Functional Specification of 'ELMTest'

Put author(s) here
(This document was generated by Ampersand vs. 2.0.0.18)

Sat Feb 26 10:18:24 W. Europe Standard Time 2011

Contents

Chapter 1

Introduction

This document defines the functionality of an information system called 'ELMTest'. It defines business services in a system where people and applications work together in order to fullfill their commitments. A number of these rules have been used as functional requirement to assemble this functional specification[1]. Those rules are listed in chapter 2, ordered by theme.

The diagnosis in chapter 3 is meant to help the authors identify shortcomings in their Ampersand script.

The conceptual analysis in chapter 4 is meant for requirements engineers and architects to validate and formalize the requirements from chapter 2. It is also meant for testers to come up with correct test cases. The formalization in this chapter makes consistency of the functional specification provable. It also yields an unambiguous interpretation of all requirements.

Chapters that follow have the builders of 'ELMTest' as their intended audience. The data analysis in chapter 6 describes the data sets upon which 'ELMTest' is built. Each subsequent chapter defines one business service. This allows builders focus on a single service at a time. Together, these services fulfill all commitments from chapter 2. By disclosing all functionality exclusively through these services, 'ELMTest' ensures compliance to all rules from chapter 2.

[1] To use agreements as functional requirements characterizes the Ampersand approach, which has been used to produce this document.

Chapter 2

Shared Language

This chapter defines the natural language, in which functional requirements of 'ELMTest' can be discussed and expressed. The purpose of this chapter is to create shared understanding among stakeholders. The language of 'ELMTest' consists of concepts and basic sentences. All functional requirements are expressed in these terms. When stakeholders can agree upon this language, at least within the scope of 'ELMTest', they share precisely enough language to have meaningful discussions about functional requirements. All definitions have been numbered for the sake of traceability.

2.1 RisicoManagementProcesje

Requirement 1: $Ifthereexist Assets called a, a', Allerisico's voorazijn-inhungezamenlijkheid-geaccepteerda'isequivalenttoDeeigenaarvanaheeftexplicietbeslotendatallerisico'smetbetrekkingtota'$

Requirement 2: $Ifthereexist Obligations called o, o', Hetrisicodatonietwordtnagekomenisgeaccepteerdo'is$

Requirement 3: $Ifthereexist LMH scalled l, l', lequals l'isequivalenttolequals l'orlequals l'orlequals l'$

2.2 Loose ends...

This paragraph shows remaining fact types and concepts that have not been used in previous paragraphs.

Chapter 3

Diagnosis

This chapter provides an analysis of the Ampersand script of 'ELMTest'. This analysis is intended for the authors of this script. It can be used to complete the script or to improve possible flaws.

ELMTest does not assign rules to roles. Therefore we define a generic role, User, to do all the work that is necessary in the business process. ELMTest assigns rules to roles. The following table shows the rules that are being maintained by a given role.

rule	AssetOwner	SecurityOfficer
acceptatie van asset risicos		
acceptatie van individuele risicos		
lmh_atoms		

Concepts Asset, Obligation, LMH, and Person remain without a definition.

Relations *assetManager*, *statAssetRA*, *dAssetRA*, *obligationOf*, $V_{[ONE \times Asset]}$, *oblRisk*, *dOblRA*, *statOblRA*, $V_{[LMH \times Obligation]}$, $'L'_{[LMH \times LMH]}$, *lth*, $'H'_{[LMH \times LMH]}$, and $'M'_{[LMH \times LMH]}$ are explained by an automated explanation generator. If these explanations are not appropriate, add PURPOSE RELATION statements to your relations.

Chapter 4

Conceptual Analysis

This chapter provides an analysis of the principles described in chapter 2. Each section in that chapter is analysed in terms of relations and each principle is then translated in a rule.

4.1 RisicoManagementProcesje

Figure 4.1 shows a conceptual diagram of this theme.

