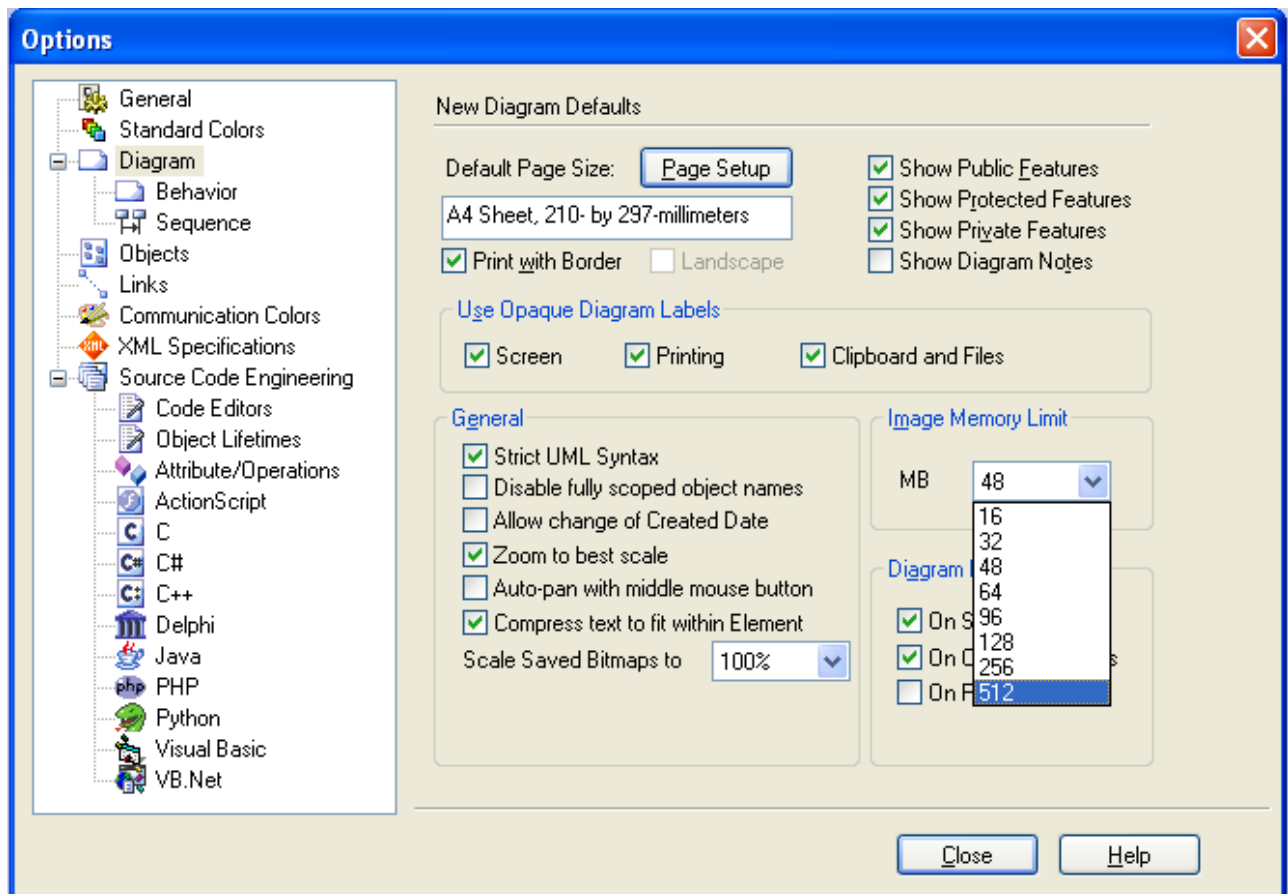


Figure 50: option menu, diagram setting

13.3 Documenting or Annotating Models/Projects via EA

13.3.1 EA Model Registers

- Model Tasks can be used while under development but must be removed before release.
- Model Issues can be used while under development. For further information see Chapter 15
- A Model Glossary function exists, and may be used to define terms used within a project.

13.3.2 Notes on or Attached to Elements

- see §8.4.1

13.3.3 Publishing Control Information

- Before release a general note must be added to the Navigation Page, see §6.2.

13.3.4 Publishing to HTML

- When producing an HTML report for external review (Project, Documentation, HTML report or [Shift F8], and Generate) check the 'Glossary' and 'Model Issues' boxes on the dialogue to ensure their inclusion.

- The issues and glossary entries are accessible from the HTML by expanding the System hyperlink in the Activities window and clicking on 'Glossary' and 'Model Issues', or as a separate file (Issues.htm).
- Note that for Activity Diagrams, activity notes are displayed by clicking the activity box, but notes within structured activities are not displayed.

13.4 EA Common Action Instructions

13.4.1 Auto-Numbering of Elements

- The unique identifier is generated using the Auto-numbering feature in EA (see Settings | Auto Name Counters.) The prefix should be the project identifier (normally between 2 and 4 letters, as per domain specific part of FileCM IDKey). The counter should start at 001. The suffix should identify the type of artefact: see Figure 51, in this example UC for Use Case – element types to have auto numbering applied (and their suffix identifiers) include: Activities (ACT), Use Cases (UC), Actors (A), Requirements (RQ) and Classes (CLA).
- The numbering can be activated to populate either the *Name* field, *Alias* field or both. (Generally this should be applied to the Alias field; for merge elements it should be applied to the Name field).
- Note - the name field can be overwritten, with no way of restoring the auto-number.
- Note – if an element is created with an auto-numbered alias and is then cut/copied and pasted as a new element the alias is NOT incremented (i.e. the alias is not unique). This should be avoided if possible and is a significant deterrent to cut/copy and pasting elements

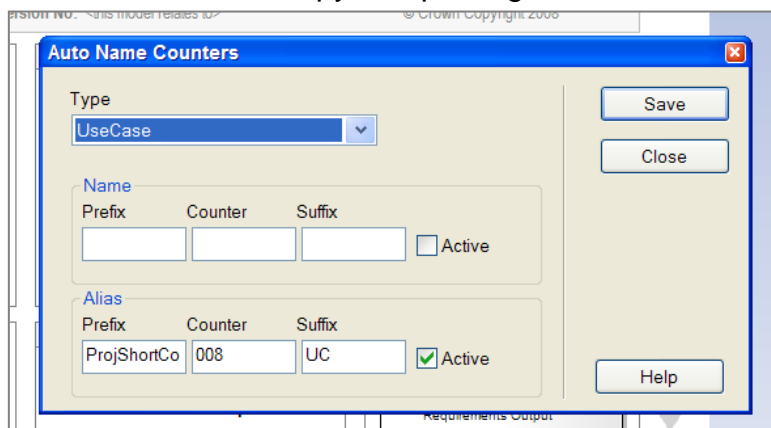


Figure 51: auto element naming

13.4.2 Drawing Right Angle Links (Associations, Flows etc)

- Position cursor at required point on line, right click and choose 'Bend line at cursor'.

13.4.3 Deleting Elements

- Deleting elements from diagrams does not delete the element from the model, just from the diagram.

- Deletion from the model can be achieved either by deletion from the 'Project View' (Right click | Delete) or by deletion from a diagram using ctrl Del rather than 'point & click' deletion.

13.4.4 Set as the model default

- Display the relevant diagram
- From the Menu Bar
- Diagram | Make Model default

13.4.5 Create a package

A package is a grouping mechanism within the Project View. Interchange of parts of an EA project file involves the export and import of packages.

- Ensure the Activities package is highlighted in Project Browser (Quick key to display browser, Alt 0)
- From the Menu Bar
- Project | New Package
- Enter name, as per naming convention
- Click *OK*

13.4.6 Display/Hide Element Features

- From the Menu Bar
- Diagram | Properties
- (Quick key, F5, or right click on a blank space and choose *Diagram Properties*)
- Elements tab
- Check/uncheck required display configuration
- Click *OK*

13.4.7 Drag and drop the elements into the correct package

- Open the relevant diagram
- In Project Browser, right click on each element
- If *Locate in Current Diagram* isn't greyed out, drag it into that package

13.4.8 Create a new diagram

- From the Menu Bar
- Diagram | New Diagram)
- Enter name, as per naming convention
- Click *OK*

13.4.9 Generate HTML set

- Ensure that all components are saved
- In Project Browser ensure the required package is highlighted
- From the Menu Bar
- Project | Documentation | HTML Report

- Choose an output location and title
- Ensure Glossary and Test Cases are selected; determine if Model Issues should be selected
- Click Generate
- *HTML Complete* is displayed, click *OK*
- Click *Close*

13.4.10 Opening HTML Set

The HTML Extract of a model is a collection of files and must be distributed contained within a folder to ensure that relative links are resolvable. In most cases this folder will itself be 'ZIPped' or 'RARed' (to minimise overall size and to allow for distribution) which requires un-packing as a first action. NB. it is common practice for email distribution of zip/rar files to be subject to security constraints, to avoid distribution problems the filename extension is often changed from .zip/.rar and must be changed back to .zip/.rar to allow for unzipping.

Once the zip/rar file is opened the user must place the contents within his/her own file system because the browser will be unable to resolve links to files stored in temporary folders (such as those used by zip/rar files).

Within the (now unpacked and moved) files is a file called 'index.htm', the model is opened by viewing this file within a browser (by double-clicking). It should be noted that the HTML Extract includes active content which may conflict with the security settings of the reading platform. This active content is important for navigation within the model. It is recommended that security settings are set to allow active content to be read. Where this is not possible it is likely that the user will be prompted to accept blocked content.

The following statement is recommended as advice to receivers of the HTML set – 'The enclosed file is in the form of a ZIPped/RARed HTML set, however the contents of the zip/rar file must be placed on your own file system to allow your browser to resolve links. When the contents are unpacked please double click the file 'index.htm' to open the model. (NB. the HTML model includes active content which may conflict with the security settings of the reading platform. This active content is important for navigation within the model. It is recommended that security settings are set to allow active content to be read).'

13.4.11 Generate RTF output

- Ensure that all components are saved
- In Project Browser ensure the required package is highlighted
- From the Menu Bar
- Project | Documentation | Rich Text Format (RTF) Report
- Choose an output location and title
- Select and Apply Template
- Click Generate
- Document successfully created is displayed, click Close

13.4.12 Import XMI

- Create a new unnamed package in the Project Browser
- Select the package in the Project Browser and Right click it
- From the Displayed menu options
- Choose Import/Export | Import Package from XMI file
- Click Import Package from XMI File ...
- Find the file required for import using the Windows browse option
- Click *Import*
- Click the Yes button (you will be warned that the *current package will be overwritten*) – the import will now take place
- Click the *Close* dialogue button

13.4.13 Export to XMI

- Select the required package in the Project Browser and Right click it
- From the Displayed menu options
- Choose Import/Export | Export Package to XMI file
- Click Export Package to XMI File ...
- A dialogue will open, create a path and filename for your XMI output
- Select all the options in the dialogue relevant to your output
- Click the Export button – the export process will now take place – wait until complete
- Click the *Close* dialogue button

13.4.14 Using the Search Engine within an EA Model

EA has a customisable model search facility which enables the finding of any object / element / text included within the subject model. This facility can, for example, find all the instances of a particular class's usage (particularly useful for tracking down every instance of where a datatype is used); help find orphaned classes (see §13.4.14.3); and allows the creation/saving of bespoke searches. This section provides a limited description of this facility including the most used searches, as well as showing how to create your own searches.

NB: The search facility is quite complex and the results returned can sometimes be misleading - the returned output should be checked against known parameters to ensure it provides the required information. Further details about how to use the search engine can be found in EA's help facility.

13.4.14.1 Simple Search

To perform a simple search to find where an object is located within a model:

- From the Menu Bar
- *Edit | Find in Model*, (or Ctrl F) – activates the search dialogue (see Figure 52)

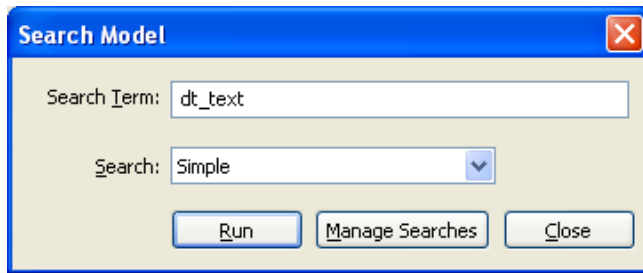


Figure 52: search dialogue

- With the Search type set to *Simple*, the search term can be input by pressing *Run* – this returns ‘hits’ in the *Output* window (see Figure 53).
- If a ‘hit’ is selected this object is highlighted in the Project Browser.

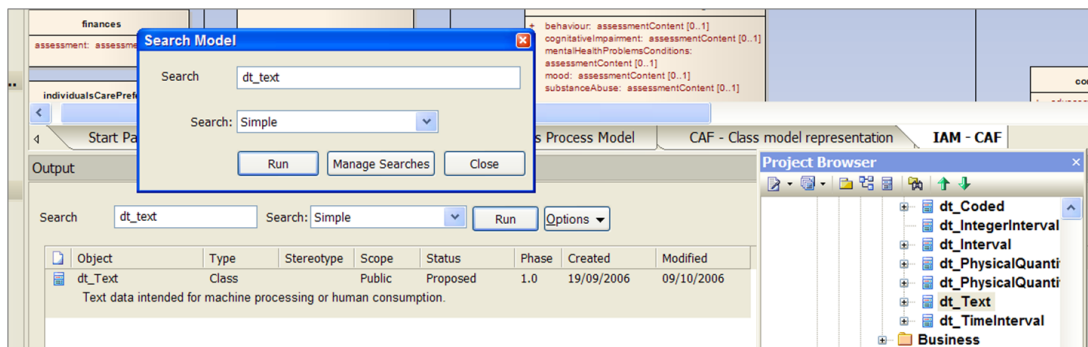


Figure 53: search output

13.4.14.2 Extended Search

An Extended Search enables searching for any occurrence of the element/text entered in the Search field. This is useful for finding multiple uses of the same item (e.g. to list every instance of where a datatype is used)

- Open the Search dialogue (using the top menu: *Edit | Find in Model*, **or** Ctrl F), enter the search text in the search field and set the Search type to *Extended*, then click on *Run*.
- The *Output* window will now show the names of all the classes and locations which hold or have an instance of the search item (see Figure 54).

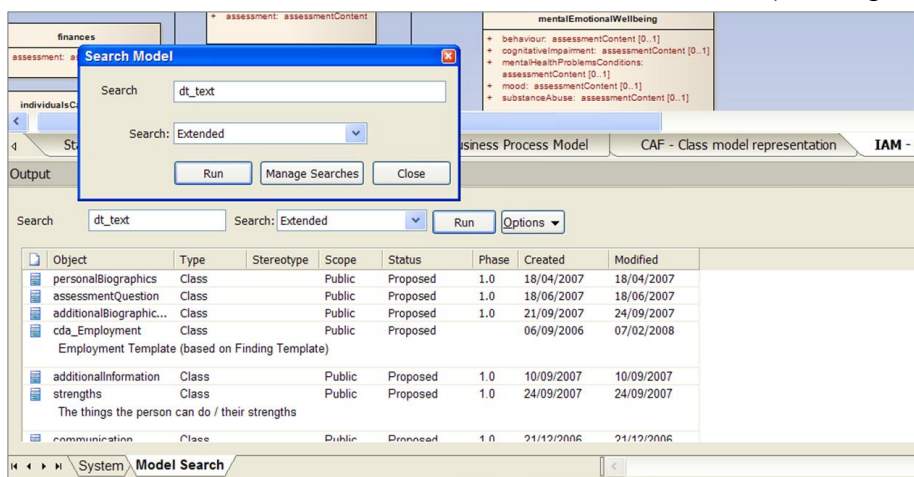


Figure 54: extended search

- The *Extended* search is also particularly useful for finding attributes within classes, which can be quite difficult to find. Type in the name of the attribute, the list will show the name of the elements where that attribute may be found (see Figure 55). The output can then be selected and the object or class it relates to is shown in the Project Browser (from where its appearance in diagrams can be identified by right-clicking and selecting *In Diagram*).

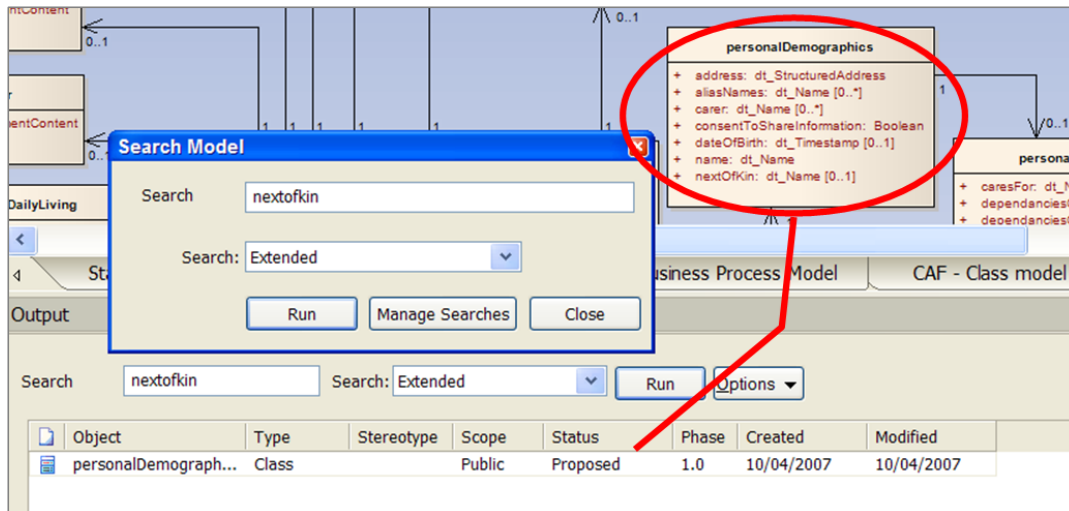


Figure 55: attribute search (using extended searching)

13.4.14.3 Find Orphans

Orphan elements are those elements which exist within a model (i.e. they appear in the Project Browser) but are not used within any diagrams. Orphan elements can occur for many reasons and a characteristic of the modelling tool EA is that deletion from a diagram does not lead to deletion from a model, this leads to wide scope for a model containing large numbers of orphans. Before publishing/reviewing a model it is required that these orphans are identified and deleted - the search function provides a very useful approach to meeting this requirement, as follows:

- Open the Search dialogue (using the top menu: *Edit | Find in Model*, **or** Ctrl F), leave the search field blank and set the Search type drop-down list (top right) to *Find Orphans* – then click on *Run*.
- The *Output* window will show the names of all the elements that are currently considered as orphaned. Clicking on the ‘hit’ results in the element being selected within the *Project Browser*.
- Delete the class if it has no further use for your model - NB: Caution - The orphan finding routine does not check if there are any child elements under an element and so it can delete active elements inappropriately (including in some cases the entire activity diagram).

13.4.14.4 Creating and Saving Searches

The EA search engine includes a set of pre-defined searches and has scope for extending this set with user-defined searches which can be named/saved for future use:

- Open the Search dialogue (using the top menu: *Edit | Find in Model*, **or** Ctrl F)
- Click on the *Manage Searches* button (see Figure 56).

- Two options exist for creating a new search:
 - To base a new search on an existing one; choose one of the pre-defined searches: *Simple*, *Extended* etc., from the search-type drop down list (upper right) - *Extended* shows nearly all the filters which support searches on most of a model's parameters. Click on the *Copy Search* button; input a search name and enter the required search parameters.

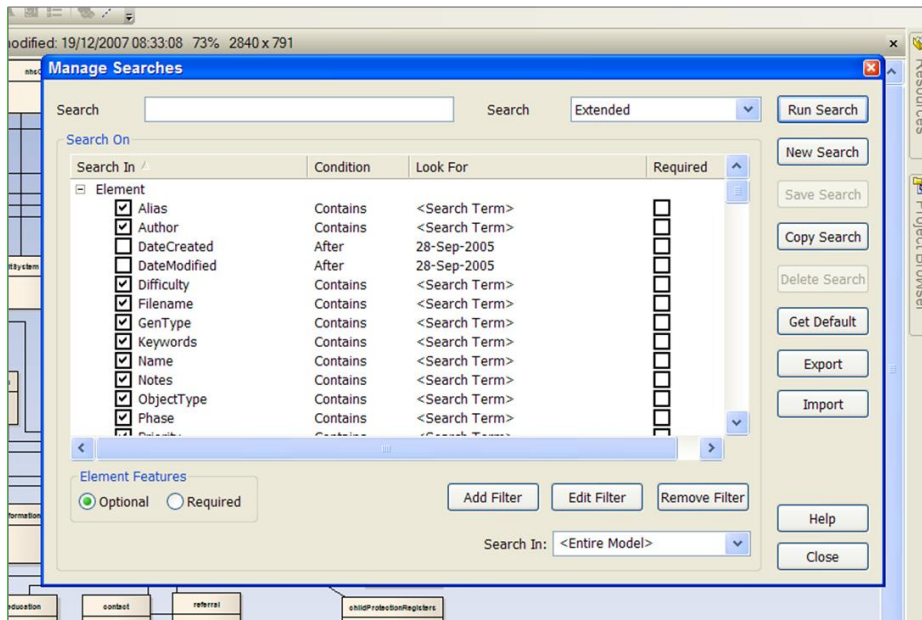


Figure 56: search management dialogue

- To create a new search without modifying an existing search click on the *New Search* button in the search management dialogue (see Figure 56) and enter a name for your search in the blank field shown in the dialogue – make sure the relevant query builder radio buttons are selected and then click the *Save Search* button. Once created, the new search can have filters added/edited.
- **NB:** It is advisable that new/modified filters are executed (by clicking the *Run* button) to ensure the search operates as required.
- Saving the search makes it available for future use and it will be added to the drop-down list along with *Simple*, *Extended* etc.
- An Example SQL Script for a model search can be viewed in Appendix 4: SQL Model Search Script.

13.4.14.5 Printing Search Output

There are a number of options for using/printing search outputs:

- A list of all 'hits' from the search execution can be generated by right-clicking anywhere in the *Output* window (see Figure 57) and selecting *Print Results* from the menu.
- By selecting individual search 'hits' (hold down Ctrl key for more than one item) or all 'hits' (Ctrl A), it is possible to make various uses of the selection, most importantly to:
 - Copy the selected items to the clipboard for use elsewhere

- Generate an RTF Report output
- Identify in which diagrams the element represented by the 'hit' is used.

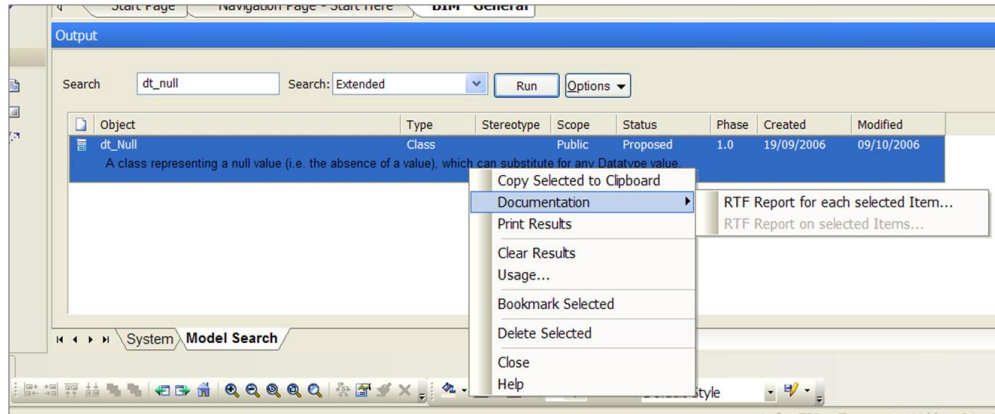


Figure 57: search output

Alternatively, a defined model search can be incorporated into RTF output. This is not an intuitive process however. The method is not described in this guidance and the analyst should study fully the tool's help files before embarking on this approach.

