Functional Specification of ‘VIRO’

Put author(s) here
(This document was generated by Ampersand vs. 2.1.0.71)

Thu May 26 10:14:29 EDT 2011

Contents

Chapter 1

Introduction

This document defines the functionality of an information system called 'VIRO'. 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 'VIRO' as their intended audience. The data analysis in chapter 7 describes the data sets upon which 'VIRO' 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, 'VIRO' 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 ‘VIRO’ can be discussed and expressed. The purpose of this chapter is to create shared understanding among stakeholders. The language of ‘VIRO’ 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 ‘VIRO’, they share precisely enough language to have meaningful discussions about functional requirements. All definitions have been numbered for the sake of traceability.

2.1 CaseRegistration

The registration of administrative cases is based on three articles from the Dutch administrative law, 'Algemene wet bestuursrecht (Awb)'. An administrative case is a legal case against a decision of an administrative authority. Within the 'VIRO' context the terms 'case' and 'legal case' will always refer to an administrative case.

Article 1:1 Awb

1. 'Administrative authority' means:

    a. an organ of a juristic person governed by public law, or
    b. any other person or body vested with public authority.

Article 6:4 Awb

1. An objection is lodged by filing a notice of objection with the administrative authority which took the decision.
2. An administrative appeal is lodged by filing a notice of appeal with an appellate authority.

3. An appeal to an administrative court is lodged by filing a notice of appeal with that court.

Article 8:24 Awb

1. A party may be assisted or represented by an authorized representative.
2. The district court may require written authorization of a representative.
3. Paragraph 2 does not apply to members of the Dutch Bar.

Notices as referred to in art.6:4 Awb are filed in the case file. Written authorizations as referred to in 8:24 par.3 Awb are filed in the case file.

Requirement 1 (rule@line70): The plaintiff in an administrative case is a juristic person

Requirement 2 (rule@line73): The defendant in an administrative case is an administrative authority as referred to in art.1:1 Awb.

Requirement 3 (rule@line84): Written authorizations for representatives of a case are not put in the case file

Requirement 4 (rule@line92): Every administrative case is either an appeal or an objection or an appeal to an administrative court. (Art.6:4 Awb)

Requirement 5 (rule@line103): Every party is either a person or an organization or an administrative authority.

Requirement 6 (rule@line107): Members of the government, i.e., Ministers and Secretaries of State, are administrative authorities according to the constitution.

2.2 Sessions

When administrative cases are brought before the district court, they are assigned to a scheduled session with a certain panel of judges. The registration of sessions is based on article 8:10 Awb.

Article 8:10 Awb

1. Cases brought before the district court shall first be considered by a single-judge panel.
2. If the single-judge panel deems a case unsuitable for consideration by a single judge, it will refer it to a three-judge panel. A single-judge panel may also refer a case to a three-judge panel for other reasons.

3. If a three-judge panel thinks that a case is suitable for further consideration by a single judge, it may refer it to a single-judge panel.
4. A case may be referred at any stage of the proceedings. The case shall then be resumed at the point at which it was referred.

The organization of panels is defined in the Dutch law for the administration of justice, 'Wet op de rechterlijke organisatie (RO)'. For example article 6:1 RO describes that there are single-judge and three-judge panels, of which the members are recognized judges. Article 41 par.1 RO defines that the district court is seated in the main city of the jurisdiction. Article 41 par.2 RO allows for local offices of the district court.

For illustration purposes, article 8:7 par.1 Awb will be maintained within the 'VIRO' context.

Requirement 7 (rule@line123): a session can be identified by its panel, its city and its date.

Requirement 8 (rule@line135): A judge at a session is a member of the panel that runs the session.

Requirement 9 (rule@line137): The clerk of a session must be the clerk of the court where the session is held.

Requirement 10 (rule@line143): All sessions are scheduled

Requirement 11 (rule@line149): Administrative authorities as referred to in art.8:7 par.1 Awb are administrative authorities as referred to in art.1:1 Awb.

Requirement 12 (rule@line151): An appeal lodged against a decision of an administrative authority of a province or municipality, or a water management board, or a region as referred to in article 21 of the 1993 Police Act, or of a joint body or public body established under the Joint Arrangements Act, falls within the jurisdiction of the district court within whose district the administrative authority has its seat. (art. 8:7 par.1 Awb.)