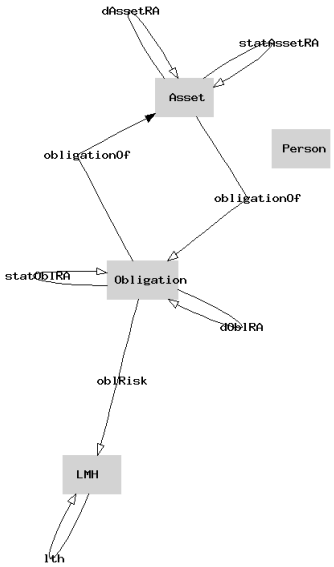

Figure 4.1: Conceptual model of RisicoManagementProcesje

acceptatie van asset risicos To arrive at the formalization in equation 4.12, the following five relations are introduced.

$$\begin{aligned} statAssetRA &: Asset \times Asset && (4.1)\\ obligationOf &: Obligation \rightarrow Asset && (4.2)\\ statOblRA &: Obligation \times Obligation && (4.3)\\ obligationOf &: Obligation \rightarrow Asset && (4.4)\\ dAssetRA &: Asset \times Asset && (4.5) \end{aligned}$$

A signal is produced when:

$$statAssetRA = dAssetRA \cup I_{[Asset \times Asset]} \cap obligationOf\ ;\ obligationOf \cap \overline{obligationOf}\ \dagger statOblRA; obliga \quad (4.6)$$

This corresponds to requirement 2.1 on page 3.

acceptatie van individuele risicos To arrive at the formalization in equation 4.13, the following three relations are introduced.

$$\begin{aligned} lth &: LMH \times LMH && (4.7)\\ oblRisk &: Obligation \times LMH && (4.8)\\ dOblRA &: Obligation \times Obligation && (4.9) \end{aligned}$$

We use definition 4.3 (statOblRA). A signal is produced when:

$$statOblRA = dOblRA \cup I_{[Obligation \times Obligation]} \cap oblRisk; (I_{[LMH \times LMH]} \cup lth);' L'_{[LMH \times LMH]}; V_{[LMH \times Obli} \quad (4.10)$$

This corresponds to requirement 2.1 on page 3.

lmh_atoms A signal is produced when:

$$I_{[LMH \times LMH]} = {}' L'_{[LMH \times LMH]} \cup' M'_{[LMH \times LMH]} \cup' H'_{[LMH \times LMH]} \quad (4.11)$$

acceptatie van asset risicos We also use definitions 4.3($statOblRA$), 4.1($statAssetRA$), 4.4($obligationOf$), 4.4($obligationOf$), and 4.5($dAssetRA$). A signal is produced when:

$$statAssetRA = dAssetRA \cup I_{[Asset \times Asset]} \cap obligationOf\ ;\ obligationOf \cap \overline{obligationOf}\ \dagger statOblRA; obliga \quad (4.12)$$

acceptatie van individuele risicos We also use definitions 4.3($statOblRA$), 4.7(lth), 4.8($oblRisk$), and 4.9($dOblRA$). A signal is produced when:

$$statOblRA = dOblRA \cup I_{[Obligation \times Obligation]} \cap oblRisk; (I_{[LMH \times LMH]} \cup lth);' L'_{[LMH \times LMH]}; V_{[LMH \times Obli} \quad (4.13)$$

lmh_atoms A signal is produced when:

$$I_{[LMH \times LMH]} = {}' L'_{[LMH \times LMH]} \cup' M'_{[LMH \times LMH]} \cup' H'_{[LMH \times LMH]} \quad (4.14)$$

Chapter 5

Process Analysis

ELMTest does not assign rules to roles. Therefore we define a generic role, User, to do all the work that is necessary in the business process. ELMTest assigns services to roles. The following table shows the services that are available to a given role.

Role	Service
AssetOwner	decideAssetRisks decideObligationRisks decideAllRisks
SecurityOfficer	decideAllRisks estimateRisk1 estimateRisk2 estimateRisk3
	Asset1 Assets1 Assets2 Person LMH Assets Obligations Persons LMHs

ELMTest assigns roles to relations. The following table shows the relations, the content of which can be altered by anyone who fulfills a given role.