13.4.15 Caption Bar

- The caption bar appears at the top of the diagram pane, and shows diagram properties (name, created date etc)
- The bar also contains an X button, which will close the current diagram.
- If the caption bar is not visible, uncheck the *Hide Diagram Caption Bar* setting. (View | Visual Style | Hide Diagram Caption Bar)

13.4.16 Format Hyperlink to Diagram

- Drag diagram from Project Browser to main diagram pane
- In 'Select Element Type' dialogue box, choose to drop the diagram as a Hyperlink
- Right click on the hyperlink
- Select *Properties* from the menu
- To change the displayed name of the link, edit the *notes* section; to Hide the icon check the '*Hide Icon*' box
- Click *OK*
- To change the Font, Right click on the hyperlink, select *Appearance* then *Set Font*.

13.4.17 Printing and Exporting for use in MS Word

- The diagram title is not included on a hard copy print out unless the diagram is set to 'Fit to one page'.
- The simplest way to send the model to Word for external review is to copy all and paste into Word (CTRL A, CTRL C, and then, in Word, CTRL V). Note that the diagram names are always included. These diagrams can be annotated by a right click and 'Edit Picture'.

- NB an alternative approach exists which involves the generation of an RTF (Rich Text Format, which is readable in Word) report using Project, Documentation, Rich Text Format Report and then completing the dialogue box.

13.4.18 Copying Elements

- When copying an element a dialogue box is displayed asking to confirm it as a 'Simple Link', click 'Only show this dialogue when Ctrl+Mouse drag is used' to disable the dialogue box.
- Use Ctrl+Mouse drag if you need the dialogue box in the future.

13.4.19 Creating Links Between Elements that are spatially separate

- Reduce the diagram until the two elements are visible.
- If this isn't possible, connect the link from the correct element to **any** element in close proximity, then highlight the link, right click and choose *Connection Detail* and *Set Source and Target* and change the destination to the correct element.
- Note: Elements appear to be listed in chronological order of creation. This obviously can be used to redirect existing links between elements, as opposed to deleting and re-entering.
- To redirect an existing object flow, add an object node to the new activity (*Right click, Insert Embedded Element, Object Node*) and then amend the flow, as above, to point to the new object node, rather than the activity and delete the redundant object node from the activity (if necessary).

13.4.20 Moving and Resizing Elements

- To move an element (or multiple elements) in small incremental steps, highlight the required element(s) and press Shift and the arrow key for the required direction.
- Similarly to re-size an element (or multiple elements) in small incremental steps, highlight the required element(s) and press CTRL and the horizontal or vertical key to re-size in the required direction.

13.4.21 Moving Between Open Diagrams

- The ALT and left and right arrow keys cycles through all open diagrams

13.4.22 Model Authors

- Use Settings menu (Settings | People | Project Authors) to assign authors to a project; use full names. These may then be used within the Model Tasks and Issues windows.

13.4.23 F2 Hot-key (Edit element text on diagram)

- Select an element or attribute on a diagram, press F2
- The visible text can now be edited directly in the diagram,
 - Press enter to accept / escape to cancel changes
 - N.B. It is not possible to use this method to amend information held under an item in the properties dialogue when displayed as a linked-element in folded-corner annotation boxes (e.g. notes) – these can only be amended in the relevant elements properties dialogues, Or...

- Alternately it is possible to edit any text displayed in the notes pane (use the *View / Notes* top menu) displayed to the right of the working space
 - move from element to element by selecting it
- This facility can however be used to amend the Name of any element displayed on a diagram however it was created

13.4.24 Watermark on diagrams

- Click (tools | options)
- Select Diagram Behaviour
- Enter watermark text in Text field, check "use watermark"
- Note the water mark will be active on all diagrams, and will be applied to HTML & RTF output and via "add diagram to project clipboard" (cut and paste to another application).

13.4.25 Change Connector source/ target

- Right click the connector
- Select (Connection Detail | Set Source and Target)
- Choose 'from' or 'to' element from drop down list. Click OK.

13.4.26 Finding Diagrams in which an Element Exists

- Find the element to be queried the Project Browser and right-click on the element concerned
- Select the menu item Element | Find in Diagrams (or Ctrl U)
 - If there are any links these will be highlighted in the Element Usage dialogue which will open (no dialogue will open if there are no links)
 - This will also show you details of any other links this element has with other elements in the overall model
- Select the desired reference from the dialogue and click Open - this will take you directly to the referenced element.

13.4.27 Understanding & Using the Relationship Matrix

The Relationship Matrix enables the establishment and review of a nominal relationship between elements across the artefacts within a model. Such relationships are represented as elements within a matrix as shown in Figure 58 and are particularly useful for tracing the development of an information concept from its original insertion forward to its implementation e.g. a business/information concept may be outlined in a Use Case, defined in a SIV, further articulated by the use of one or more classes in the IAM which in turn is further implemented by classes or packages within a DID.

The relationship matrix is used to identify links and dependencies between parts of the model in order to add richness to the model and to facilitate impact analysis of changes and to ensure maintainability. Use of the relationship matrix is aided by a 'rule of thumb' that when checking the relationships between successor and predecessor products each successor product element should have only one

relationship identified in the matrix. This is simply a rule of thumb and various exceptions apply such as:

- Where abstract classes (see §11.3.1.1.5.1) are used - there **may** be multiple relationships shown in the relationship matrix.

When a relationship is required for model elements within this methodology, this is identified in the relevant section describing the element type (see §9 to §Error! Reference source not found. usually under 'EA Construction' or within its product description.

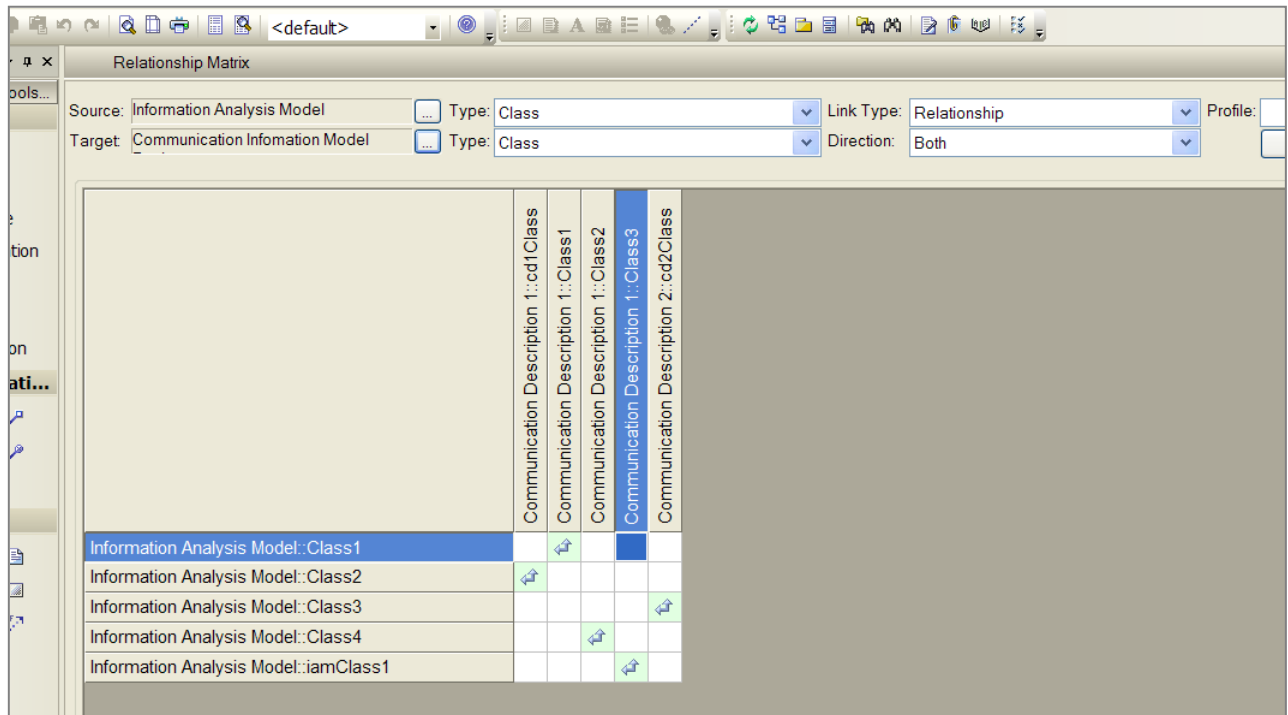


Figure 58: the relationship matrix

13.4.27.1 Viewing the Relationship Matrix

- Select *View | Relationship Matrix* from the main menu bar at the top of the screen
- Select the *Source* and *Target* packages from the tree structure (e.g. SIV and IAM, or UC)
- Select the required *Type* of element from each package (e.g. Class, Object, Activity, Package etc.)
- Select Link Type of *Relationship*.
- All elements of the chosen type in that package will then automatically be displayed in the matrix (NB: if the model is large displaying the selection may take some time).
 - This review profile can also be saved to make future access to this selection easier, to do this:

- Click on the *Options* button at the top-right of the Relationships Matrix window
- Select *Profiles | Save as New Profile* and enter an appropriate name for the profile
- The profile can then be recalled at any time by selecting the saved profile from the drop-down *Profile* dialogue in the top-right of the window at any time
- The saved profile can also be amended and re-saved by using the *Profiles | Update Current Profile* option
- Or the profile deleted by use of the *Profiles | Delete Current* option
 - The profile setting can be refreshed (i.e. where view settings have been modified during viewing) by clicking on the *Refresh* button at the top-right of the Relationships Matrix window.

13.4.27.2 Creating/Deleting a Relationship between Elements

- Select *View | Relationship Matrix* from the main menu bar at the top of the screen
- Select the *Source* and *Target* packages from the tree structure (e.g. SIV and IAM, or UC)
- Select the required *Type* of element from each package (e.g. Class, Object, Activity, Package etc.)
- Select Link Type of *Relationship*:
- Set the Direction as *Source > Target* or *Target > Source* (NB: although there is a *Both* setting – this is a view only setting and cannot be used to create relationships)
- Right click in the square where two related elements intersect and select *Create New Relationship*.
- Bi-directional relationships must be completed as two separate actions (e.g. Source > Target and then as Target > Source – or vice versa - 'both' is a view-only setting)
 - Relationships can also be viewed or deleted in the elements own properties dialogue box using the Link tab for each element.
- Double-clicking the square displaying the relationship created in the matrix opens that relationships properties dialogue – from here you may add annotation regarding the relationship
 - NOTE: the same content for any annotation must be added to bi-directional relationships (i.e. Source > Target and Target > Source or vice versa) to be available in the *Both* view (one of the current quirks of EA is that different source and target combinations do not display annotations from the same root relationship when *Both* is selected)!

- *Hints and Tips:* to set up multiple relationships at the same time, hold the Ctrl key down and click in the required intersecting squares. Right-clicking on any one of the selected items leads to the option to *Create New Relationship*, which will then be done for all the squares selected (NB the relationships exist in only one direction).

Changing the Direction of the relationships at the top of the dialogue while the squares remain selected allows multiple relationships in the opposite direction to be created (however adding annotations has to be undertaken individually up for each relationship in each direction).

Alternately the matrix can also be populated in the following way:

- Select an element in the *Project Browser*
- Right-click *Add* then *Create Link*.
- Select Direction as Outgoing (or Incoming) and the Link Type as Relationship.
- Select the *Select Target Type* to that of your choice (e.g. *Class*).
- Chose the required element from the *Package* and *Name* list below and click *OK*.
 - The above then needs to be repeated for the opposite of the *Direction* already created (e.g. if *Outgoing* was previously selected then an *Incoming* needs creating / or vice versa) to create a bi-directional link.
 - Annotations can be added to the link using the properties dialogues Link tab by double clicking on the displayed link in question (again note that any annotations must be added to links in both directions)

13.4.27.3 Printing or Exporting a View of the Matrix

- To print the currently displayed view of the matrix:
 - Click on the *Options* button at the top-right of the Relationships Matrix window
 - Select the *Matrix | Print* options (NB: some matrices are very big and should be reviewed prior to printing – using the *Matrix | Print Preview* option)
- To export the currently displayed view of the matrix to a CSV (Comma Separated Value) file for external review
 - Click on the *Options* button at the top-right of the Relationships Matrix window
 - Select the *Matrix | Export to CSV,,,* option (NB: some matrices are very big and can be reviewed prior to exporting – using the *Matrix | Print Preview* option)
 - Choose an appropriate name and location for your export file and click *OK*

13.4.27.4 Relationship Matrix Display Options

The Relationship Matrix has a number of selection/display options to choose from which can be accessed by:

- Clicking on the *Options* button at the top-right of the Relationships Matrix window and selecting the *Options* choice from the drop-down menu displayed (Figure 59)
- Check the boxes against the following options:
 - **Include Source Children** - to recursively include child packages and contents under the *Source*
 - **Include Target Children** - to recursively include child packages and contents under the *Target*
 - **Include All Extended Meta Types** - to include elements that are extensions of a specified meta-type. For example, if there are Block elements (extending Class) in the package, selecting this option and specifying the type Class includes Class and Block elements, and any further derivatives of Block in the matrix.
 - **Sort Axes** - to ensure package elements display in alphabetical order
 - **Show Package Names** - to hide or show package names; useful for shortening the displayed text for packages that have long names
 - **Use Element Alias If Available** – displays an element's alias instead of its name, if one has been defined.
 - **Show Level Numbering If Available** – to reproduce level numbering in the Relationship Matrix, if it is turned on in the Project Browser.

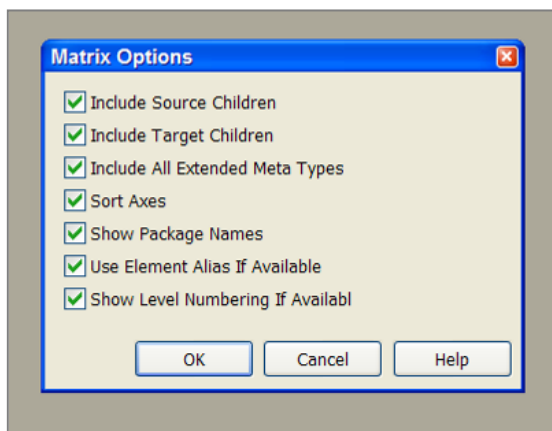


Figure 59: relationship matrix display options

13.4.28 Tracing Related Elements

- Where relationships have been created as described in §13.4.27, it is possible to bring all the related elements together in one diagram for ease of maintenance.
- Create a new diagram, drag on the element to trace
- Right click on the element, select Add/ Insert Related Elements (see Figure 60)
- Select the required number of levels (beware, selecting a lot of levels may be slow to run on a large model)
- Restrict the *Link Type* and *Element Type* if appropriate.

- Click *OK*
- **Note:** The diagram will not automatically update when further relationships are added elsewhere in the model. The diagram must be recreated/ refreshed by selecting the element, and repeating the process of Inserting Linked Elements.

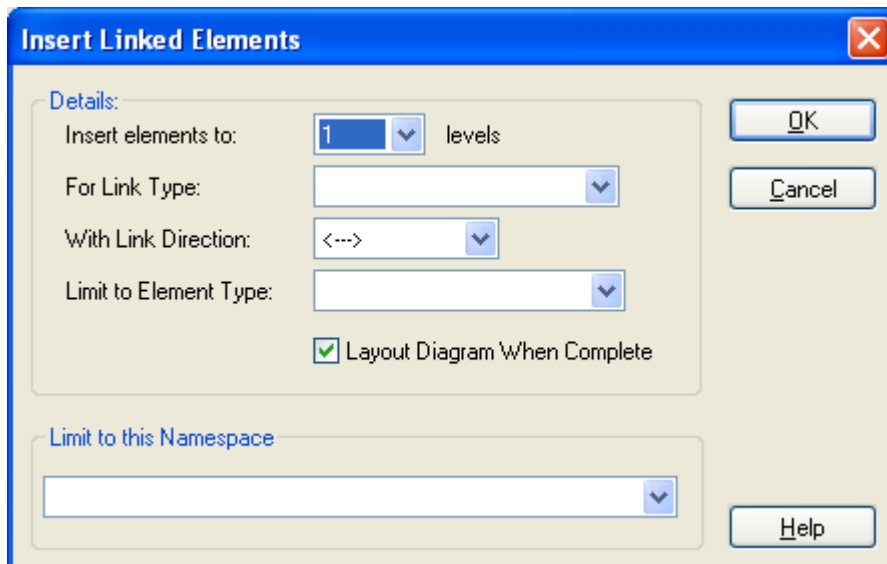


Figure 60: insert linked elements

13.4.29 Indicating Nominal Relationships

It may be necessary to indicate the existence of a nominal relationship between elements within a model; this can be achieved by creating a *trace* association with the stereotype set to <<nominal>> (this is a non-standard UML stereotype). A *Trace* is a type of *Dependency* association; however when used with the stereotype <<nominal>> it should be read as non-computable and should be displayed without directional arrows (*Direction: Unspecified*) - even if the elements concerned mutually rely on the same computation to achieve their content (if a computable or hierarchical dependency is required this should be indicated elsewhere by the use of other types of UML associations). A description of the nature of the *nominal* relationship between elements should be added to the dependency associations' properties dialogue (see Figure 61). Where a *nominal* trace is created it should not be taken to infer that any changes/risks/issues/hierarchies or other conditions relating to one element are applicable to any other elements linked with this type of association.

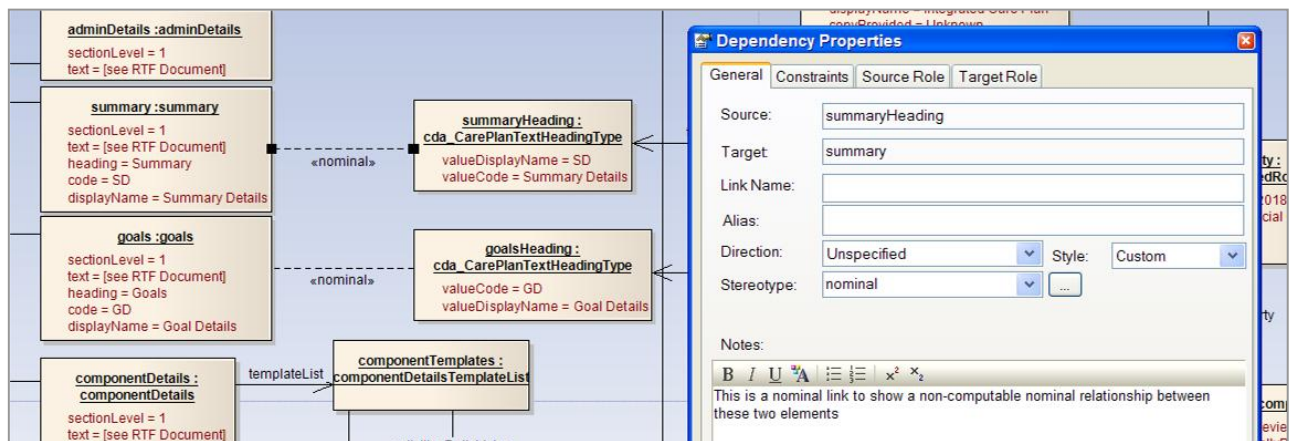


Figure 61: creating a nominal relationship

Whilst *Dependencies* between elements across different diagrams/packages may be reviewed using the Relationship Matrix (see §13.4.27), it is not possible to isolate those using different stereotypes and therefore the relationship matrix is not an effective means to identify these types of associations; however they may be reviewed from within a diagram or as part of an elements properties dialogue using the Link tab.

13.4.30 Linking Documents to the EA Model (native and HTML output)

- Create Document Artefact (from the Component toolbox), see Figure 62

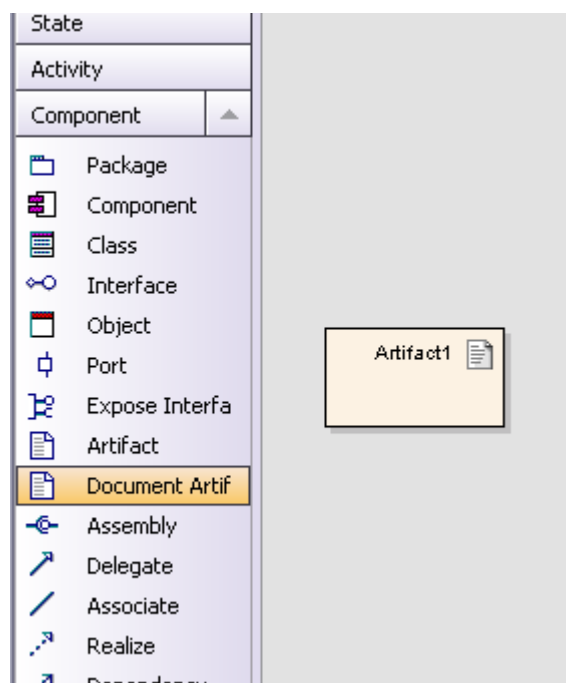


Figure 62: document artefact

- Place the created document artefact in it appropriate position within the project browser
- Double-Click on the created document artefact to open the blank document within the EA editor
- If you are prompted with a 'New Linked Document' dialogue click Cancel (NB this function allows documents to be created within the model, however it is anticipated that externally created documents are to be linked to the model, thus no template is required)
- Right Click on the blank document, select File, and then select Import...
- Locate and select the document to be linked (**NB** this **must** be a RTF document)
- Click Open

NB when the linked document is output to HTML the formatting and numbering of the document is liable to be changed.