2.3 CaseFiles

This paragraph describes the theme 'CaseFiles'.

Requirement 13 (file documents): Documents are put in a case file except for written authorizations for representatives.

2.4 Scheduling

This paragraph describes the theme ‘Scheduling’.

Requirement 14 (assign cases): Cases must be assigned to a process in a scheduled session.

Chapter 3

Diagnosis

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

VIRO does not specify which roles may change the contents of which relations. VIRO assigns rules to roles. The following table shows the rules that are being maintained by a given role.

rule	Archivist	Scheduler
file documents	×	
assign cases		×

Concepts Party, LegalCase, City, Document, AreaOfLaw, DocumentType, CaseType, Process, Session, Panel, Court, Role, Date, CourtOfAppeal, TimeStamp, and Text remain without a definition.

The purpose of relations *caseFile*, *documentType*, *remark*, *sent*, *received*, *areaOfLaw*, *caseType*, *objection*, *appealToAdminCourt*, *appeal*, *writtenAuthOf*, *plaintiff*, *authFor*, *authBy*, *defendant*, *legalCase*, *session*, *panel*, *scheduled*, *judge*, $clerk_{[Session \times Party]}$, *broughtBefore*, *location*, $seatedIn_{[Court \times City]}$, *occured*, *jurisdiction*, *district*, *court*, *members*, $clerk_{[Court \times Party]}$, *localOffices*, *actsas*, *organization*, *person*, *administrativeAuthority*, *memberOfGovernment*, *administrativeAuthorityAwb87*, and *domicile* is not documented.

Relations *writtenAuthOf*, *authBy*, *areaOfLaw*, *documentType*, *caseType*, *session*, *court*, *actsas*, $seatedIn_{[Court \times City]}$, $seatedIn_{[CourtOfAppeal \times City]}$, *district*, *localOffices*, *sent*, *received*, and *remark* are not used in any rule.

Figure 3.1 shows a conceptual diagram with all relations declared in 'CaseRegistration'.

Figure 3.2 shows a conceptual diagram with all relations declared in 'Correspondence'.

On line numbers line 70, file "fsVIROENG.adl", line 73, file "fsVIROENG.adl", line 84, file "fsVIROENG.adl", line 92, file "fsVIROENG.adl", line 103, file

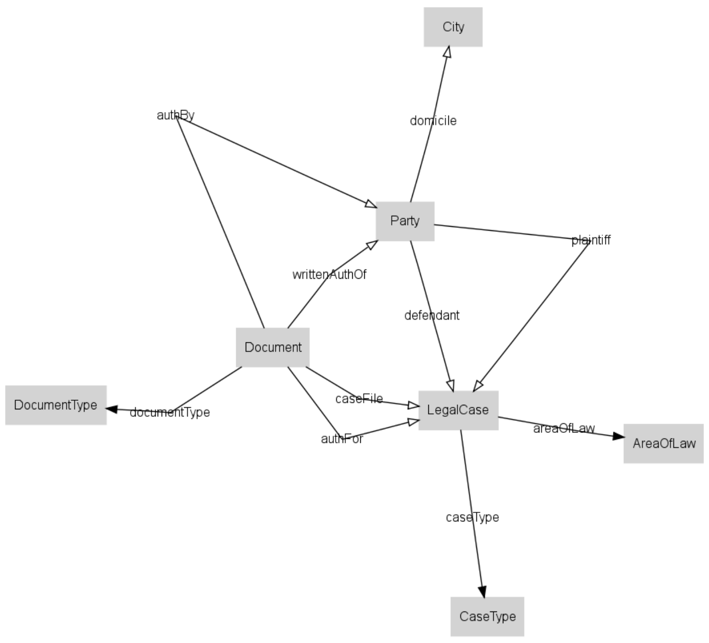


Figure 3.1: Concept analysis of relations in CaseRegistration

"fsVIROENG.adl", line 107, file "fsVIROENG.adl", line 123, file "fsVIROENG.adl", line 135, file "fsVIROENG.adl", line 137, file "fsVIROENG.adl", line 143, file "fsVIROENG.adl", line 149, file "fsVIROENG.adl", line 151, file "fsVIROENG.adl", line 112, file "fsVIROENG.adl", and line 163, file "fsVIROENG.adl" of file fsVIROENG.adl, rules are defined without a proper explanation of their purpose.