Role	Relation
AssetOwner	$dAssetRA$ $dOblRA$
SecurityOfficer	$oblRisk$
	$assetManager : Asset \times Person$ $obligationOf : Obligation \times Asset$ $statAssetRA : Asset \times Asset$ $statOblRA : Obligation \times Obligation$ $lth : LMH \times LMH$

Chapter 6

Data structure

The requirements, which are listed in chapter 2, have been translated into the data model in figure 6.2. There are two data sets, one association, no generalisations, and no aggregations. ELMTest has a total of four concepts.

Figure 6.1: Classification of ELMTest

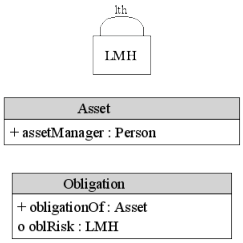

Figure 6.2: Conceptual data model of ELMTest

ELMTest has the following associations and multiplicity constraints.

relation	total	surjective
obligationOf		√

Additionally, the homogeneous relations come with the following properties :

relation	Rfx	Irf	Trn	Sym	Asy	Prop
statAssetRA				√	√	√
dAssetRA				√	√	√
dOblRA				√	√	√
statOblRA				√	√	√
lth			√		√	

6.1 Obligation

The attributes in Obligation have the following multiplicity constraints.

attribute	type	mandatory	unique
obligationOf	Asset	√	√
oblRisk	LMH		√
statOblRA	Obligation		√
dOblRA	Obligation		√

6.2 Asset

The attributes in Asset have the following multiplicity constraints.

attribute	type	mandatory	unique
assetManager	Person	√	√
statAssetRA	Asset		√
dAssetRA	Asset		√

6.3 lth

The attributes in lth have the following multiplicity constraints.

attribute	type	mandatory	unique
tLMH	LMH	√	√

Chapter 7

Asset1

Service Asset1 gives users the functionality to work wit assets. Service Asset1 is not used by any role.Service Asset1 operates from one instance of Asset. In order to maintain rules, instances of Asset may be created or deleted by this service automatically. Rules acceptatie van asset risicos and acceptatie van individuele risicos are being maintained by this service, without intervention of the user. Figure 7.1 shows the knowledge graph of this service.

Figure 7.1: Language diagram of Asset1

This service has the following fields.

- manager(verplicht)
  display on start: $I_{[Asset \times Asset]}$
- accepted status
  display on start: $I_{[Asset \times Asset]}$
- manually accepted
  display on start: $I_{[Asset \times Asset]}$
- obligations(lijst)
  display on start: $I_{[Asset \times Asset]}$

Chapter 8

Assets1

Service Assets1 gives users the functionality to work wit ONEs. Service Assets1 is not used by any role.Service Assets1 operates from one instance of ONE. In order to maintain rules, instances of Asset may be deleted automatically by this service. Rules acceptatie van asset risicos and acceptatie van individuele risicos are being maintained by this service, without intervention of the user. Figure 8.1 shows the knowledge graph of this service.

Figure 8.1: Language diagram of Assets1

This service has one field:

Assets(lijst, verplicht)

Chapter 9

Assets2

Service Assets2 gives users the functionality to work wit assets. Service Assets2 is not used by any role.Service Assets2 operates from one instance of Asset.If there exists a Asset called o, If there exists a Asset called a', True and not(Alle risico's voor a' zijn - in hun gezamenlijkheid - geaccepteerda) In order to maintain rules, instances of Asset may be deleted automatically by this service. Rules acceptatie van asset risicos and acceptatie van individuele risicos are being maintained by this service, without intervention of the user. Figure 9.1 shows the knowledge graph of this service.

Figure 9.1: Language diagram of Assets2

This service has the following fields.