13.4.31 Creating/Exporting a (Requirements) CSV File

1. Select the model or package to be output as a CSV file (using the project browser)
2. Select Project | Import/Export | CSV Import/Export (Ctrl-Alt-C) from the top menu – this opens the CSV Import/Export dialogue
3. Select the Edit/New button
4. Select an existing CSV specification or type in a new name for a new specification
5. Ensure the delimiter is set to a comma
6. Type in a name (and file location) for the output file
7. Set Export as the Default Direction
8. Set Requirement (if suitable) as the Default Type
9. Review the existing File specification items
10. If there are any specification items that should be removed – select the item and click the Remove Field button
11. To add new items from the Available Fields list, click on the item from the displayed list ...
12. and click the Add Field button (repeat 'add' & 'remove' steps as required)
13. When specification completed, save by clicking the Save and then the Close button
14. The specification dialogue will disappear and the CSV Import/Export dialogue will reappear – ensure the required specification name is now showing in the Specification box...
15. ...and that the output file is correct
16. Then click the Run button
17. To view the CSV output select the View File button

18. Click Close to exit this process

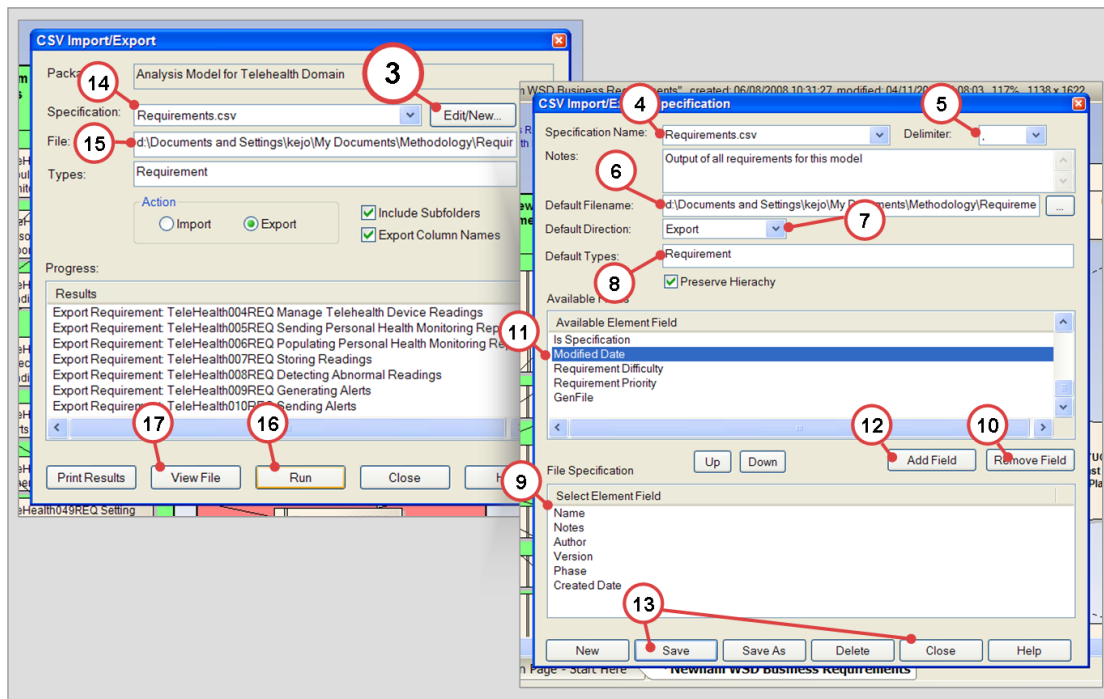


Figure 63: exporting CSV file

13.4.32 Creating Relative Paths to External Documents

- Use the 'Files' tab of the element's properties dialogue box as shown in Figure 64.

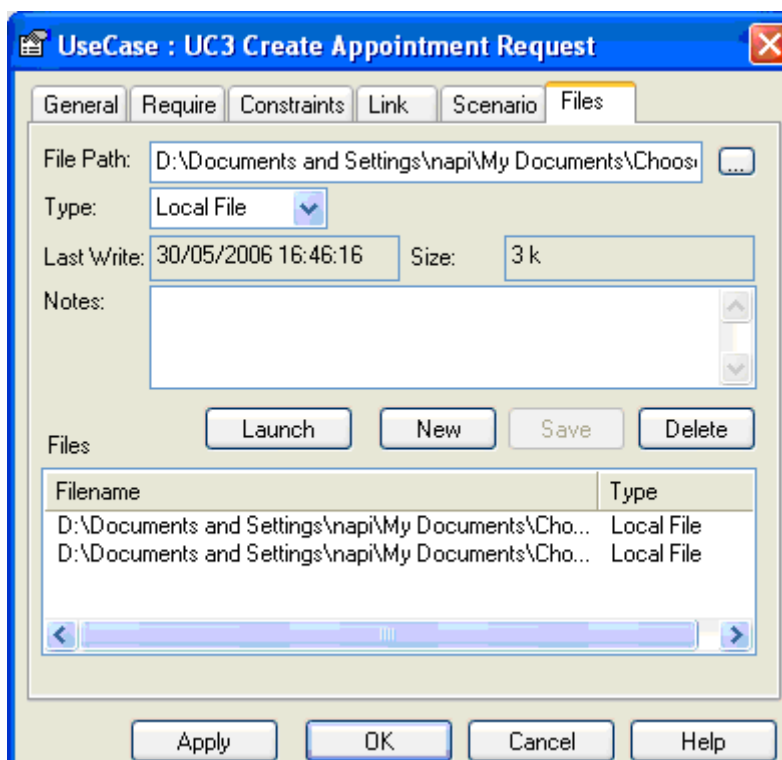


Figure 64: external files (a)

- By default, the paths to the files are absolute, not relative to the location of the EA file. This is not ideal where we wish to deliver the native version of the project, along with any supporting external files. It creates no issues however when we publish HTML documentation as the external files are copied into the HTML file structure and are accessed via the browser, this requires that the hyperlinked files option is selected for inclusion in the Generate HTML Report dialogue box (see Figure 65)

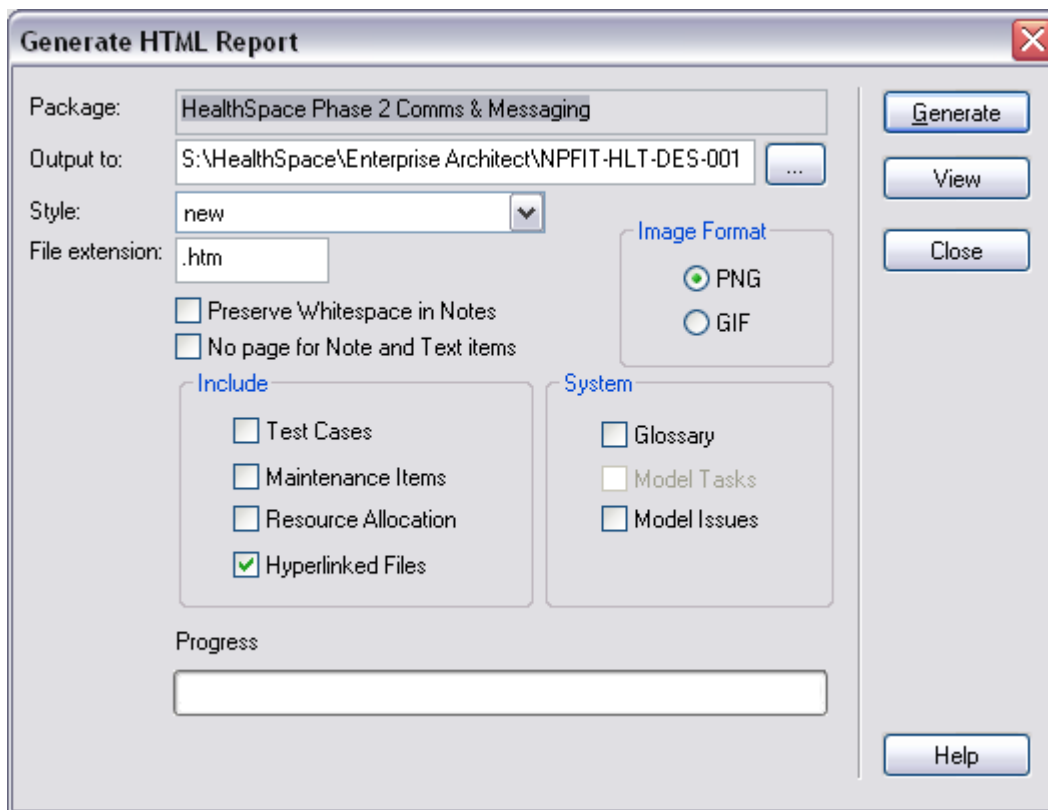


Figure 65: HTML report options

- In the example given above, the full path is:
D:\Documents and Settings\napi\My Documents\Choose and Book\3 Choose and Book Appointment Request Creation and Update\Create and Update Appointment Request Validation Matrix.htm
- We can make the path relative to the location of the EA file so that we can deploy the EA file and the external files anywhere and still be able to create HTML documentation correctly. (Note that an alternative approach is to store the files on a shared drive, but this does not permit a complete copy of the documentation to be stored in and retrieved from FileCM).
- To make the path relative, manually edit this in EA. In the File Path edit box, delete the first part of the file name, leaving just the directory (with no preceding backslash) and the file name. Press Save (see Figure 66).

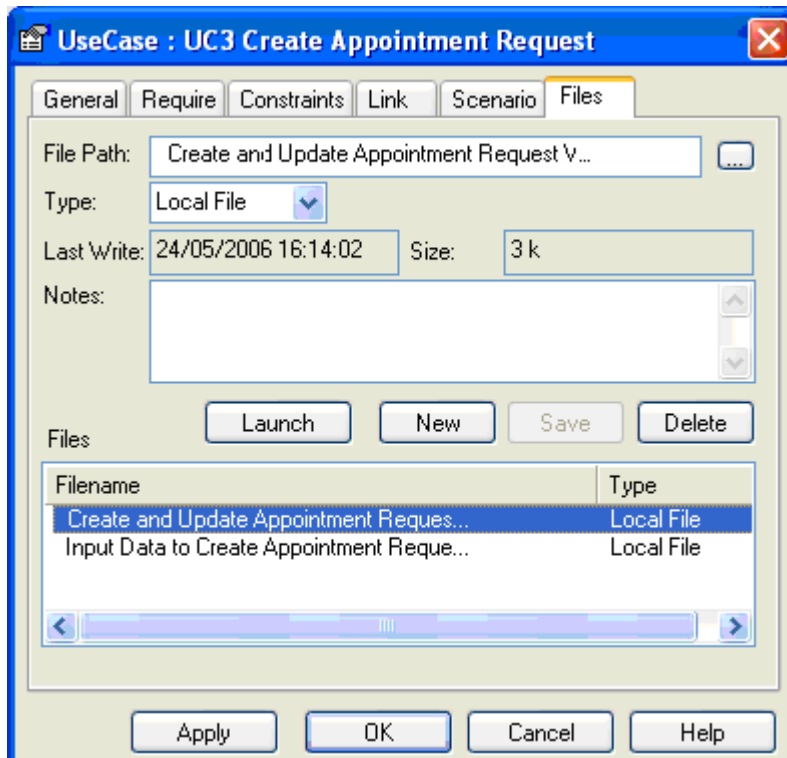


Figure 66: external files (b)

- The path is now: Create and Update Appointment Request Validation Matrix.htm
- **Storing in FileCM:** Zip the EA file together with the supporting files and store the resulting zip in FileCM.

13.4.33 Creating Hyperlinks to External Documents

- Right click on the background and choose 'Hyperlink' from the menu at 'Insert Element at Cursor'

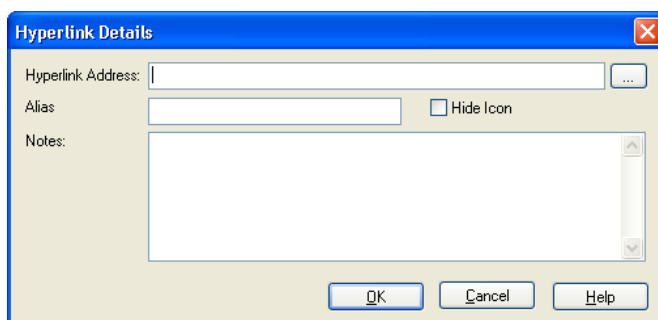


Figure 67: hyperlink details dialogue

- Click the browse button (ellipsis) and navigate to the required document

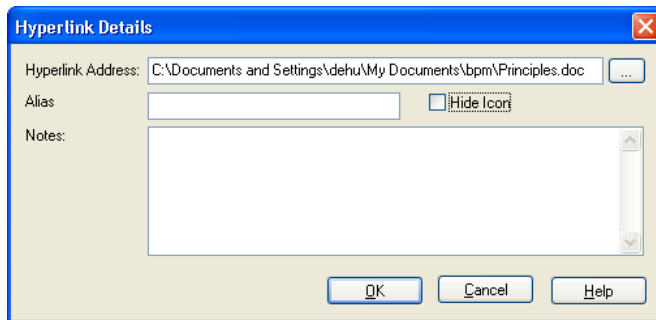


Figure 68: hyperlink address

- By default the path is absolute and must be made relative to allow distribution to reviewers. Delete the file path leave only the filename of the document.
- Add a name to the Alias box, which may be the name of the document or a more explanatory name and check the Hide Icon box.

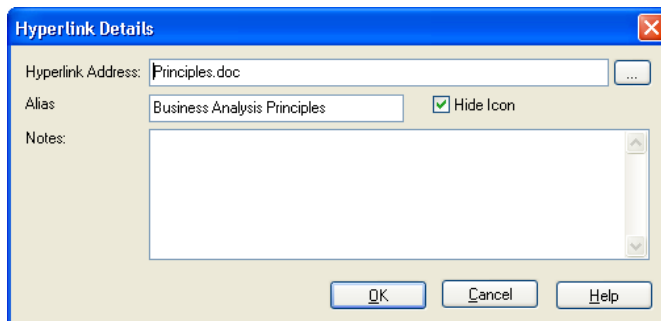


Figure 69: hyperlink alias

- Click OK and format the name as appropriate, (blue underlined for example)

13.4.34 Creating an Association Class

- From the toolbox, select the Class category, find the Association Class icon (Figure 70):

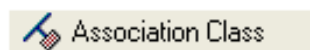


Figure 70: Association Class Icon

- Click once. Click on the source class in the diagram and hold the mouse down while you drag the line to the target element - release the mouse button.
- EA will draw the link and add the class. Give the class a suitable name when prompted by EA.
- The association has the same name as the class and should be given suitable source and target multiplicities. Note that by default an association with an association class is directed

13.4.35 Diff Facility

- EA has a 'diff' facility which allows two versions of a package to be compared, as shown below:
- First you need to create a baseline from which to compare:
 - Select the root package of the baseline model in the Project Browser

- Right click on it and choose the Package Control and then the Configure option
- In the dialogue that opens give the XMI file a name and a path (see Figure 71)

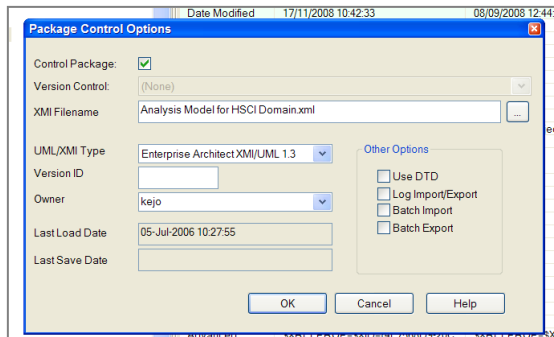


Figure 71 set baseline file

- Then you need to open the file you wish to compare with it (usually a newer version of the model)
 - Using the project browser select the same root package and right click on it
 - From the menu options choose Package Control and then Compare with XMI File
 - From the windows dialogue that opens up select the baseline XMI file and click Open
 - The process will start automatically from this point
- All the elements of the package are compared (see Figure 72),
- If you look through the model hierarchy you will see that items that have been changed are flagged in the right hand pane showing what has changed - the left hand pane shows more detail of the changes that have taken place.
- This facility will allow reviewers of complex models to identify changes when re-reviewing.
- The results of the comparison can be logged to an XMI file using the small icon menu bar at the top of the viewing pane,
- NOTE: the Compare Utility is only available in the Corporate and Professional versions of EA



It is possible to compare a package with either an XML export file from an earlier version of the EA model, or with a baseline held inside the current version of the model.

- Open the older EA model and select the package in the project browse
- Right click and select Import/Export → Export package to XML file (see §13.4.13)
- Save the XML file to a suitable location with a meaningful name.
- Open the newer EA model and select the package in the project browser.
- Right click and select Package Control → Compare with XML file.
- Select the appropriate XML file in the file Open dialog and hit the 'Open' button.
- The EA Compare Utility will open and analyse the model differences.

- Compare options:

- Page 146 of 211

- To set the compare options select 'Compare Options' from the small menu bar at the top of the Compare Utility window, the dialogue shown in Figure 73 will be displayed.

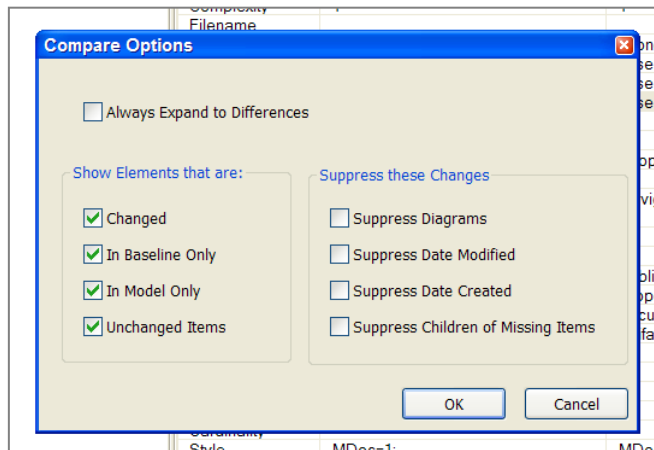


Figure 73: EA compare options

- To ignore the date modified and created, check the appropriate checkboxes.
- Suppress diagrams will ignore cosmetic changes such as the precise coordinates of an element in the diagram. Check this unless this information is of relevance to the reviewers.
- Shallow compare is useful when comparing very large packages (or indeed an entire model exported to XML) as it will speed up the initial comparison; only the top level elements are compared and then when each is clicked, a comparison of its child elements is performed.

Log to XML:

- The results of the comparison can be logged to an XML file. As above, select Log to XML from the small menu bar at the top of the Compare Utility window. Enter a suitable name. The resulting file will need some processing before it is legible in Word, HTML, etc as an amendment history.

Other Compare Utility Options:

- Right clicking an item in the left-hand pane of the utility also offers a number of other options, see Figure 74:

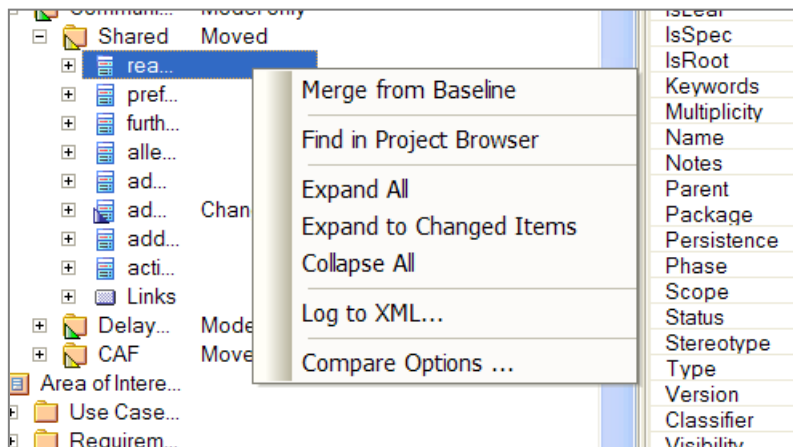


Figure 74: EA compare utility options

- The currently selected element may be located in the project browser.
- All elements below the currently selected one may be expanded, using Expand All.
- Expand to Changed Items usefully only expands those items which have been added, changed or deleted, making these easier to review.
- Collapse All collapses all elements below the currently selected one.

13.4.37 Creating Stereotypes

- To create a new stereotype, select the Settings menu
- Selects UML
- Select Stereotypes to give the stereotype dialogue (see Figure 75)
- The minimum input required is the stereotype name and the base class that it is applicable to. NB the enumerations available for the base class do not include 'attribute' – if an attribute stereotype is required this must be entered manually

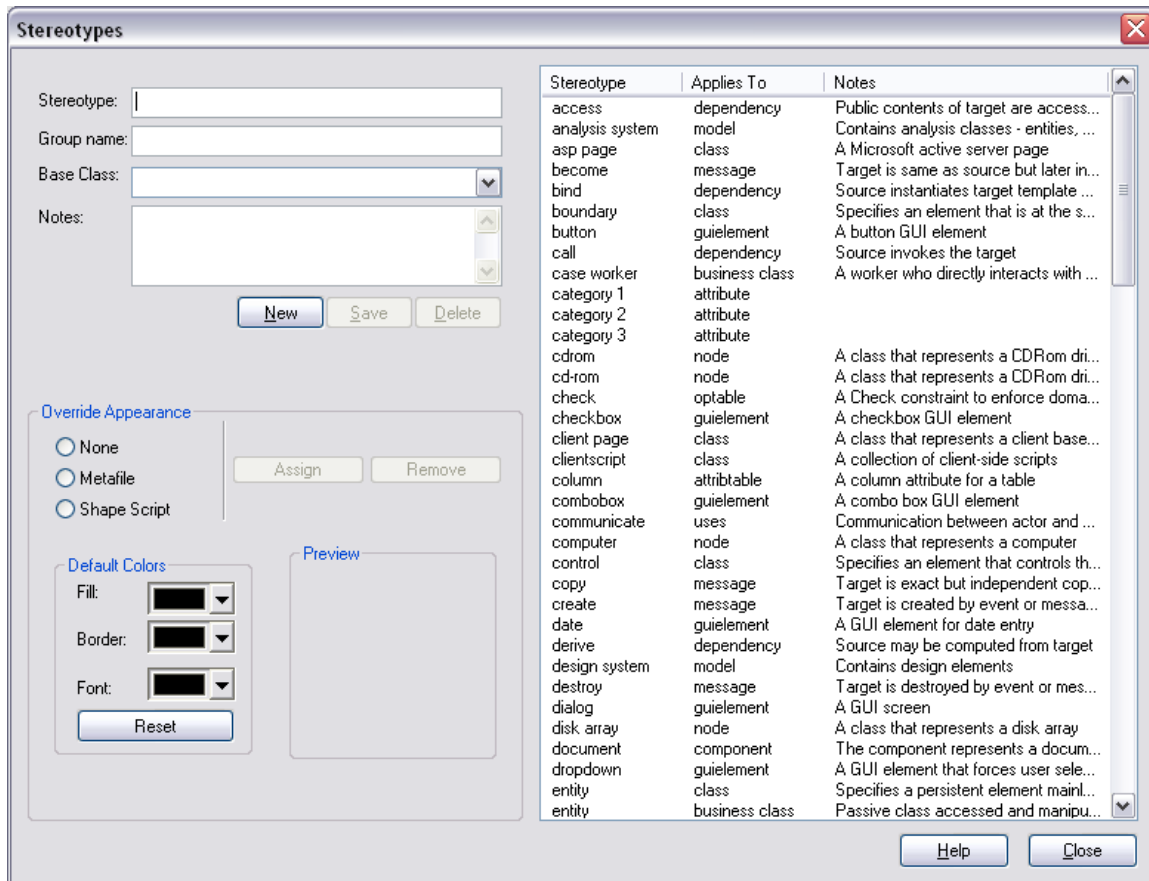


Figure 75: stereotype dialogue

13.4.38 Bookmarking Elements

- To create a new bookmark, select the elements to be bookmarked
- Select the Edit menu
- Select Bookmark Selected

The deletion of bookmarks can be achieved by repeating the above steps for a selected and bookmarked element. All bookmarks can be deleted from the model using the Clear All Bookmarks option from the Edit menu

N.B. The Bookmark symbol can be obscured by association symbols (e.g. generalisation, aggregation).