All rules in all processes are linked to roles.

The role-rule assignments in any of the described processes have been assigned to rules within that same process.

The following table represents the population of various relations.

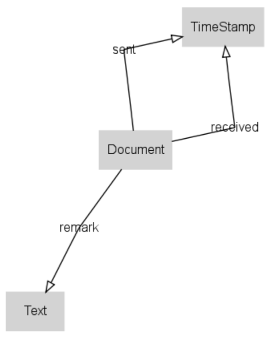


Figure 3.2: Concept analysis of relations in Correspondence

Concept	Population
Party	38
LegalCase	3
City	59
Document	10
AreaOfLaw	1
DocumentType	5
CaseType	2
Process	3
Session	4
Panel	7
Court	20
Role	7
Date	4
CourtOfAppeal	6
TimeStamp	18
Text	10

Relation	Population
plaintiff : *Party* × *LegalCase*	3
defendant : *Party* × *LegalCase*	3
domicile : *Party* × *City*	4
writtenAuthOf : *Document* × *Party*	5
authFor : *Document* × *LegalCase*	4
authBy : *Document* × *Party*	4
areaOfLaw : *LegalCase* × *AreaOfLaw*	3
caseFile : *Document* × *LegalCase*	5
documentType : *Document* × *DocumentType*	10
caseType : *LegalCase* × *CaseType*	3
appeal : *LegalCase* × *LegalCase*	2
appealToAdminCourt : *LegalCase* × *LegalCase*	1
person : *Party* × *Party*	33
organization : *Party* × *Party*	2
administrativeAuthority : *Party* × *Party*	3
memberOfGovernment : *Party* × *Party*	1
session : *Process* × *Session*	3
legalCase : *Process* × *LegalCase*	3
panel : *Session* × *Panel*	4
court : *Panel* × *Court*	7
members : *Party* × *Panel*	14
judge : *Session* × *Party*	4
clerk : *Session* × *Party*	4
clerk : *Court* × *Party*	4
actsas : *Party* × *Role*	38
broughtBefore : *LegalCase* × *Court*	3
scheduled : *Session* × *Date*	4
occured : *Session* × *Date*	2
administrativeAuthorityAwb87 : *Party* × *Party*	2
seatedIn : *Court* × *City*	20
seatedIn : *CourtOfAppeal* × *City*	6
location : *Session* × *Court*	4
district : *Court* × *CourtOfAppeal*	20
localOffices : *City* × *Court*	39
jurisdiction : *City* × *Court*	59
sent : *Document* × *TimeStamp*	10
received : *Document* × *TimeStamp*	10
remark : *Document* × *Text*	10

This script contains work in progress. The following table provides details with line numbers from the original script file.

rule	line#	#signals
file documents	112	1
assign cases	163	1

Rule ‘file documents’ says:

Documents are put in a case file except for written authorizations for representatives. This rule contains workArchivist) by ‘Document letter 2009/33-9854’. Rule ‘assign cases’ says:

Cases must be assigned to a process in a scheduled session. This rule contains workScheduler) by ‘LegalCase SBR 02/74331’. The population in this script violates no rule.

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 CaseRegistration

The registration of administrative cases is based on three articles from the Dutch administrative law, 'Algemene wet bestuursrecht (Awb)'. An administrative case is a legal case against a decision of an administrative authority. Within the 'VIRO' context the terms 'case' and 'legal case' will always refer to an administrative case.

Article 1:1 Awb

1. 'Administrative authority' means:

    a. an organ of a juristic person governed by public law, or
    b. any other person or body vested with public authority.

Article 6:4 Awb

1. An objection is lodged by filing a notice of objection with the administrative authority which took the decision.
2. An administrative appeal is lodged by filing a notice of appeal with an appellate authority.
3. An appeal to an administrative court is lodged by filing a notice of appeal with that court.

Article 8:24 Awb

1. A party may be assisted or represented by an authorized representative.
2. The district court may require written authorization of a representative.
3. Paragraph 2 does not apply to members of the Dutch Bar.