- accepted
  display on start: $\overline{statAssetRA}$
- obligations(lijst)
  display on start: $\overline{statAssetRA}$

Chapter 10

decideAssetRisks

Service decideAssetRisks gives users the functionality to work wit assets. Service decideAssetRisks is meant to be used by AssetOwner.Service decideAssetRisks operates from one instance of Asset.If there exists a Asset called o, If there exists a Asset called a', True and not(Alle risico's voor a' zijn - in hun gezamenlijkheid - geaccepteerda) In order to maintain rules, instances of Asset may be deleted automatically by this service. Rules acceptatie van asset risicos and acceptatie van individuele risicos are being maintained by this service, without intervention of the user. Figure 10.1 shows the knowledge graph of this service.

Figure 10.1: Language diagram of decideAssetRisks

This service has the following fields.

- accepted
  display on start: $\overline{statAssetRA}$
- obligations(lijst)
  display on start: $\overline{statAssetRA}$

Chapter 11

decideObligationRisks

Service decideObligationRisks gives users the functionality to work wit obligations. Service decideObligationRisks is meant to be used by AssetOwner.Service decideObligationRisks operates from one instance of Obligation. In order to maintain rules, instances of Obligation may be deleted automatically by this service. Rules acceptatie van individuele risicos and lmh_atoms are being maintained by this service, without intervention of the user. Figure 11.1 shows the knowledge graph of this service.

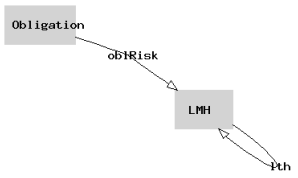

Figure 11.1: Language diagram of decideObligationRisks

This service has the following fields.

- risk
  display on start: $I_{[Obligation \times Obligation]}$
- accepted
  display on start: $I_{[Obligation \times Obligation]}$

Chapter 12

decideAllRisks

Service decideAllRisks gives users the functionality to work wit assets. Service decideAllRisks is meant to be used by AssetOwner and SecurityOfficer.Service decideAllRisks operates from one instance of Asset.If there exists a Asset called o, If there exists a Asset called a', True and not(Alle risico's voor a' zijn - in hun gezamenlijkheid - geaccepteerda) In order to maintain rules, instances of s Asset and Obligation may be deleted automatically by this service. Rules acceptatie van asset risicos, acceptatie van individuele risicos, and lmh_atoms are being maintained by this service, without intervention of the user. Figure 12.1 shows the knowledge graph of this service.

Figure 12.1: Language diagram of decideAllRisks

This service has the following fields.

- accepted
  display on start: $\overline{statAssetRA}$
- obligations(lijst)
  display on start: $\overline{statAssetRA}$
  - risk
    display on start: $\overline{statAssetRA}$; $oblRisk$
  - accepted
    display on start: $\overline{statAssetRA}$; $dOblRA$

Chapter 13

estimateRisk1

Service estimateRisk1 gives users the functionality to work wit assets. Service estimateRisk1 is meant to be used by SecurityOfficer.Service estimateRisk1 operates from one instance of Asset. In order to maintain rules, instances of Asset may be deleted automatically by this service. Rules acceptatie van asset risicos and acceptatie van individuele risicos are being maintained by this service, without intervention of the user. Figure 13.1 shows the knowledge graph of this service.

Figure 13.1: Language diagram of estimateRisk1

This service has the following fields.

- accepted
  display on start: $I_{[Asset \times Asset]}$
- obligations(lijst)
  display on start: $I_{[Asset \times Asset]}$

Chapter 14

estimateRisk2

Service estimateRisk2 gives users the functionality to work wit obligations. Service estimateRisk2 is meant to be used by SecurityOfficer.Service estimateRisk2 operates from one instance of Obligation. In order to maintain rules, instances of Obligation may be deleted automatically by this service. Rules acceptatie van individuele risicos and lmh_atoms are being maintained by this service, without intervention of the user. Figure 14.1 shows the knowledge graph of this service.

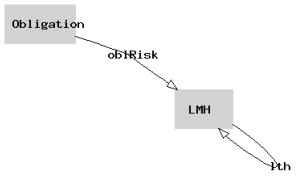

Figure 14.1: Language diagram of estimateRisk2

This service has the following fields.