13.4.39 Searching for Bookmarked Elements

- Open the Search view (the keyboard shortcut is Ctrl + F)
- Select Find Bookmarked Elements in the Search field
- Run the search to identify all elements which are bookmarked

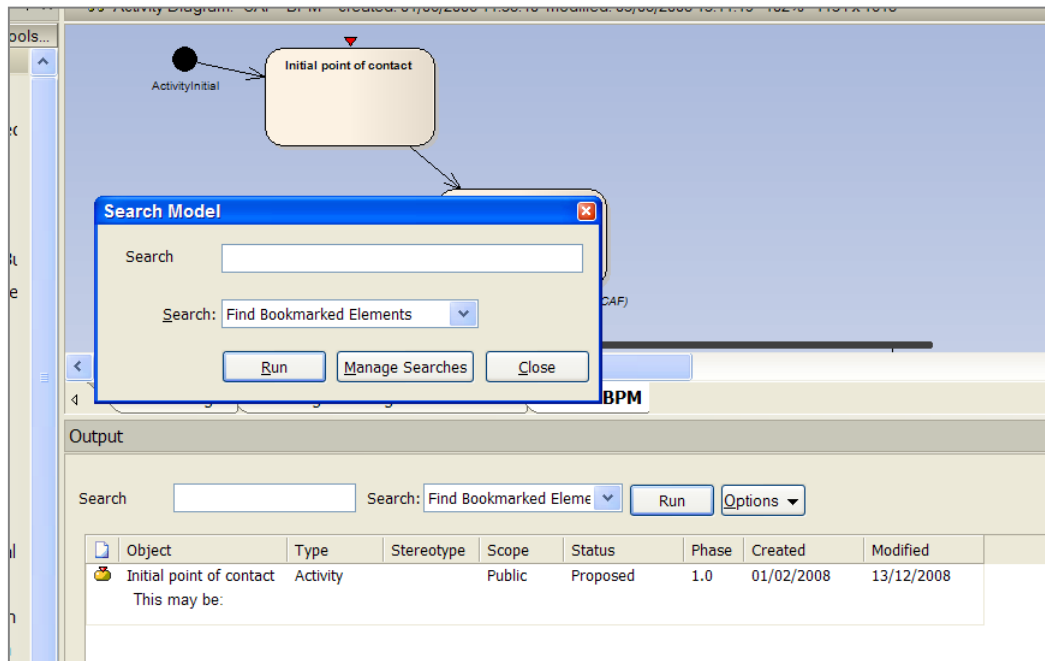


Figure 76: bookmark search view

13.4.40 Adding and Updating Glossary Entries

- Glossary entries can be entered in 2 ways:
 - glossary dialogue: this dialogue is accessed by Project – Documentation -- Glossary
 - System window: opened from the View menu. Once open, the issues are displayed within the Project Glossary tab
- Glossary entries can be displayed through the system window and/or the glossary dialogue

Glossary entries can be output into RTF form from the project issues dialogue. However, it is expected that these will be usually be output in HTML. NB to output in HTML form the 'Glossary' check box must be checked in the 'Generate HTML Report' dialogue.

13.4.41 Setting 'Rectangle Notation'

A number of elements in EA can have their appearance modified to 'rectangle notation'. This is particularly relevant to Activities. Modifying appearance to rectangle notation can be achieved by right-clicking the element, selecting Advanced then selecting Use Rectangle Notation.

13.4.42 Adding Tagged Values

'Tagged Values' can be associated with model elements to allow these values to be accessed and output in a more structured way than is possible with notes. The addition of a tagged value to an element involves right-clicking on the element, selecting 'Add' then selecting 'Tagged Value' to display the Tagged Value dialogue (and screen), see Figure 77. From this dialogue the tag type (e.g. 'Originator') can be selected/entered and the value associated with the tag can be input.

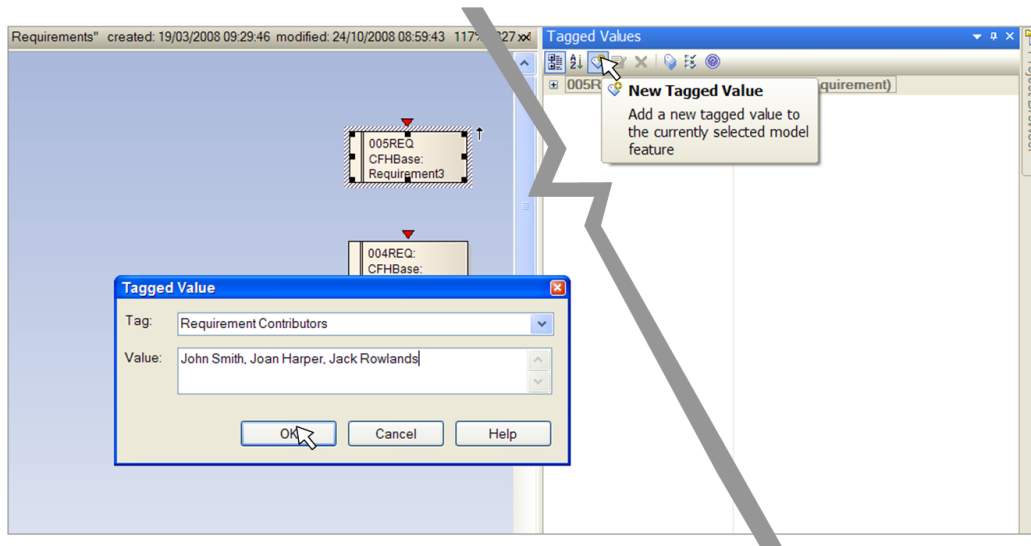


Figure 77: tagged value dialogue & window

To amend the default Tagged Values items, select the Settings | UML | Tagged Value Types menu.

To create a drop-down list of default values for a tagged value item enter the content into the Detail field within the Tagged Value Types Dialogue (see Figure 78).

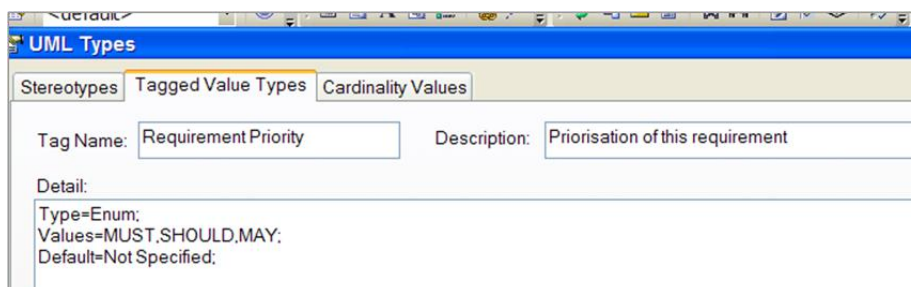


Figure 78: tagged value types dialogue

The tagged values associated with a selected element can be viewed in the Tagged Values window of EA (View | Tagged Values [Ctrl+Shift+6]).

13.4.43 Adding Risks

Risk items are recorded within an EA model by selecting *View | Project Management* (or Ctrl Shift 7). This opens the *Project Management* window (see Figure 79). A small icon driven menu is available just below the window banner and provides for *New*, *Save*, *Delete*, *Print* etc functions (see annotations in Figure 79). Click on an element / package in a diagram or from the *Project Browser* and complete the fields in the open window to record a risk item. The box to the left of the *Project Management* window displays a list of any already saved risk items captured against this element – click on these to show their content and editing if required (remembering to save the contents before leaving this item).

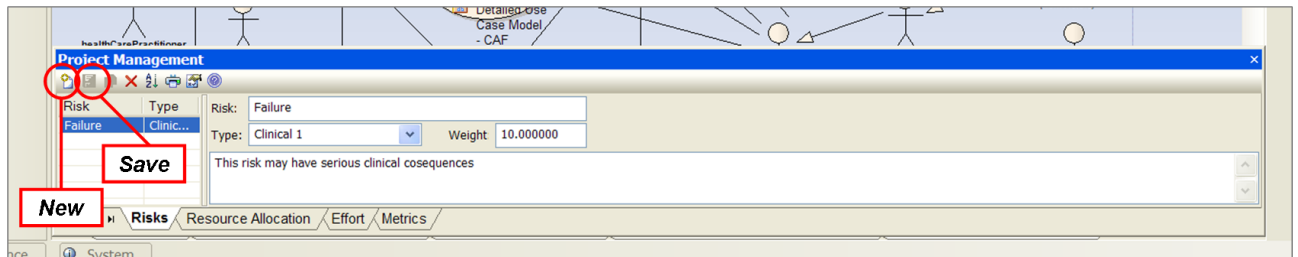


Figure 79: project management window

Creating a set of *Risk Type* headings for the drop-down list from which to classify the entry involves selecting the *Settings | Project Indicators* menu option and clicking on the *Risk* tab. From here a risk type can be set with a description and a standard weighting for this item type (typical risk headings could include: Clinical, Timescale, Organisational, Resources etc.).

13.4.44 Adding Issues

13.4.44.1 Issues Related to Model Elements/Structures

Issue items relating to model elements/structures are recorded within an EA model by selecting *View | Maintenance* (Alt 4) from the top main menu. This opens the *Maintenance* window (see Figure 80) from which the *Element Issues* tab is selected. This displays the fields used to define the issue - details, priority, owner etc. A small icon driven menu is available just below the window banner and provides for *New*, *Save*, *Delete*, *Print* etc functions (see annotations in Figure 80). The box to the left of the *Maintenance* window displays a list of any already saved issue items captured against this element – click on these to show their content and editing if required (remembering to save the contents before leaving this item).

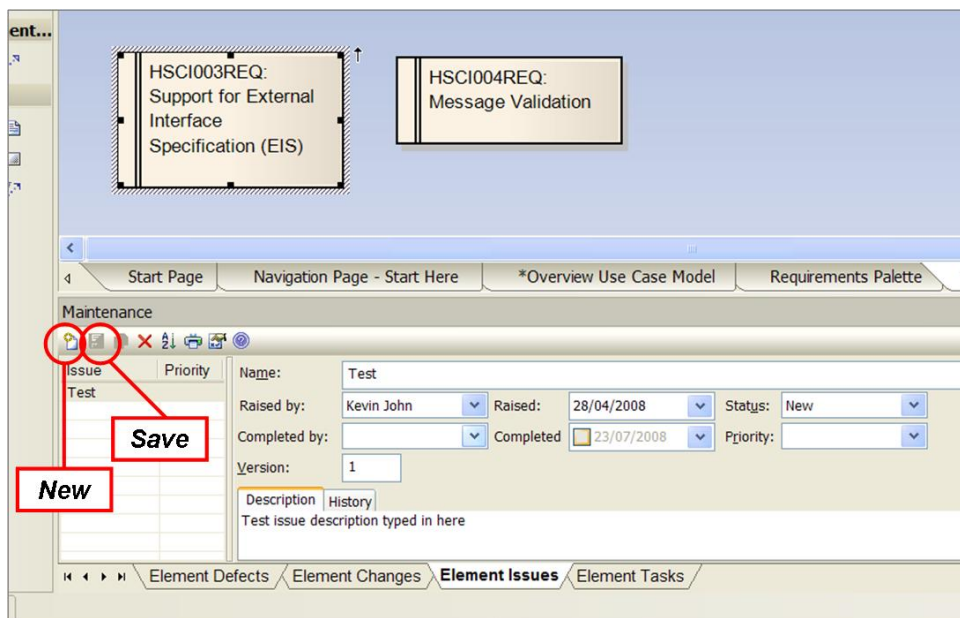


Figure 80: maintenance window

13.4.44.2 General Issues

General issue items (e.g. relating to the whole model) relating to model are recorded within an EA model by selecting *View | System* (Alt 2) from the main top main item.

This opens the *System* window. Double-clicking an empty line in the displayed window opens an issue dialogue (see Figure 81) in which to specify the issue, its description and priority etc. Any pre-existing issues can be selected (double-click) from the list displayed and edited in the *corresponding* dialogue; to save an amended entry use the *Apply* button (see Figure 81). The set of general issues can be output in RTF form by right-clicking in the list and selecting *Create RTF Report*.

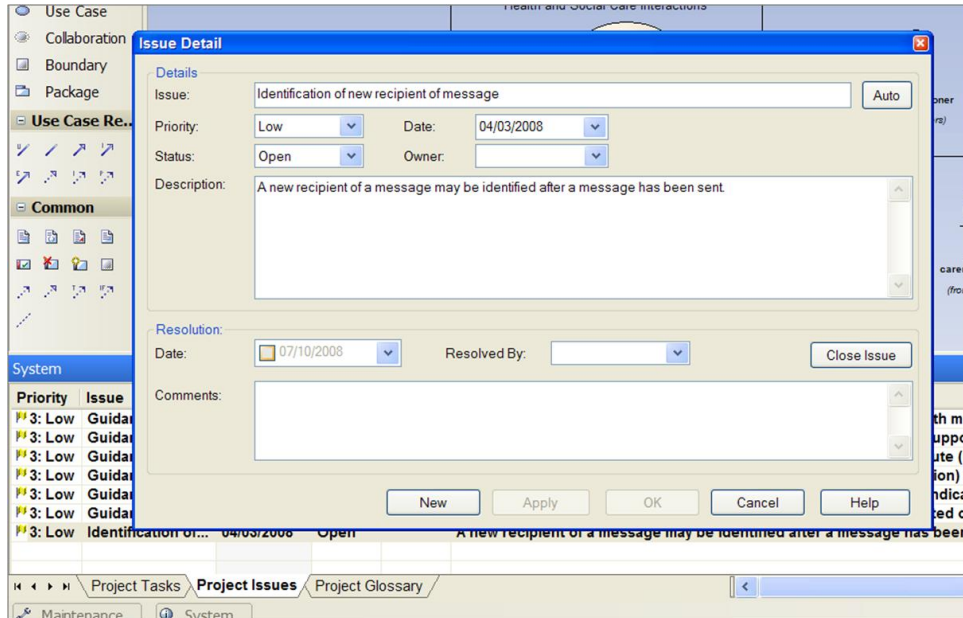


Figure 81: issue dialogue (system view)

13.4.45 Adding Changes

Change items are recorded within an EA model by selecting *View | Maintenance* (Alt 4) from the top main menu. This opens the *Maintenance* window (see Figure 80) from which the *Element Changes* tab is selected. This displays the fields used to define the change – name, description, priority, status etc. A small icon driven menu is available just below the window banner and provides for *New*, *Save*, *Delete*, *Print* etc functions (see annotations in Figure 80). The box to the left of the *Maintenance* window displays a list of any already saved change items captured against this element – click on these to show their content and editing if required (remembering to save the contents before leaving this item).

14 Appendix 2: Subversion and EA

14.1 Initial set up of Subversion (SVN) and EA

It is assumed that the user has local administration rights to their computer.

1. To access and use the repository:
2. Create a local directory for the repository. If SVN isn't installed follow the steps below, or jump to step 3.

Download tortoise SVN from <http://tortoisesvn.net/>

Right click on the local directory and select settings from the Tortoissvn context menu

Select Network and enter the settings Server Address = isa2lds, Port=8080, Timeout=60 secs and enter your own Username and Password details

3. Run SVN checkout

Set the URL of repository: to <https://svn.connectingforhealth.nhs.uk/svn/public/lra> (for LRA Project. This will be different for other projects)

Set the Checkout directory: to your local directory

Click OK to download the LRA folders and files

4. Run SVN Update on your LRA working copy to maintain up to date copies (but the update must also be performed in EA, see below under Using EA, to maintain an up to date EA project)
5. Download CollabNet Subversion Command-Line Client (NOT CollabNet Subversion Server and Client) from <http://www.open.collab.net/downloads/subversion/>
6. In an existing or new EA project. Note: there will be id clashes if your existing project contains version controlled folders from the previous EA model repository, in which case you will need to create a new project. Select Project | Version Control | Version Control Settings
7. Ensure "Save nested versioned controlled packages to stubs only is checked" and enter and save the following configuration:

Select Subversion unique id = svn_ea_lra_main path = <working copy root folder>\TRUNK\main\model (this is the shared development repository)

The workstation settings should be populated from your previous configuration

(...\Program Files\CollabNet Subversion Client\svn.exe/)

8. Close dialogue and right click on a node in project browser and select Package Control | Import Model Branch
9. Select the svn_ea_lra_main configuration - this should bring up the lra_main.eab model branch file, then select file and click ok
10. Select yes in resulting dialogue to import branch as root folder and wait for import to complete (this may take several minutes)

For more details on setup see

https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/doc/LRA_PE_Distributed_Authoring_With_EA.doc.

For specific LRA Project team work you will need to do the following:

Add the UML profile for LRA Technical Modelling resource package to your project as described below (from section 3.2 of

https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/infrastructure/doc/LRA_TMA-R1_Content_Description.doc)

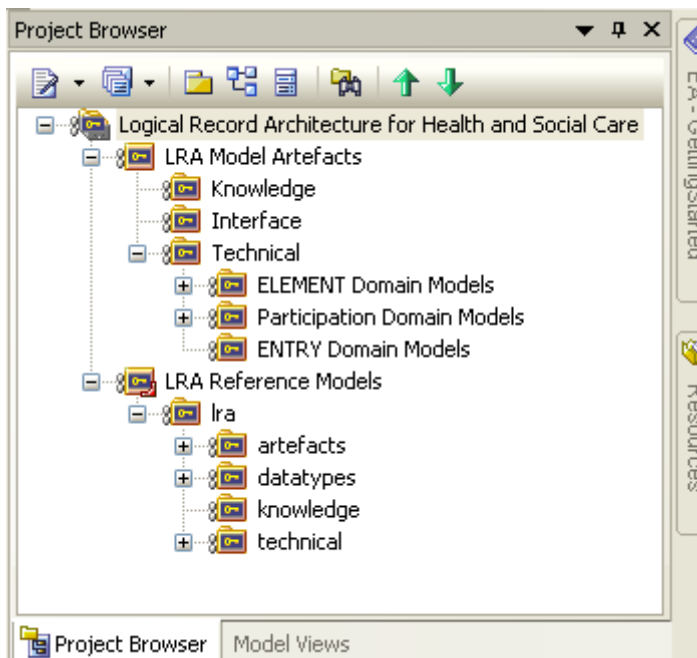
- update / checkout UML profiles folder and contents from: https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/model/uml_profiles/ to SVN working copy;
- select Resources tab in EA (next to Project Browser), right click on UML Profiles folder in resources tree and select Import Profile; and
- Navigate to UML profiles folder in file browser, multi-select all of the files and select open.

After the import has completed the UML Profiles folder will be updated with a UML Profile for LRA Technical Modelling subfolder so that ELEMENT model links are rendered correctly.

The profile package can be downloaded from

https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/model/uml_profiles/EA_UML_Profile-LRA_Technical_Modelling.xml.

You should now have an EA project that looks something like the screenshot below



The Logical Record Architecture model root node includes the following folders:

- \LRA Model Artefacts\Knowledge - root folder for developing Knowledge Model Artefacts (for use by Knowledge Modelling Team)

- \LRA Model Artefacts\Interface - root folder for maintaining shared (Interface) Model Artefacts
- \LRA Model Artefacts\Technical - root folder developing Technical Model Artefacts (for use by Technical Modelling Team).
- \Reference Models - is the namespace root of the reference model packages. The current thinking is that the Technical Model Infrastructure reference model classes should not be used directly within any of the \LRA Model Artefacts packages. The Knowledge Model reference model classes are located under Reference Models\lra\artefacts\knowledge

With regard to using the shared model repository it's up to the Knowledge team to decide what content is maintained under \LRA Model Artefacts\Knowledge and it is organised.

Note also that the project can have any number of additional model root nodes - either private or shared by members of the Knowledge Modelling team. For instance there is a separate model root node for shared access amongst the Production Environment Team.

****IMPORTANT** - if you add a new folder to version control using Package Control | Add branch to version control, it is important to ensure that you select the correct configuration (e.g. svn_ea_lra_main, depending on where the folder is located) otherwise EA will let you save it to the wrong model repository which would cause a great deal of confusion.

The EA environment provides access control and version management for ongoing model development as described in

https://svn.connectingforhealth.nhs.uk/svn/public/lra/TRUNK/prod_env/doc/LRA_PE_Distributed_Authoring_With_EA.doc.

If you need to export a snapshot of all or part of the repository at particular point in time then I would recommend you use the normal model export functionality (i.e. right click on package and select Import/Export | Export package to XML file [with "Enable full round trip" ticked]). You can then save the resulting output to FileCM, Subversion or wherever you choose. Be aware that to view the output in EA you will either need to strip the GUIDs on import or import it into a project that is not linked to the repository (otherwise id clashes will occur).

14.2 Using EA in an SVN environment

14.2.1 Updating you EA Model to the latest SVN copy

To download the latest changes to your local directory from within EA;

- Highlight the top node 'Logical Record Architecture for Health and Social Care'
- Right click | Package Control | Get All Latest | Click OK to accept default of 'Import changed files only (recommended)'
- All changed files are imported to you local machine




This only updates your local machine with the files used by EA, to update the entire repository directory use a standard SVN update.

14.2.2 Checking out

Each package is locked as default and must be checked out before it can be amended, in order to avoid edit conflicts.

To check out a package:

- Right click on a package | Package Control | Check Out...
- Package is checked out (Icons indicate version control status of each package), as below

	This package is version controlled and checked out to you, therefore you can edit the package.
	This package is version controlled and not checked out to you, therefore you cannot edit the package (unless you check the package out).
	This package is version controlled, but you checked it out whilst not connected to the version control server . You can edit the package but there could be version conflicts when you check the package in again.

Checking out a package only checks out that package and does not check out any sub-packages; each must be checked out individually.

14.2.3 Checking in

It is important that packages are checked back in as soon as amendments are complete to allow other members of the team to view the changes (and check out and amend, if necessary).

- Right click on a package | Package Control | Check In...
- An 'Add Comment' dialog box appears with a default comment of Date and Time but an informative comment should be made to indicate the type of amendments made to the package.
- Click OK and package is checked in (icon reverts to a locked directory folder)

14.2.4 Multiple check in

When checking in multiple packages it is possible to check all child packages with one procedure.

- Select the highest level package, even if it isn't checked out
- If multiple edits have been made, choose the top level 'LRA Model Artefacts' to ensure that packages aren't accidentally left checked out
- Right click on the package | Package Control | Check In Branch...'
- A dialog box with all checked out packages which are children of the chosen package is displayed
- Click All and all packages are selected (or highlight individual packages if required)
- An 'Add Comment' dialog box appears with a default comment of Date and Time but informative comments should be made to indicate the type of amendments made to each package.
- Click OK and the packages are checked in

14.2.5 Undo checkout

If amendments have been made which are incorrect and need to be discarded or a package has been checked out but no amendments have been made, the check out can be undone

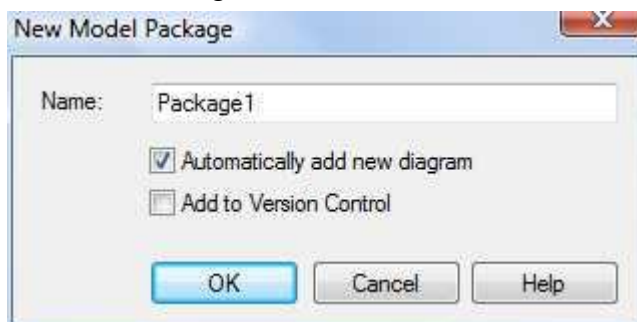
- Right click on the package | Package Control | Undo Check Out...
- A dialog box warning that all local modifications will be discarded is displayed
- Click Yes and model is reverted to pre check out status
- Note, there is no comment box as no changes have been made

14.2.6 Version control

Every package must be under version control and it is essential that the files are maintained under appropriate folders.