Notices as referred to in art.6:4 Awb are filed in the case file. Written authorizations as referred to in 8:24 par.3 Awb are filed in the case file.

Figure 4.1 shows a conceptual diagram of this theme.

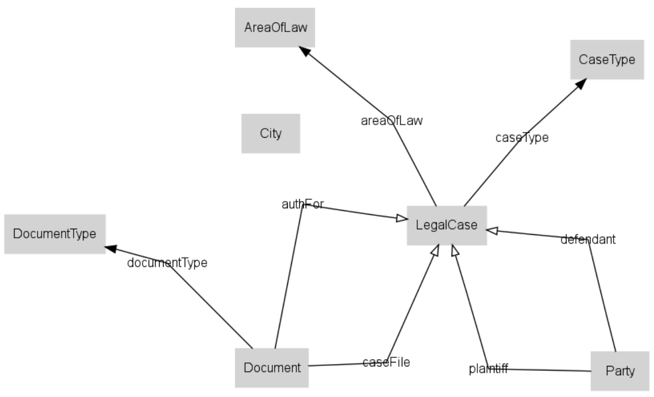


Figure 4.1: Conceptual model of CaseRegistration

rule@line70 To arrive at the formalization in equation 4.5, the following four relations are introduced.

$$\begin{aligned} \mathit{plaintiff} &: \mathit{Party}\times\mathit{LegalCase} && (4.1)\\ \mathit{plaintiff} &: \mathit{Party}\times\mathit{LegalCase} && (4.2)\\ \mathit{organization} &: \mathit{Party}\times\mathit{Party} && (4.3)\\ \mathit{person} &: \mathit{Party}\times\mathit{Party} && (4.4) \end{aligned}$$

This means:

$$\mathit{plaintiff}\,;\mathit{plaintiff}\ \cap I_{[\mathit{Party}]} \vdash \mathit{person}\cup\mathit{organization} \qquad (4.5)$$

This corresponds to requirement 2.1 on page 5.

rule@line73 To arrive at the formalization in equation 4.9, the following three relations are introduced.

$$\begin{aligned} \mathit{defendant} &: \mathit{Party}\times\mathit{LegalCase} && (4.6)\\ \mathit{defendant} &: \mathit{Party}\times\mathit{LegalCase} && (4.7)\\ \mathit{administrativeAuthority} &: \mathit{Party}\times\mathit{Party} && (4.8) \end{aligned}$$

This means:

$$\mathit{defendant};\mathit{defendant}\ \cap I_{[\mathit{Party}]} \vdash \mathit{administrativeAuthority} \qquad (4.9)$$

This corresponds to requirement 2.1 on page 5.

rule@line84 To arrive at the formalization in equation 4.12, the following two relations are introduced.

$$\begin{aligned} authFor &: Document \times LegalCase && (4.10) \\ caseFile &: Document \times LegalCase && (4.11) \end{aligned}$$

This means:

$$authFor \cap caseFile \vdash -V \tag{4.12}$$

This corresponds to requirement 2.1 on page 5.

rule@line92 To arrive at the formalization in equation 4.16, the following three relations are introduced.

$$\begin{aligned} objection &: LegalCase \times LegalCase && (4.13) \\ appealToAdminCourt &: LegalCase \times LegalCase && (4.14) \\ appeal &: LegalCase \times LegalCase && (4.15) \end{aligned}$$

This means:

$$I_{[LegalCase]} \vdash (appeal \cup appealToAdminCourt \cup objection) \cap (\overline{appeal} \cup \overline{appealToAdminCourt} \cup \overline{objection}) \tag{4.16}$$

This corresponds to requirement 2.1 on page 5.

rule@line103 We use definitions 4.8 (*administrativeAuthority*), 4.3 (*organization*), and 4.4 (*person*). This means:

$$I_{[Party]} \vdash (person \cup organization \cup administrativeAuthority) \cap (\overline{person} \cup \overline{organization} \cup \overline{administrativeAutho} \tag{4.17}$$

rule@line107 In order to formalize this, a relation memberOfGovernment is introduced (4.18):

$$memberOfGovernment : Party \times Party \tag{4.18}$$

Beside that, we use definition 4.8 (administrativeAuthority) to formalize requirement 2.1 (page 5): This means:

$$memberOfGovernment \vdash administrativeAuthority \tag{4.19}$$

4.2 Sessions

When administrative cases are brought before the district court, they are assigned to a scheduled session with a certain panel of judges. The registration of sessions is based on article 8:10 Awb.