- risk
  display on start: $I_{[Obligation \times Obligation]}$
- accepted
  display on start: $I_{[Obligation \times Obligation]}$

Chapter 15

estimateRisk3

Service estimateRisk3 gives users the functionality to work wit assets. Service estimateRisk3 is meant to be used by SecurityOfficer.Service estimateRisk3 operates from one instance of Asset. In order to maintain rules, instances of s Asset and Obligation may be deleted automatically by this service. Rules acceptatie van asset risicos, acceptatie van individuele risicos, and lmh_atoms are being maintained by this service, without intervention of the user. Figure 15.1 shows the knowledge graph of this service.

Figure 15.1: Language diagram of estimateRisk3

This service has the following fields.

- accepted
  display on start: $I_{[Asset \times Asset]}$
- obligations(lijst)
  display on start: $I_{[Asset \times Asset]}$
  - risk
    display on start: *oblRisk*
  - accepted
    display on start: *dOblRA*

Chapter 16

Person

Service Person gives users the functionality to work wit persons. Service Person is not used by any role.Service Person operates from one instance of Person. In this service, no objects are made or removed automatically. Rule acceptatie van asset risicos is being maintained by this service, without intervention of the user. Figure 16.1 shows the knowledge graph of this service.

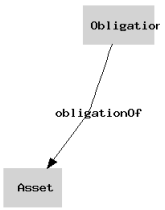

Figure 16.1: Language diagram of Person

This service has one field:

assetManager∼(lijst)

Chapter 17

LMH

Service LMH gives users the functionality to work wit LMHs. Service LMH is not used by any role.Service LMH operates from one instance of LMH. In this service, no objects are made or removed automatically. Rule acceptatie van individuele risicos is being maintained by this service, without intervention of the user. Figure 17.1 shows the knowledge graph of this service.

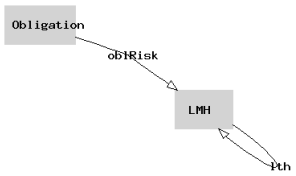

Figure 17.1: Language diagram of LMH

This service has one field:

oblRisk∼(lijst)

Chapter 18

Assets

Service Assets gives users the functionality to work wit ONEs. Service Assets is not used by any role.Service Assets operates from one instance of ONE. In this service, no objects are made or removed automatically. Figure 18.1 shows the knowledge graph of this service.

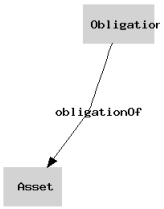

Figure 18.1: Language diagram of Assets

This service has one field:

Asset(lijst, verplicht)

Chapter 19

Obligations

Service Obligations gives users the functionality to work wit ONEs. Service Obligations is not used by any role.Service Obligations operates from one instance of ONE. In this service, no objects are made or removed automatically. Figure 19.1 shows the knowledge graph of this service.

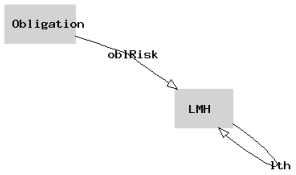

Figure 19.1: Language diagram of Obligations

This service has one field:

Obligation(lijst, verplicht)

Chapter 20

Persons

Service Persons gives users the functionality to work wit ONEs. Service Persons is not used by any role.Service Persons operates from one instance of ONE. In this service, no objects are made or removed automatically. Figure 20.1 shows the knowledge graph of this service.

Figure 20.1: Language diagram of Persons

This service has one field:

Person(lijst, verplicht)

Chapter 21

LMHs

Service LMHs gives users the functionality to work wit ONEs. Service LMHs is not used by any role.Service LMHs operates from one instance of ONE. In this service, no objects are made or removed automatically. Figure 21.1 shows the knowledge graph of this service.

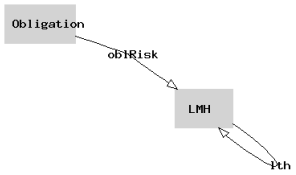

Figure 21.1: Language diagram of LMHs

This service has one field:

LMH(lijst, verplicht)