14.2.6.1 To add a new package

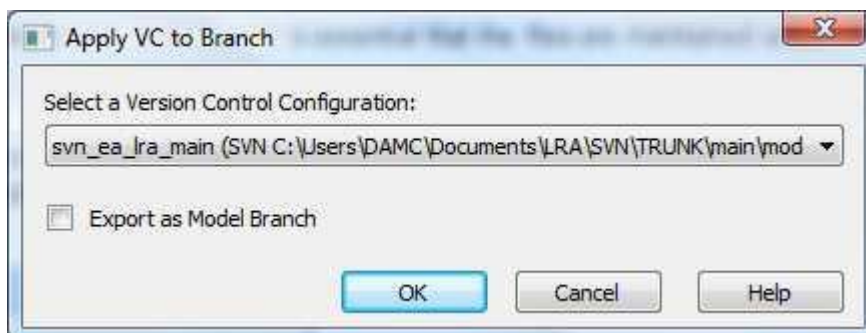
- Select the parent package
- Click the new package icon or right click on the parent package and choose Add | Add Package...



- Ensure that 'Add to Version Control' is unchecked
- If a new diagram is to be added immediately ensure that 'Automatically add new diagram' is checked.

14.2.6.2 Add the package to version control

- Select the package
- Right click and choose Package Control | Add Branch to Version Control...



- Ensure that Export as Model Branch is unchecked
- Click OK

The package is placed under version control and checked in. If packages are added to a package under version control they will not be automatically under version control.

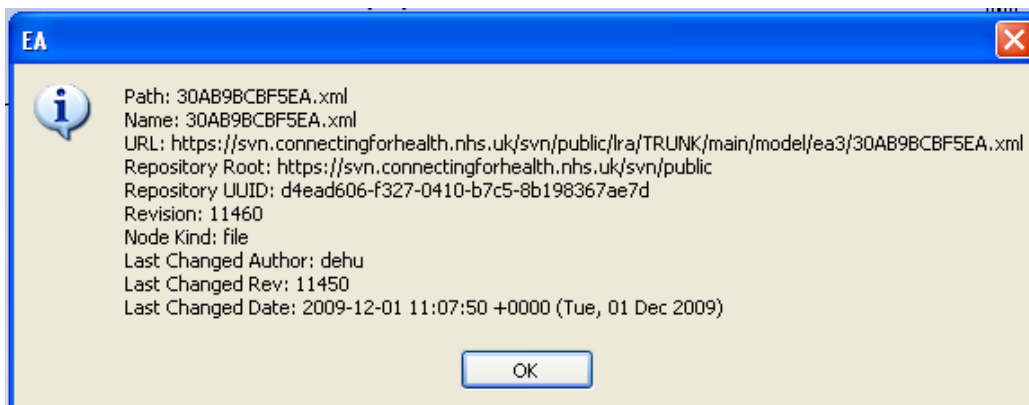
To add child packages, select the parent and follow the above procedure to add the entire branch to version control.

14.2.7 Checking out mismatches

The check out process within EA uses the SVN program's lock to prevent others from updating the same file but the two processes are separate in that EA logs a file as checked out and locks the file. So occasionally the two states may not match; EA may believe a file is checked out but SVN has no lock on the file.

In this situation it is necessary to use SVN directly to align the file's SVN status to match that within EA. The commonest situation is for the package to appear to be checked in within EA but an error occurs when trying to check it out because EA states it is already 'checked out'. The following procedure should align the relevant file:

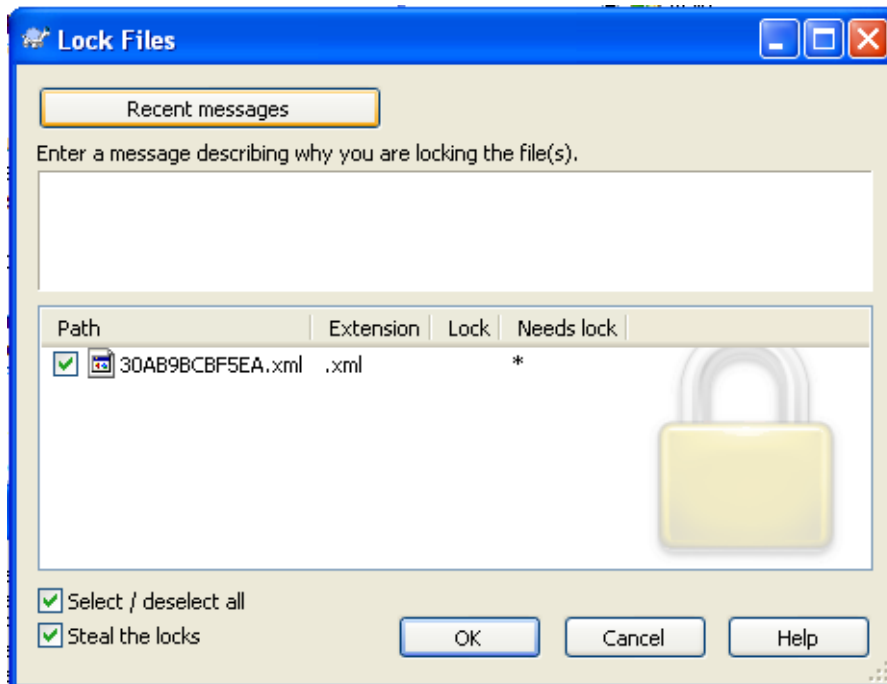
- Right click on the package and choose Package Control | File Properties and note the file name



- Navigate to the relevant file in your local directory

(If the lock is held by you then you should be able to simply right click and choose Tortoise SVN | Release lock, otherwise follow the steps below)

- Right click and choose SVN Get lock... (Which is either on the Context menu or under Tortoise SVN | Get lock..)



- Ensure that Steal the locks is checked and click OK, when it finishes click OK, this will break the lock held by the user.
- Right click and choose Tortoise SVN | Release lock, when it finishes click OK, this will release the lock and restore the file to the unlocked state to match that within EA (Checked in).
- Check that the file is able to be checked out within EA
- Enter an appropriate name and click OK

15 Appendix 3: Methodology for the Identification and Maintenance of Renal Candidate Data Elements

15.1 Background

The following information is taken from the experience gained during LRA Project for releases 2 and 3. It describes the LRA Analysis Phase development process. It concentrates on the transformation process of the IAM through to the DID and subsequent Candidate and Complete Data Elements. It also describes actions required to support collaborative working with the Terminology and Technical modelling teams who were both working on the same model using the same tool. Although many of these factors will not be present for LRA phase, elements of this approach were regarded as good practice and worth consideration for LRA Phase 1.

15.2 Introduction

Following the production of the Information Analysis Model (IAM), as part of the Logical Analysis for a particular domain, the Logical Record Architecture analysis needs to be undertaken

This analysis culminates in the production of the following artefacts:

- Domain Information Descriptions (DID)
- Data Element Definitions Palette comprising of
 - Candidate Data Elements
 - Completed Data Elements Definitions

During the review of each IAM there will be data items which have already been reviewed and modelled as interface artefacts but, in the initial stages, the majority will be new data items. These new data elements are candidate data elements which will be assessed and, if appropriate, modelled in the Interface section and further refined in the Technical section of the LRA model.

So the purpose of the LRA analysis is to produce a complete, coherent, traceable set of data items from an IAM, and ultimately a domain, which are linked to existing, defined items or identified as candidate items. As the candidate items are modelled within the technical section, the data items within the knowledge model must be updated to correctly reference their ultimate representation.

15.3 Analysis of Data items in DID

The IAM is a logical representation of the information requirements for a particular domain and has subsections within the overall IAM model.

The data is extracted from data sheets provided by stake holders and its representation in the IAM which are generally arranged in groups of data (forms) required for a particular function (e.g. Pre Haemodialysis session). It is proposed that

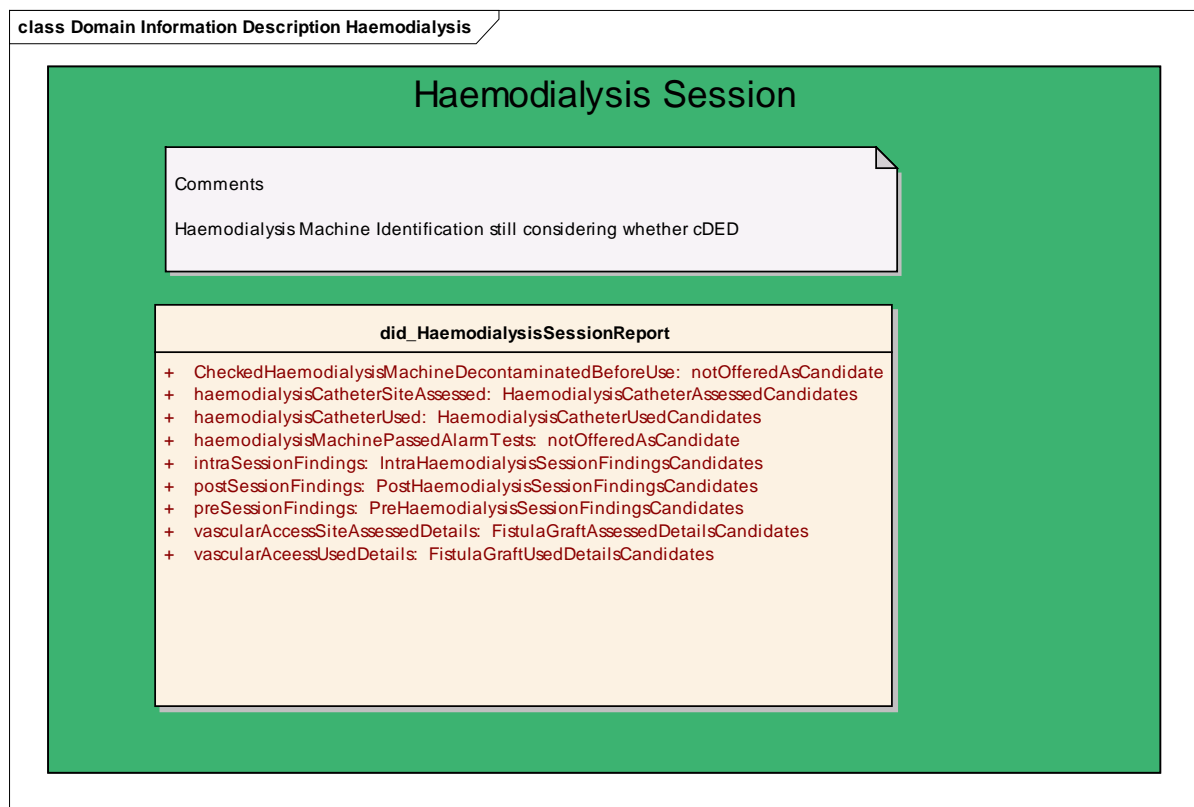
these groups are retained in the DIDs to allow traceability back to the original information requirements and modelled as classes in the DID and named in the format 'did_<data group name>' e.g. 'did_HaemodialysisSessionReport'

As an attribute is analysed, a check is made to see if it has already been identified. The data element can be found in one of two locations:-

- In the interface section as a defined Element
- Already requested as a candidate Entry

If the attribute has already been identified in the interface section then this is included in the class and the 'Type' chosen from the drop down list to link to the previously defined data item e.g. target dry weight: TargetDryBodyWeightObservation

There will be duplicates within the same pathway because some data items are required a number of times in the care process but it is useful to allow this in the DID tables and define the item as a 'Type' of the previously identified data item as this will strengthen its claim as a valid cded.



If an attribute has been previously identified as a candidate data item it is included in a Candidates class, for example PostHaemodialysisSessionFindingsCandidates, and linked to the appropriate candidate data item.

class Domain Information Description Haemodialysis

Haemodialysis Session Candidate Data Items

PostHaemodialysisSessionFindingsCandidates

- + /actualLengthOfHaemodialysisSession: cded_ActualLengthOfHaemodialysisSession
- + actualLengthOfHaemodialysisTreatment: cded_ActualLengthOfHaemodialysisTreatment
- + dialysateFlowRate: cded_DialysateFlowRate
- + ECGDerivedHeartRate: cded_ECGHeartRate
- + haemodialysisSessionStopDateAndTime: cded_HaemodialysisSessionStopDateAndTime
- + haemodialysisUltrafiltrationVolume: cded_HaemodialysisUltrafiltrationVolume
- + postHaemodialysisAssessment: cded_PostHaemodialysisAssessment
- + postHaemodialysisBloodSugar: cded_PostHaemodialysisBloodSugar
- + postHaemodialysisCapillaryRefillTime: cded_PostHaemodialysisCapillaryRefillTime
- + postHaemodialysisComment: cded_PostHaemodialysisComment
- + postHaemodialysisDiastolicBloodPressure: cded_PostHaemodialysisDiastolicBloodPressure
- + postHaemodialysisOxygenSaturation: cded_PostHaemodialysisOxygenSaturation
- + postHaemodialysisPulseRate: cded_PostHaemodialysisPulseRate
- + postHaemodialysisSystolicBloodPressure: cded_PostHaemodialysisSystolicBloodPressure
- + postHaemodialysisTemperature: cded_PostHaemodialysisTemperature
- + postHaemodialysisWeight: cded_PostHaemodialysisWeight
- + volumeOfBloodProcessedDuringHaemodialysis: cded_VolumeOfBloodProcessedDuringHaemodialysis

This class is referenced by the parent class as for example PostSessionFindings: PostHaemodialysisSessionFindingsCandidates see above.

Any data items which have not been previously modelled must be entered as candidate data items, see below for details, and then linked within as above.

15.4 Creation of Candidate Data definitions

Following the analysis of the data requirements for a particular section of the domain (e.g. Home Haemodialysis Initial Assessment) a diagram is created to represent the candidate data definitions. These are usually arranged within Boundaries to reflect the structure from the data sheets provided by stake holders

The candidate data definitions are also added to a diagram that contains all the items from that domain

The class is named appropriately to be as specific as possible. A comprehensive description of the candidate is required in the notes section of the class.

class Home Haemodialysis Initial Assessment																							
<h3>Physical Ability And Motivation Candidates</h3> <table border="1"> <tr> <td>cded_PatientsPhysicalAbilityToPerformHaemodialysis</td> </tr> <tr> <td>+ Value: CS [1..*]</td> </tr> </table> <table border="1"> <tr> <td>cded_PatientsMotivationAndCommitmentForHomeHaemodialysis</td> </tr> <tr> <td>- Value: CS</td> </tr> </table> <table border="1"> <tr> <td>cded_CarerAssessedToPerformHaemodialysis</td> </tr> <tr> <td>+ Value: ED.Text [0..1]</td> </tr> </table> <table border="1"> <tr> <td>cded_CarersPhysicalAbilityToPerformHaemodialysis</td> </tr> <tr> <td>+ Value: CS [0..*]</td> </tr> </table> <table border="1"> <tr> <td>cded_CarersMotivationAndCommitmentForHomeHaemodialysis</td> </tr> <tr> <td>+ Value: CS</td> </tr> </table> <table border="1"> <tr> <td>cded_PhysicalAbilityAndMotivationComment</td> </tr> <tr> <td>+ Value: ED.Text [0..1]</td> </tr> </table>	cded_PatientsPhysicalAbilityToPerformHaemodialysis	+ Value: CS [1..*]	cded_PatientsMotivationAndCommitmentForHomeHaemodialysis	- Value: CS	cded_CarerAssessedToPerformHaemodialysis	+ Value: ED.Text [0..1]	cded_CarersPhysicalAbilityToPerformHaemodialysis	+ Value: CS [0..*]	cded_CarersMotivationAndCommitmentForHomeHaemodialysis	+ Value: CS	cded_PhysicalAbilityAndMotivationComment	+ Value: ED.Text [0..1]	<h3>Haemodialysis Status</h3> <table border="1"> <tr> <td>cded_StableOnHaemodialysis</td> </tr> <tr> <td>+ Value: CS</td> </tr> </table> <table border="1"> <tr> <td>cded_SignificantConcomitantDisease</td> </tr> <tr> <td>+ Value: CS</td> </tr> </table> <table border="1"> <tr> <td>cded_OverallAdherenceToHaemodialysisTreatmentRegimen</td> </tr> <tr> <td>+ Value: CS</td> </tr> </table> <table border="1"> <tr> <td>cded_GoodFunctioningVascularAccess</td> </tr> <tr> <td>+ Value: CS</td> </tr> </table> <table border="1"> <tr> <td>cded_HaemodialysisStatusForHomeHaemodialysisComment</td> </tr> <tr> <td>+ Value: ED.Text [0..1]</td> </tr> </table>	cded_StableOnHaemodialysis	+ Value: CS	cded_SignificantConcomitantDisease	+ Value: CS	cded_OverallAdherenceToHaemodialysisTreatmentRegimen	+ Value: CS	cded_GoodFunctioningVascularAccess	+ Value: CS	cded_HaemodialysisStatusForHomeHaemodialysisComment	+ Value: ED.Text [0..1]
cded_PatientsPhysicalAbilityToPerformHaemodialysis																							
+ Value: CS [1..*]																							
cded_PatientsMotivationAndCommitmentForHomeHaemodialysis																							
- Value: CS																							
cded_CarerAssessedToPerformHaemodialysis																							
+ Value: ED.Text [0..1]																							
cded_CarersPhysicalAbilityToPerformHaemodialysis																							
+ Value: CS [0..*]																							
cded_CarersMotivationAndCommitmentForHomeHaemodialysis																							
+ Value: CS																							
cded_PhysicalAbilityAndMotivationComment																							
+ Value: ED.Text [0..1]																							
cded_StableOnHaemodialysis																							
+ Value: CS																							
cded_SignificantConcomitantDisease																							
+ Value: CS																							
cded_OverallAdherenceToHaemodialysisTreatmentRegimen																							
+ Value: CS																							
cded_GoodFunctioningVascularAccess																							
+ Value: CS																							
cded_HaemodialysisStatusForHomeHaemodialysisComment																							
+ Value: ED.Text [0..1]																							

The only attribute required for the class is a value attribute. The value type is selected from the drop down list which have been created in the LRA Reference Models using ISO 21090 data types (LRA_21090_Datatypes_Specification.doc).

Multiplicity is recorded in the 'Detail' tab of the Attribute. Units and permissible values are recorded in the Attribute Notes.

When producing Candidate Data Element Definitions, ensure that these are created before referencing them as a 'type' in an attribute of a Candidates class (in the DID), otherwise any changes made to the cded Class name may not link or update correctly to the attribute type

These candidate data definitions need to be picked up by the technical team and matched to existing definitions or modelled as new

15.5 Maintenance of the DID and Data Elements Definition Palette

15.5.1 Candidate Data Elements

When candidate data elements are accepted by the technical team and defined as interface artefacts they remain in the candidate data definitions.

Both these sections can be regarded as dynamic. As candidate elements are modelled or matched the following procedure needs to be undertaken

15.5.1.1 Procedure for Review and Acceptance of Candidate Data Elements

The following procedure describes the placement of a new Candidate Data Element Description (cDED), its review and its subsequent model completion

As new cDEDs are identified, they are placed in the Renal Candidate Data Definitions package and appropriate diagram. Where useful for clarity, they are placed within a descriptive boundary.

A Collaborative Communication Log has been created to enable members of the knowledge modelling, technical modelling and terminology teams to obtain clarification, share ideas and update the modelling status of each data element content requirement.

This log has been created in Enterprise Architect within the Logical Record Architecture for Health and Social Care.LRA Model Artefacts.Knowledge.LRA Analysis.Data Element Definitions Palette.Candidate Data Elements.cDED Status - <Domain> package (where <Domain> specifies the relevant knowledge domain).

Within this log each Candidate Data Element Definition specified in the knowledge space will have an entry. Each log entry is modelled within Enterprise Architect as an Issue. The Issues are viewed as an element list within a Maintenance Diagram as per the screenshot below:

Name	Alias	Status	Difficulty	Priority	Pha...	Ver...	Author	Created	Modified
cDED_HaemodialysisTreatmentType	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 08:...	15/10/2009 09:...
cDED_OralFluidRestriction	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 08:...	15/10/2009 08:...
cDED_HaemodialysisTreatmentFrequency	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	14/10/2009 15:...	15/10/2009 08:...
cDED_DurationOnHaemodialysis	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 09:...	15/10/2009 09:...
cDED_TargetDryWeight	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 09:...	15/10/2009 09:...
cDED_HaemodialysisBloodFlowRate	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 09:...	15/10/2009 09:...
cDED_DialyserName	Technical	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 09:...	15/10/2009 09:...
cDED_DialysateName	Knowledge	Proposed	Medium	Medium	1.0	1.0	peot1	15/10/2009 09:...	15/10/2009 09:...

Each Issue has a series of properties which can be edited in the Issue dialogue (see screenshot below):

The screenshot shows a window titled 'Issue' with a 'Properties' tab selected. The 'Short Description' field contains 'cded_OralFluidRestriction'. The 'Alias' field contains 'Technical'. The 'Status' dropdown is set to 'Proposed', and the 'Type' dropdown is set to 'Functional'. The 'Difficulty' dropdown is set to 'Medium', and the 'Phase' field contains '1.0'. The 'Priority' dropdown is set to 'Medium', and the 'Version' field contains '1.0'. The 'Author' dropdown is set to 'peot1', and the 'Last Update' field contains '15/10/2009'. The 'Key Words' field is empty, and the 'Created' field contains '15/10/2009'. The 'Notes' field is empty and has a rich text editor toolbar above it. At the bottom right are 'OK', 'Cancel', and 'Help' buttons.

Within the LRA Collaborative working approach these properties are used as follows:

Property name	Description
Short Description	The name of the candidate data element definition.
Alias	<p>A free-text field to describe the team with which the current responsibility for this candidate data element definition resides. Possible values are:</p> <p>Knowledge</p> <p>Technical</p> <p>Terminology</p> <p>(or a comma separated combination of the above)</p> <p>This field shall only be blank if the value in the status field is Approved.</p>
Status	<p>Describes the current modelling status of this candidate data element definition. The available states are used as follows:</p> <p>Approved – all modelling is complete. The value in the Alias field shall be blank.</p> <p>Implemented – technical modelling is complete but capturing fulfilment of knowledge requirements is outstanding. The value in</p>

	<p>the Alias field shall be Knowledge.</p> <p>Mandatory – not used.</p> <p>Proposed – discussion / clarification ongoing. Any allowable value in the Alias field may be used.</p> <p>Validated – not used.</p>
Difficulty	Not used. Auto-populated by EA but ignored.
Priority	Not used. Auto-populated by EA but ignored.
Author	Auto-populated by EA. Do not edit.
Key Words	Not used.
Type	Not used. Auto-populated by EA but ignored.
Phase	Not used. Auto-populated by EA but ignored.
Version	Not used. Auto-populated by EA but ignored.
Last Update	Auto-populated by EA. Do not edit.
Created	Auto-populated by EA. Do not edit.
Notes	<p>A free-text field to capture ongoing analysis comments and requests for further clarification on this candidate data element definition. Each comment is prefixed with the initials of the author of the comment.</p> <p>Once technical modelling is complete (i.e. the value in the Alias and Status fields are Knowledge and Implemented respectively) a comment should be entered to describe the Interface model that fulfils the requirements of this candidate data element definition.</p>

15.5.1.2 Creating a report

Select appropriate diagram e.g. 'Haemodialysis Session Candidate Data Definitions'

Right click on an empty area of diagram and select Show Diagram as Element List

From Element menu bar (just above actual list, not the drop down Element list)
change drop down menu from All to Class

1. Select all classes by selecting top element and shift click on bottom element
2. From Element menu bar select the Rich Text Report icon (next to Print icon)
3. Root Element should display Selected Elements
4. Choose an output file name and path
5. Choose an appropriate template e.g. *Element list - constrained* from *Use Template* drop down list
6. Click on *Generate*
7. The report can be viewed using the *View* button, if required

15.5.2 DID

The aim is to acquire definitions for all the candidate data elements identified in the DID and link the data item to its technical realisation.