Article 8:10 Awb

1. Cases brought before the district court shall first be considered by a single-judge panel.

2. If the single-judge panel deems a case unsuitable for consideration by a single judge, it will refer it to a three-judge panel. A single-judge panel may also refer a case to a three-judge panel for other reasons.

3. If a three-judge panel thinks that a case is suitable for further consideration by a single judge, it may refer it to a single-judge panel.

4. A case may be referred at any stage of the proceedings. The case shall then be resumed at the point at which it was referred.

The organization of panels is defined in the Dutch law for the administration of justice, 'Wet op de rechterlijke organisatie (RO)'. For example article 6:1 RO describes that there are single-judge and three-judge panels, of which the members are recognized judges. Article 41 par.1 RO defines that the district court is seated in the main city of the jurisdiction. Article 41 par.2 RO allows for local offices of the district court.

For illustration purposes, article 8:7 par.1 Awb will be maintained within the 'VIRO' context.

Figure 4.2 shows a conceptual diagram of this theme.

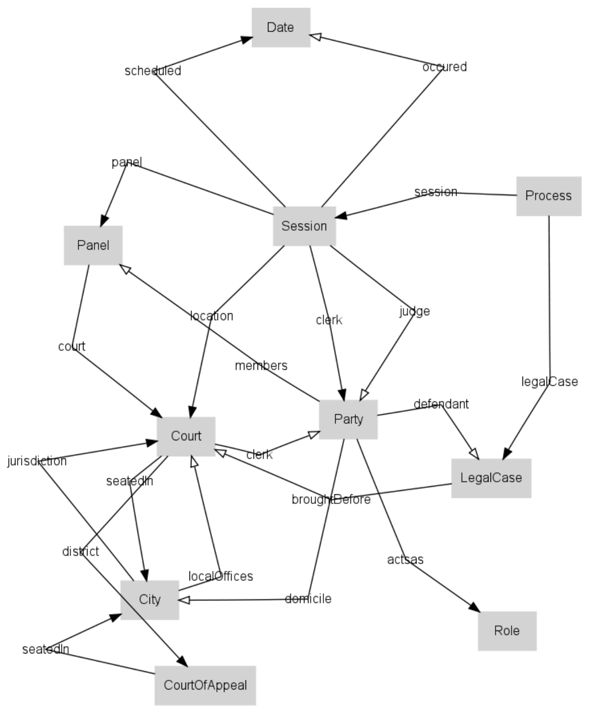


Figure 4.2: Conceptual model of Sessions

rule@line123 To arrive at the formalization in equation 4.26, the following six relations are introduced.

$$scheduled : Session \rightarrow Date \quad (4.20)$$
$$scheduled : Session \rightarrow Date \quad (4.21)$$
$$location : Session \rightarrow Court \quad (4.22)$$
$$location : Session \rightarrow Court \quad (4.23)$$
$$panel : Session \rightarrow Panel \quad (4.24)$$
$$panel : Session \rightarrow Panel \quad (4.25)$$

This means:

$$panel; panel \cap location; location \cap scheduled; scheduled \vdash I_{[Session]} \quad (4.26)$$

This corresponds to requirement 2.2 on page 6.

rule@line135 To arrive at the formalization in equation 4.29, the following two relations are introduced.

$$judge : Session \times Party \quad (4.27)$$
$$members : Party \times Panel \quad (4.28)$$

We also use definitions 4.25 (*panel*) and 4.25 (*panel*). This means:

$$judge \vdash panel; members \quad (4.29)$$

This corresponds to requirement 2.2 on page 6.

rule@line137 To arrive at the formalization in equation 4.32, the following two relations are introduced.

$$clerk : Session \rightarrow Party \quad (4.30)$$
$$clerk : Court \times Party \quad (4.31)$$

We also use definitions 4.23 (*location*) and 4.23 (*location*). This means:

$$clerk \vdash location; clerk \quad (4.32)$$

This corresponds to requirement 2.2 on page 6.