This is achieved by:

- Checking for fully defined data definitions in the cDED Status section
- Identifying the corresponding candidate data definitions
- Changing the 'Type' from a 'cded' type to the new type (modelled ELEMENT), see below

15.5.2.1 *Changing the Type from a 'cded' type to the new type (Modelled ELEMENT)*

When a candidate has been modelled, the corresponding DID attribute 'type' needs to be changed from the candidate CDED to the modelled <<ELEMENT>> and moved from the candidate class to the appropriate DID class.

This is achieved by the following procedure:

- Find candidate element in project browser and then find in diagrams and select its location in the interface space (usually as a CompositeStructure Diagram) and click Open. The cded will be associated with the modelled element in the diagram
- Find the modelled element in the Project Browser and make a note of its position in the browser
- Navigate to the appropriate DID diagram and find the entry in the candidate class
- Open the attribute in the DID and replace the 'Type' with the associated ELEMENT, finding it via the browser button (not using the drop down)
- Move the attribute from the candidate class to the DID class using the Project Browser

The diagram below shows the did_HaemodialysisCatheterInsertionAndRemovalRecord and HaemodialysisCatheterInsertionAndRemovalRecordCandidates classes following some technical modelling

class Domain Information Description Haemodialysis

Haemodialysis Catheter

did_HaemodialysisCatheterInsertionAndRemovalRecord

- + anaestheticUsedInCreationOfDialysisAccess: HaemodialysisAccessAnaesthesia
- + haemodialysisCatheterComplication: HaemodialysisCatheterComplication
- + haemodialysisCatheterDressingPostInsertion: HaemodialysisCatheterDressing
- + haemodialysisCatheterDressingPostRemoval: HaemodialysisCatheterDressing
- + haemodialysisCatheterGaugeInserted: HaemodialysisCatheterGauge
- + haemodialysisCatheterInsertionAccessSite: HaemodialysisCatheterInsertion
- + haemodialysisCatheterInsertionAccessSiteLaterality: HaemodialysisCatheterInsertion
- + haemodialysisCatheterInsertionAndRemoval: HaemodialysisCatheterInsertionAndRemovalRecordCandidates
- + haemodialysisCatheterInsertionComment: HaemodialysisCatheterInsertionComment
- + haemodialysisCatheterInsertionDate: HaemodialysisCatheterInsertion
- + haemodialysisCatheterInsertionEffectivenessOutcome: HaemodialysisCatheterEffectivenessOutcome
- + haemodialysisCatheterInsertionGuidanceTechnique: HaemodialysisCatheterInsertion
- + haemodialysisCatheterInsertionProcedureComplications: HaemodialysisCatheterInsertionProcedureComplication
- + haemodialysisCatheterInsertionReason: HaemodialysisCatheterInsertionReason
- + haemodialysisCatheterInsertionType: HaemodialysisCatheter
- + haemodialysisCatheterLengthInserted: HaemodialysisCatheterLength
- + haemodialysisCatheterLumensLockedWith: HaemodialysisLumensSolution
- + haemodialysisCatheterPostInsertionFlow: HaemodialysisCatheterPostInsertionFlow
- + haemodialysisCatheterRemovalAccessSite: HaemodialysisCatheterRemoval
- + haemodialysisCatheterRemovalAccessSiteLaterality: HaemodialysisCatheterRemoval
- + haemodialysisCatheterRemovalComment: HaemodialysisCatheterRemovalComment
- + haemodialysisCatheterRemovalDate: HaemodialysisCatheterRemoval
- + haemodialysisCatheterRemovalProcedureComplications: HaemodialysisCatheterRemovalProcedureComplication
- + haemodialysisCatheterRemovalReason: HaemodialysisCatheterRemovalReason
- + haemodialysisCatheterRemovalType: HaemodialysisCatheterRemoval
- + haemodialysisCatheterSerialNumber: notOfferedAsCandidate
- + suturesInsertedOnHaemodialysisCatheterInsertion: HaemodialysisCatheterSutures
- + suturesInsertedOnHaemodialysisCatheterRemoval: HaemodialysisCatheterSutures

class Domain Information Description Haemodialysis

Haemodialysis Catheter Candidate Data Items

HaemodialysisCatheterInsertionAndRemovalRecordCandidates

- + haemodialysisCatheterComplicationDiagnosisDate: cded_DateOfDialysisAccessComplication
- + haemodialysisCatheterInsertionProcedureOutcome: cded_HaemodialysisCatheterInsertionProcedureOutcome
- + imagingUsedToCheckHaemodialysisCatheterPosition: cded_ImagingUsedToCheckHaemodialysisCatheterPosition

16 Appendix 4: SQL Model Search Script.

16.1 Introduction

The following is a SQL script that can be used in the EA Custom Model Search functionality as part of a user defined search. It is primarily aimed at returning associated requirements from a given selected requirement. The query returns:

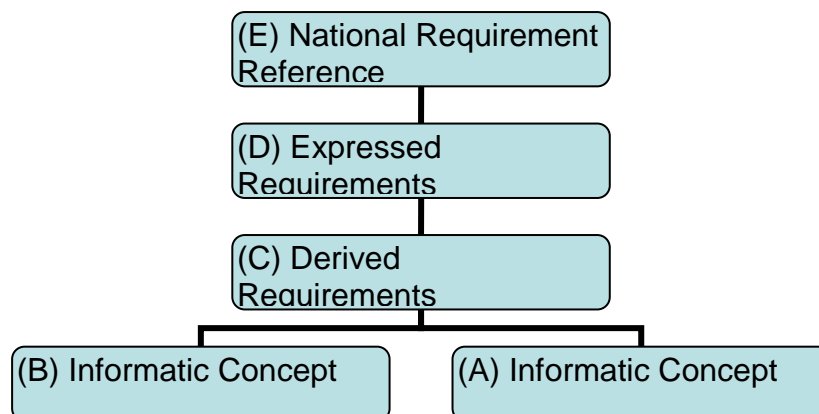
- Requirement name
- Requirement type
- Requirement stereotype

The script can be amended to return other Requirement parameters if required

NOTE: You will have to set EA to use Jet 4.0. This points EA to a more recent version of Access engine. No wider compatibility issues have been identified with this.

16.2 Query Logic

Assuming the following structure of Requirements:



The query script will return:

[Search on Informatic Concept A](#)

Output: Informatic Concept A, Derived Requirements C, Expressed Requirements D, National Requirement Reference E

[Search on Informatic Concept B](#)

Output: Informatic Concept B, Derived Requirements C, Expressed Requirements D, National Requirement Reference E

[Search on Derived Requirements C](#)

Output: Informatic Concept A, Informatic Concept B, Derived Requirements C, Expressed Requirements D, National Requirement Reference E

[Search on Expressed Requirements D](#)

Output : Informatic Concept A, Informatic Concept B, Derived Requirements C, Expressed Requirements D, National Requirement Reference E

[Search on National Requirement Reference E](#)

Output: Informatic Concept A, Informatic Concept B, Derived Requirements C, Expressed Requirements D, National Requirement Reference E

16.3 SQL Script

```

select t.ea_guid AS CLASSGUID, t.Object_Type AS CLASSTYPE, t.Name,
t.object_type as Type, IIF(t.Stereotype = 'Functional', "Expressed Requirement",
IIF(t.Stereotype = 'Report', "Expressed Requirement", t.Stereotype)) as Stereotype
, t.object_id
from
(
SELECT t_connector_4.Start_Object_ID AS C_OBJECT_ID
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID
WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search
Term>'))

UNION

SELECT t_connector_6.Start_Object_ID AS C_OBJECT_ID
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID
WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search
Term>'))

UNION

```

```

SELECT t_connector_5.Start_Object_ID AS C_OBJECT_ID
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID

```

```

WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search
Term>'))

```

UNION

```

SELECT t_connector.Start_Object_ID AS C_OBJECT_ID
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID

```

```

WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search
Term>'))

```

UNION

```

SELECT t_connector_1.Start_Object_ID AS C_OBJECT_ID
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON

```

```
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID
```

```
WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search
Term>'))
```

UNION

```
SELECT t_connector_2.Start_Object_ID AS C_OBJECT_ID
```

```
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID
```

```
WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search
Term>'))
```

UNION

```
SELECT t_connector_2.End_Object_ID AS C_OBJECT_ID
```

```
FROM ((t_object INNER JOIN t_connector ON t_object.Object_ID =
t_connector.End_Object_ID) LEFT JOIN ((t_connector AS t_connector_1 LEFT JOIN
t_connector AS t_connector_2 ON t_connector_1.End_Object_ID =
t_connector_2.Start_Object_ID) LEFT JOIN t_connector AS t_connector_3 ON
t_connector_2.End_Object_ID = t_connector_3.Start_Object_ID) ON
t_object.Object_ID = t_connector_1.Start_Object_ID) INNER JOIN (t_object AS
t_object_1 LEFT JOIN ((t_connector AS t_connector_5 LEFT JOIN t_connector AS
t_connector_6 ON t_connector_5.Start_Object_ID = t_connector_6.End_Object_ID)
LEFT JOIN t_connector AS t_connector_4 ON t_connector_6.Start_Object_ID =
t_connector_4.End_Object_ID) ON t_object_1.Object_ID =
t_connector_5.End_Object_ID) ON t_connector.Start_Object_ID =
t_object_1.Object_ID
```

```
WHERE (((t_object_1.Name)='<Search Term>')) OR (((t_object.Name)='<Search  
Term>'))  
) conn_object,  
t_object t  
where t.object_id = conn_object.c_object_id  
  
;
```

17 Appendix 5: Product Descriptions

17.1 Stakeholder Information View

Product Name	Stakeholder Information View
Status	Required
Purpose	<p>To illustrate the major information concepts within the domain and the associations between them.</p> <p>To provide an appropriate frame of reference for subsequent analysis.</p> <p>Besides being the starting information product for work in the domain, it is a general purpose tool for any discussion of the information content of the domain.</p>
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> • Project Team: to inform and advise on the project scope. • Domain experts: to agree overall information concepts of business area. <p>External:</p> <ul style="list-style-type: none"> • All: to provide an overview and introduction to the information concepts within the domain.
Description	<p>The Stakeholder Information View is a basic overview of the information concepts and their relationships with the business area or domain, this can be represented diagrammatically or can also directly re-use existing stakeholder artefacts if they exist and specifically identifies:</p> <ul style="list-style-type: none"> • Identification of the business area(s) and system(s) with which the domain is concerned • Identification of classes (information concepts) that are shared with other domains • Identification of classes (information concepts) that are the source and destination of communications between business area(s) and between system(s) <p>It is not intended to be a definitive document, so will not capture the entire information requirement and their associated flows; however it is intended that the content should be understandable and recognisable to business users.</p> <p>The Stakeholder Information View is expected to be published, and at the point of publication it should be correct and consistent with the rest of the artefact set.</p>

Product Name	Stakeholder Information View	
Format and Content	<p>The Stakeholder Information View can be constructed as a UML Class diagram.</p> <p>The Model shows:</p> <ul style="list-style-type: none"> • Named classes – without attributes or operations, but with a brief description and an alias. • Named and directed associations – without cardinalities <p>The model may be annotated using notes but these will be of a general business nature rather than representing formal elements such as constraints.</p> <p>The Stakeholder Information View Class diagram will be produced using a suitable modelling tool, which it is assumed will support any annotation requirements.</p> <p>Alternatively, the SIV may also be a direct re-use of stakeholder artefacts if these exist in a format that helpful to assist stakeholder understanding. It may be useful to transform these into class diagram form at this stage or leave it until the Information Analysis Model to do this.</p>	
Construction Guidelines	See Authoring Guide	
Template(s)	<ul style="list-style-type: none"> • No 	
Inputs and Predecessor Products	<ul style="list-style-type: none"> • Any previous Stakeholder Information View for this or related domains. 	
Peer Products	<ul style="list-style-type: none"> • Requirements Catalogue • Stakeholder Business Model. 	
Successor Products	<ul style="list-style-type: none"> • Information Analysis Models. • Use Case Model 	
Quality Criteria	Quality Requirement	Quality Check
	Describes the domain	Project Manager and Domain expert sign off.
	Conforms to the Authoring Guide	Technical review – using review checklist
	Terminology of content must be recognisable to domain participants.	Domain expert review and sign-off

Product Name	Stakeholder Information View	
	All Actors represented on the SIV, which are contained in the Actors Catalogue, must be referenced using the Relationship Matrix (see §13.4.27).	Technical review – using review checklist
	No attributes or methods within SIV classes	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> Likely to be re-used as input to further work on this domain. 	
Examples	Domain Example: No	Generic Example: No

17.2 Stakeholder Business Model

Product name	Stakeholder Business Model
Status	Required
Purpose	<p>To provide traceability between the analysis artefacts and the domain(s)</p> <p>To gain an initial understanding of the domain</p> <p>To clarify scope of the analysis required.</p> <p>To aid the identification of candidate use cases.</p> <p>To ensure stakeholder engagement.</p> <p>To analyse and agree outline business processes for the domain.</p> <p>To inform planning and development of downstream dynamic analysis artefacts.</p>
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> Project Team: to meet above purpose. <p>External:</p> <ul style="list-style-type: none"> Users: verify the business processes and requirements. Domain Experts: verify the business processes and requirements.
Description	An unconstrained model showing the high-level domain information such as processes/flows, participants and concepts
Format and Content	Unconstrained. Where this is created by the Analyst, it is expected that a UML Activity Diagram will be typical.
Construction Guidelines	See Authoring Guide
Template(s)	No

Product name	Stakeholder Business Model	
Inputs and Predecessor Products	Relevant pre-existing work.	
Peer Products	<ul style="list-style-type: none"> Requirements Catalogue (Domain Analysis and Scoping Phase) Stakeholder Information View 	
Successor Products	<ul style="list-style-type: none"> Business Process Models Use Case Models 	
Quality Criteria	Quality Requirement	Quality check
	SBM must comply with the scope stated within the Work package and accurately reflect domain process.	Domain expert review and sign-off by all identified reviewers
	Conforms to the Authoring Guide	Technical review – using review checklist
	All ‘normal’ activities (i.e. exclude initial & final activities) must have one incoming flow and one outgoing flow.	Technical review – using review checklist
	Terminology of content must be recognisable to domain participants.	Domain expert review and sign-off
	Domain Specific information concepts to be represented within the SIV	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> Unlikely to be reusable outside the domain. Likely to be refined/elaborated as work progresses. 	
Examples	Domain Example: No	Generic Example: see Figure 43, Figure 44 & Figure 45

17.3 Requirements Catalogue

Product name	Requirements Catalogue
Status	Mandatory
Purpose	<p>To provide a complete set of requirements which are:</p> <ul style="list-style-type: none"> • stated using unambiguous language • recognisable to business stakeholders • usable within later project-cycle activities such as design, development, testing and implementation • the basis for demonstrating the fulfilment of requirements • Used to provide readers who are not familiar with modelling techniques with a point of entry to analysis/design artefacts. • Used to inform Analysis activities. • Used to inform Implementation activities and to provide a basis for traceability of requirements from source documentation through to Implementation
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> • Project Team: to meet above purpose. • Test Team: to inform test cases. • Requirements Management Team: to trace schedule requirement fulfilment. <p>External:</p> <ul style="list-style-type: none"> • Domain Experts: verify the suitability of the requirements (and by extension the project). • Suppliers: use as input into development work. • Users: use as input to User Acceptance Testing.
Description	Definitive set of LRA project or domain requirements. An individual requirement is defined to be a single statement of need for what a product or service should be or do which identifies a necessary capability, characteristic, or quality of a system in order for it to have value and utility.
Format and Content	<p>The Requirements Catalogue has 2 forms:</p> <ul style="list-style-type: none"> • The primary form is a graphic and textual set of requirements elements held in UML packages within a package hierarchy as part of the wider project hierarchy. Each package has diagrammatic representations which compose the graphic requirements elements, expose their textual content and show 'realising' UML elements. • The secondary form is an RTF document generated from the primary form and reincorporated into the model. NB.

Product name	Requirements Catalogue	
	<p>The secondary form is not edited directly; it is always generated from the primary form.</p> <ul style="list-style-type: none"> • The Requirements Catalogue has 3 layers of content: • The first layer, the “Expressed Requirements”, contains a set of requirements as extracted from source requirement documents. • The second layer, the “Derived Requirements”, is a transform of the first layer. A single Expressed Requirement may contain several atomic requirements which are recorded in this second layer as “Derived Requirements”. • The third layer, the “Informatic Concepts”, contains a listing of Informatic concepts to be found in each of the Derived Requirements. An Informatic Concept is word or a phrase expressed in the stakeholders natural language that describes an area or subject that may be important from an informatics perspective. More formal analysis and elaboration in successor artefacts will determine the extent of its importance. A single derived requirement may contain one or more Informatics Concepts. Similarly a single common Informatic Concept may be distilled from many Derived Requirements. • 	
Construction Guidelines	<ul style="list-style-type: none"> • See LRA Authoring Guide 	
Template(s)	<ul style="list-style-type: none"> • No. 	
Inputs and Predecessor Products	<ul style="list-style-type: none"> • Project Documentation • Any previous requirements documentation for this or related domains. 	
Peer Products	<ul style="list-style-type: none"> • Stakeholder Business Model • Stakeholder Information View 	
Successor Products	<ul style="list-style-type: none"> • BPM • Outline Query • Use Case Model • IAM 	
Quality Criteria	Quality Requirement	Quality Check
	The Requirements Catalogue must comply with the scope stated within the Work package	Domain expert review and sign-off

Product name	Requirements Catalogue	
	Conforms to the Authoring Guide	Technical review – using review checklist
	Domain Specific information concepts to be represented within the Stakeholder Information View	Technical review – using review checklist
	Appropriately structured – in a root 'Requirements Catalogue', with a repeating package structure and valid diagrams.	Technical review – using review checklist
	Any deprecated requirements must be held within a package named 'Deprecated Requirements' which should be placed in the 'Requirements' package of the Requirements Catalogue' package; with any deprecated requirements also removed from current diagrams	Technical review – using review checklist
	Requirements named & aliased with automatically generated identifiers	Technical review – using review checklist
	Each package contains a description in its 'notes'	Technical review – using review checklist
	Requirements Palette is grouped into 'National Requirements References', 'Expressed' and 'Derived' and 'Query' requirements	Technical review – using review checklist
	The only association types allowed between requirements are aggregations.	Technical review – using review checklist
	The only association types allowed between requirements and other elements are realisations and traces	Technical review – using review checklist
	In scope, 'current' phase derived requirement must be 'derived from' at least 1 element within the model	Technical review – using review checklist
	Terminology of content must be recognisable to domain participants.	Domain expert review and sign-off

Product name	Requirements Catalogue	
	The content of individual requirements must be cohesive, complete, unique, consistent, correct, feasible, unambiguous and verifiable.	Technical review – using review checklist
	Requirements content (i.e. notes) should be displayed in requirements diagrams	Technical review – using review checklist
	Requirement Metadata 'Phase' must identify the phase associated with the requirement. In the absence of specific project/domain terminology this must be set to '1.0'	Technical review – using review checklist
	Deprecated requirements must have their names prefixed as 'Deprecated' and the reason for its deprecation and details of any other requirements that replace it or to which its parameters have been subsumed must be added to its description (Notes)	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> Unlikely to be reusable outside the domain. Will be used as a basis for identification of Use Case Models and IAMs. 	
Examples	Domain Example: No	Generic Example:

17.4 Business Process Model

Product name	Business Process Model
Status	Required
Purpose	<p>To analyse and agree outline business processes of interest for the domain. N.B. it should be noted that the scope of the business processes is restricted to the use case model set</p> <p>To inform logical design.</p> <p>To inform planning and development of downstream dynamic analysis artefacts.</p> <p>To show how the Use Cases support business processes.</p> <p>To show the process relationships between the Use Cases.</p>
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> • Project Team: to meet above purpose. • Technical Architects: to inform technical design. <p>External:</p> <ul style="list-style-type: none"> • All: Starting point for process view of the domain. • Domain Experts: verify the business processes and requirements.
Description	<p>The Business Process Model shows one, many or all possible configurations of the Use Cases within the domain.</p> <p>The activities within the Business Process Models are the Use Cases from within the Use Case Catalogue, connected by process logic. Where such logic between Use Cases exists, this should also be represented within the Use Case Model Set (as pre/post conditions and triggers). However the BPM may not represent the definitive statement of process within the (part of the) domain, unless non-Use Case activities are included within the BPM scope.</p> <p>Where there are multiple discrete business processes within an area of interest, and where the relationships between those business processes vary depending on scenario, this is referred to as a discontinuous flow (see §12.3.2.2.3). In such cases, example configurations will be constructed:</p> <ul style="list-style-type: none"> • Each configuration will be depicted using a UML activity diagram. • The set of example configurations can be accessed from a single diagram. This diagram does not show any relationships between the configurations. • Users of the diagram may access any of the example configurations.

Product name	Business Process Model	
	Example configurations are not required where the area of interest is described by a single business process.	
Format and Content	UML Activity Diagrams	
Construction Guidelines	See Authoring Guide	
Template(s)	No	
Inputs and Predecessor Products	Stakeholder Information View Stakeholder Business Model	
Peer Products	<ul style="list-style-type: none"> • Use Case Model • Information Analysis Model 	
Successor Products	<ul style="list-style-type: none"> • Use Cases. 	
Quality Criteria	Quality Requirement	Quality check
	BPM must comply with the scope stated within the Work package and accurately reflect domain process.	Domain expert review and sign-off by all identified reviewers
	Conforms to the Authoring Guide	Technical review – using review checklist
	Activities must be represented as Use Cases on the Use Case Catalogue or denoted as being for information only.	Technical review – using review checklist
	All final activities on a Use Case Activity Diagram must be represented as outgoing flows from the parent activity on the BPM -this requires a decision to test the outcome of the Use Case.	Technical review – using review checklist
	All 'normal' activities (i.e. exclude initial & final activities) must have one incoming flow and one outgoing flow.	Technical review – using review checklist
	Terminology of content must be recognisable to domain participants.	Domain expert review and sign-off
	Where an activity represents a	Technical review – using

Product name	Business Process Model	
	Use Case this must be cross referenced using the relationship matrix functionality	review checklist
	Where Stakeholder Business Model content is not represented within the BPM, this should be identified as out of scope in the Work package	Technical review – using review checklist
	Pre and post-conditions identified in the Use Case must not conflict with process as stated in the BPM.	Technical review – using review checklist
	Domain Specific information concepts to be represented within the SIV	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> Unlikely to be reusable outside the domain. Likely to be refined/elaborated as work progresses. 	
Examples	Domain Example: No	Generic Example: see Figure 43, Figure 44 & Figure 45

17.5 Use Case

Product name	Use Case
Status	Required
Purpose	<ul style="list-style-type: none"> To describe in detail an area of business functionality. To identify interactions between business areas. To inform the logical design and elaboration of subsequent artefacts in the development lifecycle. To provide analysis fulfilment of Derived requirements.
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> Project Team: to meet above purpose. Technical Architects: May take the Use Cases as the input for their realisation work. <p>External:</p> <ul style="list-style-type: none"> Domain Experts: verify Use Cases. Users: verify Use Cases for accuracy, completeness and consistency from a user's perspective and to give their approval of those Use Cases.

Product name	Use Case
	<ul style="list-style-type: none"> Suppliers: use as input into development work.
Description	<p>A Use Case describes the interactions between a user and a system. The system may be a Business System or an Automated System.</p> <p>A Use Case is a statement of the functional and process needs of the business user.</p>
Format and Content	<p>A Use Case should include the following:</p> <ul style="list-style-type: none"> Use Case Name & Identifier. Description: This should introduce the Use Case to the reader, placing it in context with the rest of the system and/ or domain and describing it and its purpose at a very high level. Any information necessary to understand subsequent sections of Use Case should be detailed in this section. Actors: All actors referenced within a Use Case should be identified. Trigger: A description of the event(s) that initiates the Use Case should be provided. Pre-conditions: All conditions that must be satisfied if the Use Case is to be initiated. Post Conditions: Post conditions may be considered in two forms; Success and Guaranteed. The Success conditions show the state of the system after the process has run as expected. The Guaranteed condition shows the state we can expect the system to be in after an unsuccessful condition. Business Rules to be observed within the Use Case. See §8.2.3. Basic path: This describes the process steps that would be expected to happen in the majority of cases if the process was run. Alternate path(s): These are alternative process steps that can occur within the Use Case, showing the possible deviations from the basic path.
Construction Guidelines	<ul style="list-style-type: none"> See Authoring Guide
Template(s)	<ul style="list-style-type: none"> No
Inputs and Predecessor Products	<ul style="list-style-type: none"> Use Case Model. BPM
Peer Products	<ul style="list-style-type: none"> Information Analysis Model. Use Case Activity Diagram.
Successor	<ul style="list-style-type: none"> None

Product name	Use Case	
Products		
Quality Criteria	Quality Requirement	Quality Check
	Conforms to the Authoring Guide	Technical review – using review checklist
	Use Cases are physically located within the Use Case Catalogue	Technical review – using review checklist
	All actors (including specialisations) identified in the Use Case must be represented within the Use Case Model and stored in the Actors Catalogue	Technical review – using review checklist
	Pre and post-conditions identified in the Use Case must not conflict with process as stated in the BPM.	Technical review – using review checklist
	Domain specific information concepts must be represented within the corresponding IAM.	Technical review – using review checklist
	Each non-deprecated Use Case must have an equivalent activity on the BPM, which must be referenced using the Relationship Matrix.	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> Unlikely to be reusable outside the domain. Likely to be refined/elaborated as work progresses 	
Examples	<ul style="list-style-type: none"> Domain Example: No 	<ul style="list-style-type: none"> Generic Example: No

17.6 Use Case Model

Product name	Use Case Model
Status	Required (Mandatory for communication analysis domains)
Purpose	<ul style="list-style-type: none"> A single Use Case Model provides an organised view of the behaviours for a subset of the domain. The collection of Use Case Models provides an organised view of the behaviours for the whole domain. To inform the logical design and elaboration of subsequent artefacts in the development lifecycle. To provide readers with a point of entry to analysis/design artefacts.
Audience and Use	Internal: <ul style="list-style-type: none"> Project Team: to meet above purpose.

Product name	Use Case Model	
	<ul style="list-style-type: none"> Technical Architects: May take the Use Case Model as the input for their realisation work. <p>External:</p> <ul style="list-style-type: none"> Domain Experts: verify Use Cases identified in Use Case Model are within scope. Users: verify Use Cases identified in Use Case Model for accuracy, completeness and consistency from a user's perspective and to give their approval of those use cases. Suppliers: use as input into development work. 	
Description	<p>The Use Case Model consists of a use case diagram with Use Cases and Actors.</p> <p>A Use Case Model will derive from the Requirements Catalogue and provides the start point for functional analysis. There may be multiple Use Case Models describing different parts of the subject domain</p>	
Format and Content	<p>The Use Case Model will be produced using a suitable UML modelling tool and will be composed of a number of parts:</p> <ul style="list-style-type: none"> A Use Case diagram comprising: <ul style="list-style-type: none"> Actors. Use Cases (identified using a suitable identifier and name). Associations showing the interactions between Use Cases and Actors or Actors and Actors or Use Cases and other Use Cases. A System Boundary. 	
Construction Guidelines	<ul style="list-style-type: none"> See Authoring Guide 	
Template(s)	<ul style="list-style-type: none"> No 	
Inputs and Predecessor Products	<ul style="list-style-type: none"> Requirements Catalogue Business Process Model Stakeholder Information View 	
Peer Products	<ul style="list-style-type: none"> Information Analysis Model. Business Process Model. 	
Successor Products	<ul style="list-style-type: none"> Use Cases. 	
Quality Criteria	Quality Requirement	Quality Check

Product name	Use Case Model	
	Conforms to the Authoring Guide	Technical review – using review checklist
	All actors (including specialisations) identified in the Use Cases must be represented within the Use Case Model	Technical review – using review checklist
	All represented Use Cases must be stored within the Use Case Catalogue	Technical review – using review checklist
	All represented Actors must be stored within the Actors Catalogue	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> Unlikely to be reusable Likely to be refined/elaborated as work progresses 	
Examples	Domain Example: No	Generic Example: No

17.7 Use Case Catalogue

Product name	Use Case Catalogue
Status	Mandatory where use cases are developed
Purpose	<ul style="list-style-type: none"> Organise and contain the Use Cases into packages.
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> Project Team: to meet above purpose. Domain Experts: to agree the list of Use Cases to be developed. <p>External:</p> <p>All: to understand the grouping of Use Cases</p>
Description	Use Case Catalogue has no graphical representation and is simply a directory in the model hierarchy. The Use Case Catalogue is part of the Use Case Model and can have its own substructure of packages which are used to provide an organising principle for the contained Use Cases which are reused across the rest of the model.
Format and Content	The Use Case Catalogue is a directory structure containing other directories (such as the Actor Catalogue) and Use Cases
Construction Guidelines	<ul style="list-style-type: none"> See Authoring Guide

Product name	Use Case Catalogue	
Template(s)	<ul style="list-style-type: none"> No 	
Inputs and Predecessor Products	<ul style="list-style-type: none"> Requirements Catalogue 	
Peer Products	<ul style="list-style-type: none"> Use Cases Business Process Model. 	
Successor Products	<ul style="list-style-type: none"> Use Case Models. Business Process Model. 	
Quality Criteria	Quality Requirement	Quality Check
	Conforms to the Authoring Guide	Technical review – using review checklist
	All Use Cases within the artefact set must be stored within the Use Case Catalogue.	Technical review – using review checklist
	All Actors within the artefact set must be stored within the Actors Catalogue.	Technical review – using review checklist
	All Use Cases within the artefact set must appear within the Use Case Model	Technical review – using review checklist
	All Actors within the artefact set must appear within the Use Case Model	Technical review – using review checklist
	Identified Use Cases and Actors cover full scope	Domain expert review
Configuration Management	Unlikely to be reusable outside the domain.	
Examples	Domain Example: No	Generic Example: No

17.8 Information Analysis Model

Product name	Information Analysis Model
Status	Required
Purpose	<p>To provide an information model for the area of interest. It provides the domain information context for subsequent analysis and/or development.</p> <p>To allow the exploration of the domain information requirements</p>

Product name	Information Analysis Model
	against known constraining data architectures
Audience and Use	<p>Internal:</p> <ul style="list-style-type: none"> • Project Team: to determine the business data needs within the area of interest • Technical Architects: to inform technical design. <p>External:</p> <ul style="list-style-type: none"> • Domain Experts: verify the business data. • Suppliers: use as input into development work.
Description	A logical data model showing the classes, associations, cardinalities and attributes for a particular area of interest.
Format and Content	<p>The Information Analysis Model is a UML Class diagram.</p> <p>The Model shows:</p> <ul style="list-style-type: none"> • Named classes <ul style="list-style-type: none"> ○ Class description ○ Attributes • Datatypes where determined • Cardinality • Constraints • Usually a brief description ○ Operations • Constraints • Usually a brief description • Associations <ul style="list-style-type: none"> • Must have Cardinality • Direction should normally be shown • May have Name (where appropriate) • Must not have Constraints.
Construction Guidelines	<ul style="list-style-type: none"> • See Authoring Guide
Template(s)	<ul style="list-style-type: none"> • No
Inputs and Predecessor Products	<ul style="list-style-type: none"> • Stakeholder Information View • Requirements Catalogue (Informatic Concepts) • Domain Query Functionality

Product name	Information Analysis Model	
Peer Products	<ul style="list-style-type: none"> • Use Case Model(s). • Use Case Activity Diagrams. • Informatic Concepts 	
Successor Products	<ul style="list-style-type: none"> • Domain Information Descriptions • Candidate Data Elements • 	
Quality Criteria	Quality Requirement	Quality Check
	Conforms to the Authoring Guide	Technical review – using review checklist
	Encompasses full area of interest	Domain Expert Review
	Data concepts from the Use Case Model Set must be represented as classes or attributes.	Technical Review
	Data concepts from the Requirements Catalogue must be represented as classes or attributes.	Technical Review
	Call Operations from the Domain Query Functionality Catalogue must be represented as classes operations	Technical Review
	All attributes must have specified data-types.	Technical review – using review checklist
Configuration Management	<ul style="list-style-type: none"> • Likely to be re-used as input to further work on this domain 	
Examples	Domain Example: No	Generic Example: No

17.9 Outline Query Activity Diagram

Product name	Outline Query Activity Diagram
Status	Required
Purpose	<ul style="list-style-type: none"> • To describe in detail an area of business query functionality. • To identify query input and output information. • To inform the logical design and elaboration of subsequent artefacts in the development lifecycle. <p>To provide analysis fulfilment of schedule requirements. To show how the Outline Queries support Derived Requirements.</p>
Audience and Use	Internal:

Product name	Outline Query Activity Diagram	
	<ul style="list-style-type: none"> Project Team: to meet above purpose. Technical Architects: to inform technical design. <p>External:</p> <ul style="list-style-type: none"> All: Starting point for query view of the domain. Domain Experts: verify the query processes and requirements. 	
Description	The Outline Query Activity Diagram describes the input information required for specified logical query to function and the corresponding output information the query will produce between a user and a system.	
Format and Content	UML Activity Diagrams	
Construction Guidelines	See Authoring Guide	
Template(s)	No	
Inputs and Predecessor Products	Requirements Catalogue – Query Requirements	
Peer Products	<ul style="list-style-type: none"> Use Case Model BPM Information Analysis Model 	
Successor Products	<ul style="list-style-type: none"> Domain Query Functionality 	
Quality Criteria	Quality Requirement	Quality check
	Conforms to the Authoring Guide	Technical review – using review checklist
	Outline Queries are physically located within the Outline Queries Package	Technical review – using review checklist
	All Outline Queries activities at least one specified input parameter node and one specified output parameter node.	Technical review – using review checklist
	Terminology of content must be recognisable to domain participants.	Domain expert review and sign-off
Configuration Management	<ul style="list-style-type: none"> Unlikely to be reusable outside the domain. Likely to be refined/elaborated as work progresses. 	

Product name	Outline Query Activity Diagram	
Examples	Domain Example: No	Generic Example: see Figure 43, Figure 44 & Figure 45

17.10 Domain Information Description

Product name	Domain Information Description
Status	Required
Purpose	<ul style="list-style-type: none"> • The DID Class is the overall container and owner for all the, attributes, constraints and datatypes from which the Candidate Data Elements are populated and structured • Specifies the business information requirement content of an area of interest within a single domain. • States the context of the information requirements within a single domain.
Audience and Use	<ul style="list-style-type: none"> • Internal: <ul style="list-style-type: none"> ○ <i>Project Team</i>: to meet above purpose ○ <i>Domain Experts</i>: to check correct understanding of information requirements and logical domain; as well as checking the correct understanding of the relevant communication artefacts. • External: <ul style="list-style-type: none"> ○ <i>Suppliers</i>: use as input into development work; as well as supporting the understanding of the CDs
Description	The Domain Information Description holds all the information requirements for a specific area of interest within the domain in a form that can be presented to terminologists and technical modellers. The development process is iterative and inter-dependent with all DID Description classes being derived from the IAM.
Format and Content	<ul style="list-style-type: none"> • A UML Class Diagram showing a subset of elements derived from the IAM <ul style="list-style-type: none"> ○ The DID is the information requirement model for each individual area of interest to be presented to terminologists and technical modellers. • Content: <ul style="list-style-type: none"> ○ Named classes / class packages from the CIM package with:

	<ul style="list-style-type: none"> • Attributes • Datatypes <ul style="list-style-type: none"> ○ Class contents may be shown or hidden on diagrams ○ Any annotations against the model must not modify or infer data content or structure. ○ Any annotations against the model must not modify or infer data content or structure. 	
Construction Guidelines	<ul style="list-style-type: none"> • See Authoring Guide 	
Template(s)	<ul style="list-style-type: none"> • No 	
Inputs and Predecessor Products	<ul style="list-style-type: none"> • Information Analysis Model. • BPM • UCM 	
Peer Products	<ul style="list-style-type: none"> • none 	
Successor Products	<ul style="list-style-type: none"> • Candidate Data Element Definitions 	
Quality Criteria	Quality Requirement	Quality Check
	Conforms to the Authoring Guide	Technical review
	All relevant information concepts from the IAM are represented	Technical review – using review checklist
	Significant overlap between class contents or meaning identified and resolved	Technical review
	All attributes must have specified datatypes (either a Candidate Data Element, Complete Data Elements or a complex class that holds other similarly typed attributes), which are reused from the datatype hierarchy or be explicitly defined and represented as a model-specific complex datatype. Attributes must also have cardinalities defined	Technical review
	All classes should have a description; all classes should have an alias	Technical review

	All DID classes and packages must be contained in DID package and where more than one DID is produced for a model a Diagram is also required	Technical review
	Every class in the DID (and therefore also in the DID Diagram) must be linked to a one or more IAM classes using the Relationship Matrix except model specific complex datatype classes.	Technical review
Configuration Management		
Examples	Domain Example: No	Generic Example: No

17.11 Candidate Data Element Description

Product name	Candidate Data Element Description
Status	Required
Purpose	These represent the fundamental information requirement that has been identified by the upstream artefacts. They are derived either from the gap established between the Information Analysis Model and the Domain Information Description or alternatively the Domain Query Functionality will highlight the absence of suitable of Complete Data Elements class attributes or operations required to process its logic
Audience and Use	<ul style="list-style-type: none"> • Internal: <ul style="list-style-type: none"> ○ <i>Project Team</i>: to meet above purpose ○ <i>Domain Experts</i>: to check correct understanding of information requirements and logical domain; as well as checking the correct understanding of the relevant communication artefacts. • External: <ul style="list-style-type: none"> ○ <i>Suppliers</i>: use as input into development work; as well as supporting the understanding of the CDs
Description	The Candidate Data Element Description is a single class with a single attribute of a predefined type. It represents a proposed specific information requirement that is presented to Terminologists and Technical modellers in the context of a DID for consideration. It acts as a complex datatype for a corresponding DID attribute. Once the Candidate Data

	Element Description has been resolved by the Terminologists and Technical modellers it is deprecated and replaced with the resulting Completed Date Element	
Format and Content	<ul style="list-style-type: none"> • A UML Class Diagram • Content: <ul style="list-style-type: none"> ○ Named classes / class packages from the CIM package with: <ul style="list-style-type: none"> • Attribute • Datatype ○ Class contents may be shown or hidden on diagrams ○ Any annotations against the model must not modify or infer data content or structure. ○ Any annotations against the model must not modify or infer data content or structure. 	
Construction Guidelines	<ul style="list-style-type: none"> • See Authoring Guide 	
Template(s)	<ul style="list-style-type: none"> • No 	
Inputs and Predecessor Products	<ul style="list-style-type: none"> • Domain Information Description 	
Peer Products	<ul style="list-style-type: none"> • none 	
Successor Products	<ul style="list-style-type: none"> • Candidate Data Element Definitions 	
Quality Criteria	Quality Requirement	Quality Check
	Conforms to the Authoring Guide	Technical review
	All relevant information concepts from the IAM are represented	Technical review – using review checklist
	Significant overlap between class contents or meaning identified and resolved	Technical review
	All attributes must have specified datatypes Attributes must also have cardinalities defined	Technical review
	All classes should have a description; all classes should have an alias	Technical review

	All CDED classes and packages must be contained in Data Element Definitions Palette package.	Technical review
Configuration Management		
Examples	Domain Example: No	Generic Example: No

17.12 Domain Query Functionality Activity Diagram

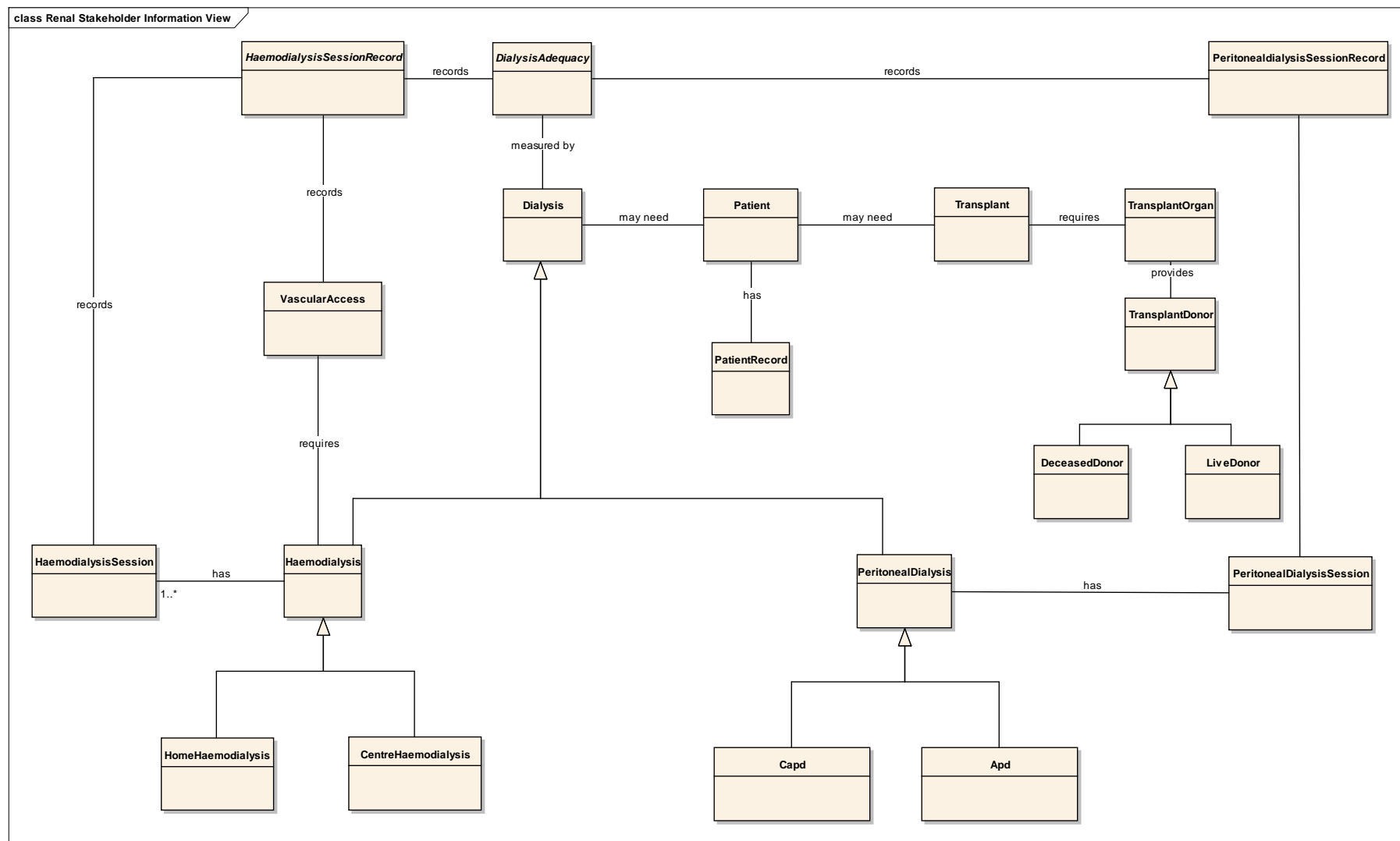
Product name	Domain Query Functionality Activity Diagram
Status	Required
Purpose	<p>To analyse and document the logical steps within the Outline Query.</p> <p>To provide the detailed representation of individual activities within the Outline Query.</p> <p>To inform logical design.</p> <p>To identify information requirements within the IAM.</p>
Audience and Use	<p>Internal:</p> <p>Project Team: to meet above purpose.</p> <p>Technical Architects: to inform technical design.</p> <p>External:</p> <p>Domain Experts: verify the query logic and requirements.</p> <p>Users: to inform implementation and process change.</p>
Description	<p>A Domain Query Functionality activity diagram is used to describe a process in terms of the sequence of its activities and subsidiary actions and flow of information between the activities. Decisions, actors and other information can also be represented in the diagrams.</p> <p>The Domain Query Functionality Activity Diagrams bounded by its Outline Query.</p>
Format and Content	A UML Activity Diagram
Construction Guidelines	See Authoring Guide
Template(s)	<ul style="list-style-type: none"> No
Inputs and Predecessor Products	<ul style="list-style-type: none"> Outline Query

Product name	Domain Query Functionality Activity Diagram	
Peer Products	<ul style="list-style-type: none"> None 	
Successor Products	IAM	
Quality Criteria	Quality Requirement	Quality Check
	Conforms to the Authoring Guide	Technical review – using review checklist
	Fit with business and purpose	Domain expert review and sign-off by an identified approver/approvers
	The name of the parent Outline Query activity within the Outline Query is the name of the Domain Query Functionality Activity Diagram.	Technical review – using review checklist
	Domain Query Functionality Activity Diagrams can only have sub-activities if they are themselves Outline Queries. Otherwise the Activity should be decomposed through using Actions	Technical review – using review checklist
	The name of a single activity within the Domain Query Functionality Activity Diagram must not replicate the name of the Outline Query.	Technical review – using review checklist
	The initial activity or Action in the Domain Query Functionality Activity Diagram is consistent with the input(s) identified in the Outline Query.	Technical review – using review checklist
	All final activities outputs should have matching post-conditions within the Outline Query.	Technical review – using review checklist
	All 'normal' activities (i.e. excludes initial, final activities, actions and Outline Queries) must have one incoming flow and one outgoing flow.	Technical review – using review checklist
	Any notes against activities do not amend the logic.	Technical review – using review checklist
	Actions must call a Class operation Technical review – using review checklist	
Configuration Management	Unlikely to be reusable Likely to be refined/elaborated as work progresses	

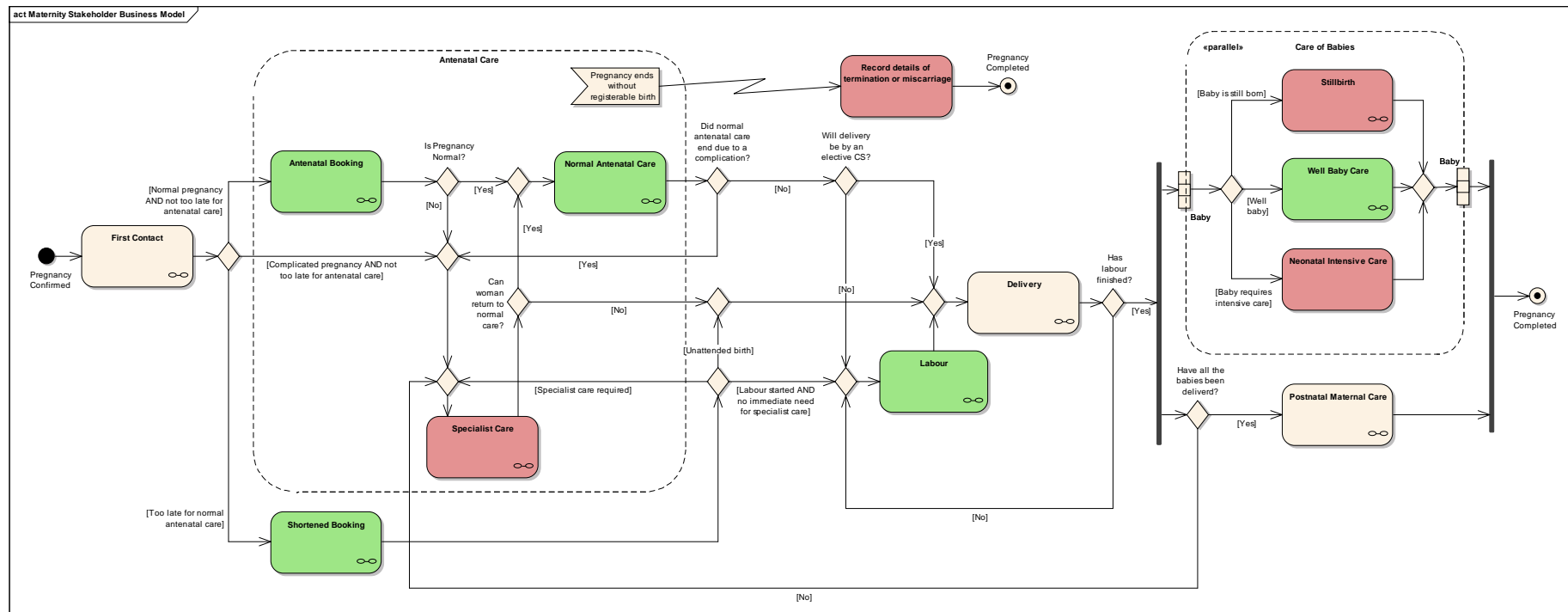
Product name	Domain Query Functionality Activity Diagram	
Examples	Domain Example: see No	Generic Example:No

18 Appendix 6: Artefact Examples

18.1 Stakeholder Information View









18.2 Stakeholder Business Model










18.3 Requirements Catalogue

custom National Requirements

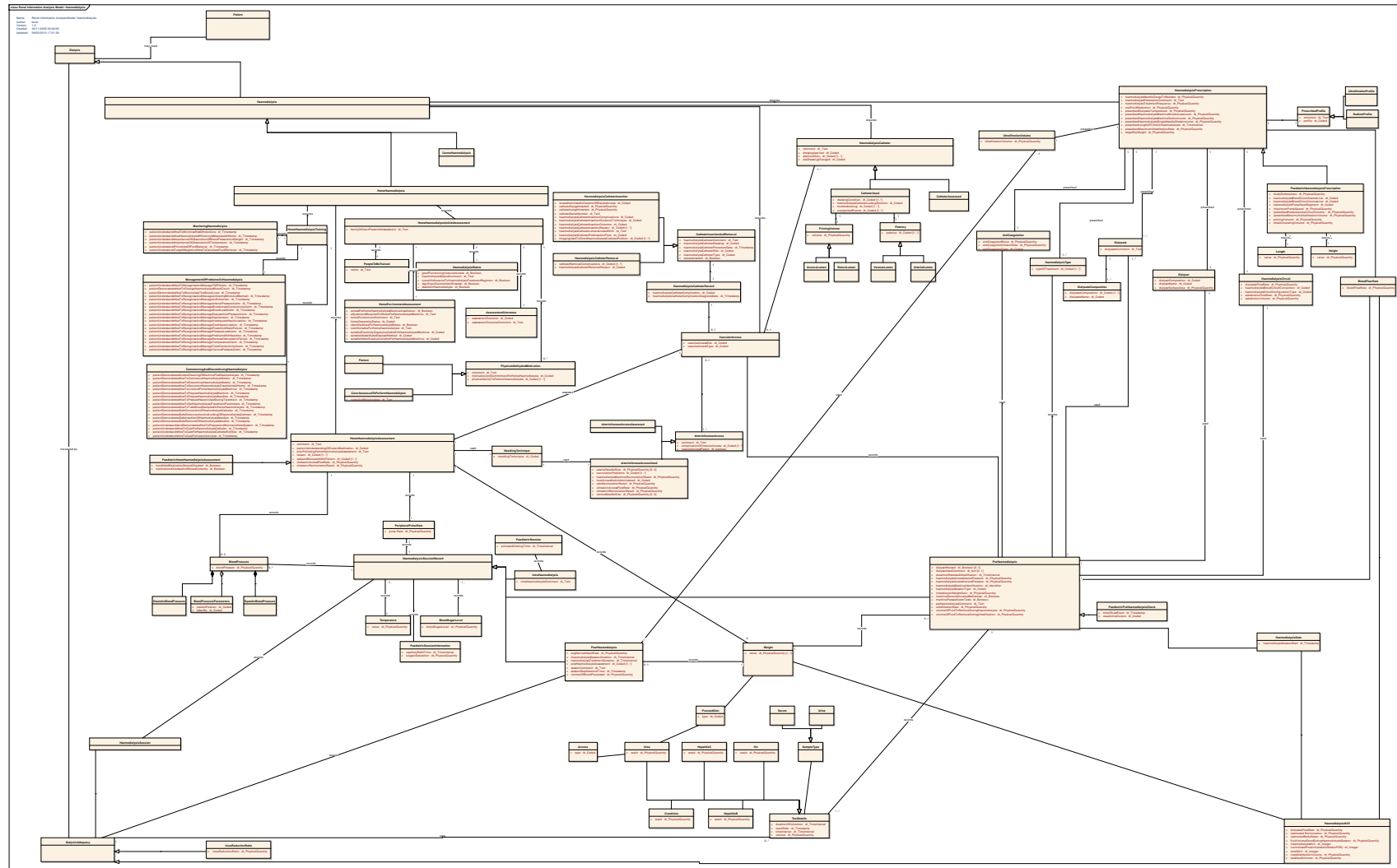
National Requirement References - Renal

-  + Chronic Kidney Disease in Adults: UK Guidelines for Identification, Management and Referral
-  + Clinical Content Specification: Specialist Renal Clinical Assessments and Records of Care Requirement for an Operational Information Standard
-  + Human Tissue Act - Removal and Storage of Tissue (July 2006)
-  + NICE Guidance
-  + OBS Renal Services 167
-  + The National Service Framework for Renal Services

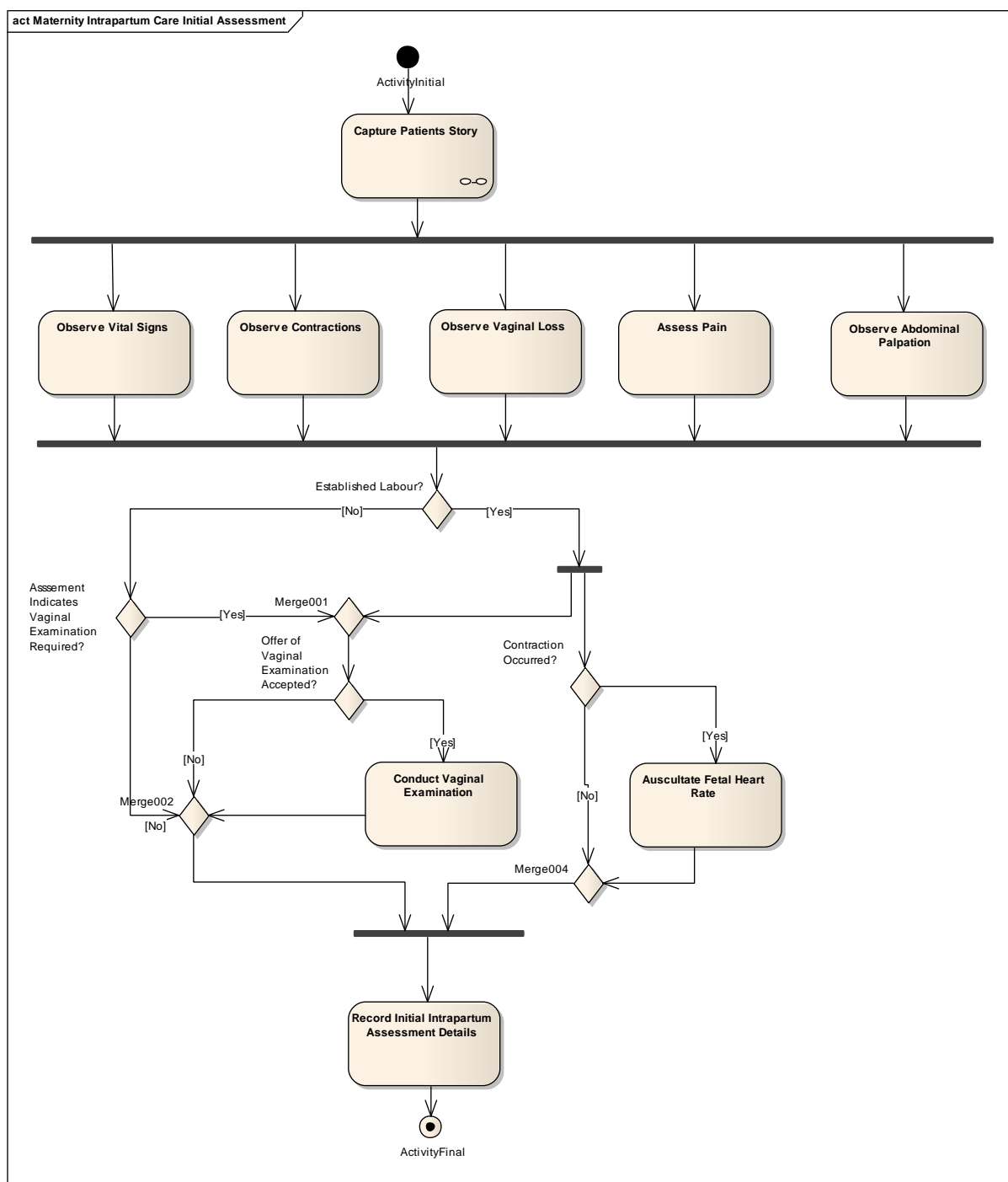
National Requirement References - Maternity

-  + TO BE CLARIFIED/CONFIRMED
-  + NICE Clinical Guideline 37 – Postnatal Care
-  + NICE Clinical Guideline 37 – Postnatal Care (Full guidance)
-  + NICE Clinical Guideline 55 – Intrapartum Care
-  + NICE Clinical Guideline 62 – Antenatal Care
-  + OBS (LSP Schedule 1.1) Maternity Services – Module 118
-  + NICE Guidance

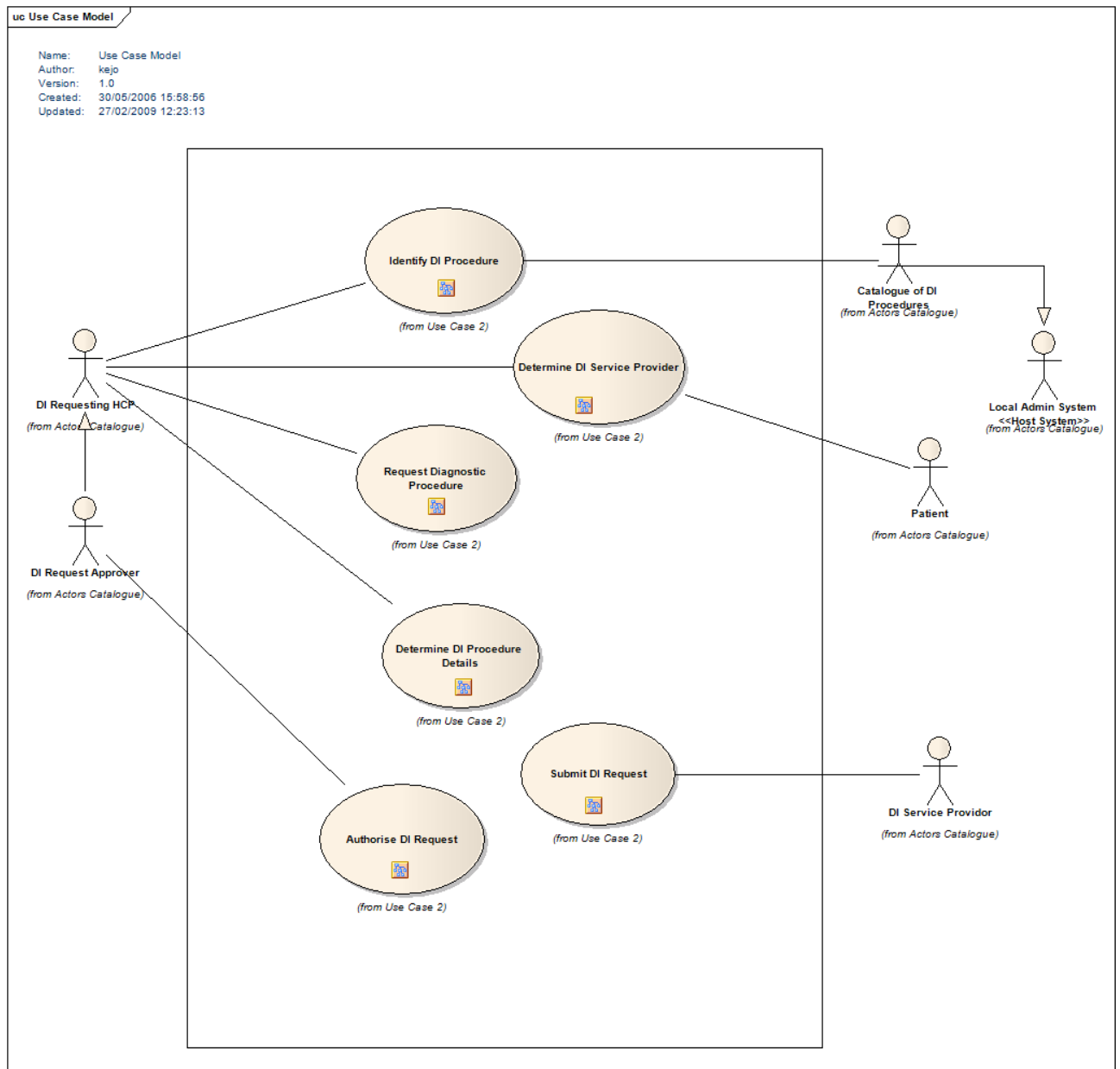
18.4 Information Analysis Model



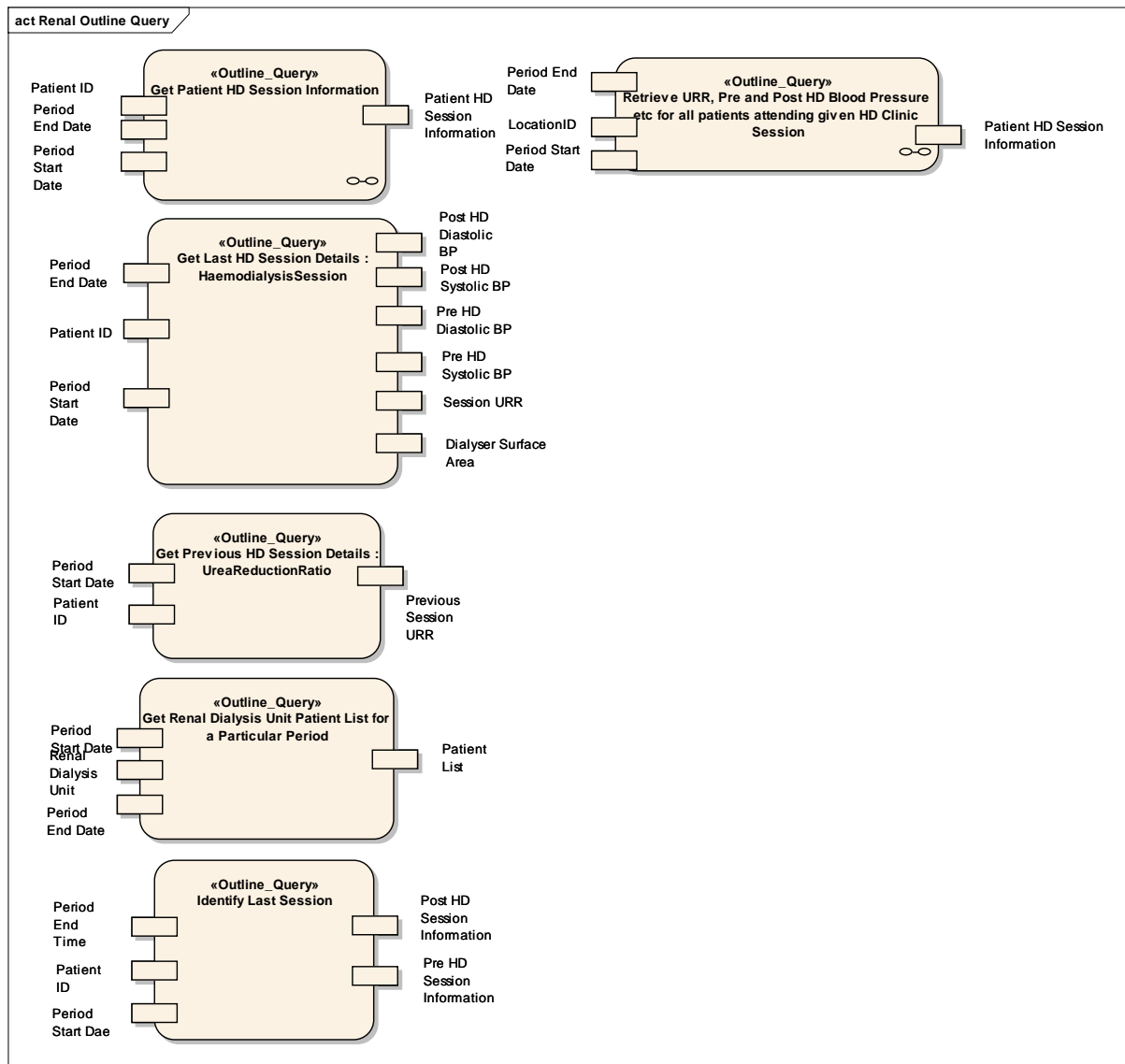
18.5 Business Process Model



18.6 Use Case Model



18.7 Outline Query Diagram



18.8 Domain Information Descriptions



18.9 Candidate Date Elements

class Haemodialysis Prescription Candidate Data Definitions

cded_OralFluidRestriction + Value: PQ	cded_LengthOfPatient + Value: PQ	Prescribed HD Anticoagulation cded_PrescribedHaemodialysisAnticoagulantType + Value: CS cded_PrescribedHaemodialysisAnticoagulantInfusionRate + Value: PQ [0..1] cded_PrescribedHaemodialysisAnticoagulantBolus + Value: PQ [0..1]
cded_PrescribedHaemodialysisTreatmentType + Value: CS [1..*]	cded_PrescribedHaemodialysisMachineSodiumLevel + Value: PQ	
cded_HaemodialysisTreatmentFrequency + Value: int	cded_PrescribedHaemodialysisMachineBicarbonateLevel + Value: PQ	
cded_TargetDryWeight + Value: PQ	cded_HaemodialysisMaximumUltrafiltrationRate + Value [0..1]	
cded_PrescribedDurationOnHaemodialysis + Value: PQ.Time	cded_PrescribedHaemodialysisUltrafiltrationVolume + Value: PQ	Used to describe Dialysate cded_PrescribedDialysateName + Value: CS [1..*] cded_PrescribedDialysateComposition + Value: PQ [1..*]
cded_PrescribedHaemodialysisBloodFlowRate + Value: PQ	cded_PrescribedHaemodialysisBloodCircuitConnection + Value: CS	
cded_PrescribedDialyserName + Value: CS	cded_PrescribedHDPumpHeadSegment + Value: CS	
cded_BodySurfaceArea + Value: PQ	cded_HaemodialysisVenousLine + Value: CS	
cded_PrescribedDialysateComment + Value: ED.Text	cded_PrescribedHaemodialysisMaximumUltrafiltrationVolume + Value: PQ	
cded_PrescribedDialysateFlowRate + Value: PQ	cded_PrescribedHaemodialysisMaximumPumpSpeed + Value: PQ	
cded_PrescribedDialysateTemperature + Value: PQ	cded_PrescribedHaemodialysisExtracorporealCircuitVolume + Value: PQ	
cded_PrescribedSubstitutionVolume + Value: PQ [0..1]	cded_PrescribedHaemodialysisCircuitConfigurationType + Value: CS	Recently added (16/11/09)
cded_PrescribedSubstitutionFlowRate + Value: PQ [0..1]	cded_PrescribedHaemodialysisSingleNeedleStrokeVolume + Value: PQ	
cded_HaemodialysisPrescriptionComment + Value: ED.Text		
cded_HaemodialysisUltrafiltrationProfileType + Value: CS		
cded_HaemodialysisSodiumProfileType + Value: CS		
cded_HaemodialysisUltrafiltrationProfileTypeComment + Value: ED.Text		
cded_HaemodialysisSodiumProfileTypeComment + Value: ED.Text		
cded_HaemodialysisArterialLine + Value: CS		
cded_PrescribedHaemodialysisNeedleGauge + Value: PQ		
cded_PrescribedHaemodialysisPrimingVolume + Value: PQ		
cded_PrescribedDialyserSurfaceArea + Value: PQ	Added 18/11	
cded_PrescribedDialyserComposition + Value: CS		

18.10 Domain Functionality Activity Diagram