rule@line143 In order to formalize this, a relation occured is introduced (4.33):

$$occured : Session \times Date \quad (4.33)$$

We also use definitions 4.21 (*scheduled*) and 4.21 (*scheduled*) to formalize requirement 2.2 (page 6): This means:

$$occured \vdash scheduled \quad (4.34)$$

rule@line149 In order to formalize this, a relation administrativeAuthorityAwb87 is introduced (4.35):

$$administrativeAuthorityAwb87 \quad : \quad Party \times Party \tag{4.35}$$

Beside that, we use definition 4.8 (administrativeAuthority) to formalize requirement 2.2 (page 6): This means:

$$administrativeAuthorityAwb87 \vdash administrativeAuthority \tag{4.36}$$

rule@line151 To arrive at the formalization in equation 4.40, the following three relations are introduced.

$$jurisdiction \quad : \quad City \rightarrow Court \tag{4.37}$$
$$domicile \quad : \quad Party \times City \tag{4.38}$$
$$broughtBefore \quad : \quad LegalCase \times Court \tag{4.39}$$

We also use definitions 4.35 (*administrativeAuthorityAwb87*), 4.15 (*appeal*), 4.7 (*defendant*), and 4.7 (*defendant*). This means:

$$appeal; defendant^{\smile}; administrativeAuthorityAwb87; domicile; jurisdiction \vdash broughtBefore \tag{4.40}$$

This corresponds to requirement 2.2 on page 6.

4.3 Correspondence

Communication data of documents in case files may be registered.

Figure 4.3 shows a conceptual diagram of this theme.

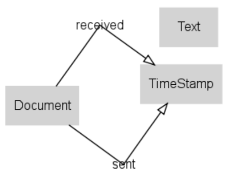


Figure 4.3: Conceptual model of Correspondence

Chapter 5

Process Analysis

VIRO does not specify which roles may change the contents of which relations. VIRO assigns rules to roles. The following table shows the rules that are being maintained by a given role.

Role	Rule
Archivist	file documents
Scheduler	assign cases

5.1 CaseFiles

Figure 5.1 shows the process model.

Figure 5.1: Process model of CaseFiles

The conceptual diagram of figure 5.2 provides an overview of the language in which this process is expressed.

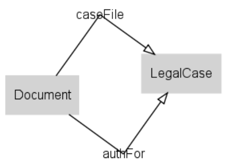


Figure 5.2: Basic sentences of CaseFiles

file documents We use definitions 4.10 (*authFor*) and 4.11 (*caseFile*). Activities that are defined by this rule are finished when:

$$I_{[Document]} \vdash caseFile; caseFile \cup authFor; authFor \tag{5.1}$$

These activities are signalled by:

$$I_{[Document]} \cap \overline{(caseFile; caseFile\ )} \cap \overline{(authFor; authFor\ )} \tag{5.2}$$

5.2 Scheduling

Figure 5.3 shows the process model.

assign cases

Figure 5.3: Process model of Scheduling

The conceptual diagram of figure 5.4 provides an overview of the language in which this process is expressed.

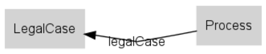


Figure 5.4: Basic sentences of Scheduling

assign cases We use definition **??** (legalCase). Activities that are defined by this rule are finished when:

$$I_{[LegalCase]} \vdash legalCase\ ; legalCase \tag{5.3}$$

These activities are signalled by:

$$I_{[LegalCase]} \cap \overline{(legalCase\ ; legalCase)} \tag{5.4}$$

Chapter 6

Function Point Analysis

The specification of 'VIRO' has been analysed by counting function points[?]. This has resulted in an estimated total of 112 function points.

data set	analysis	FP
Party	ILGV Eenvoudig	7
Session	ILGV Eenvoudig	7
LegalCase	ILGV Eenvoudig	7
Document	ILGV Eenvoudig	7
Court	ILGV Eenvoudig	7
Process	ILGV Eenvoudig	7
City	ILGV Eenvoudig	7
CourtOfAppeal	ILGV Eenvoudig	7
Panel	ILGV Eenvoudig	7
Text	ILGV Eenvoudig	7
TimeStamp	ILGV Eenvoudig	7
Date	ILGV Eenvoudig	7
Role	ILGV Eenvoudig	7
CaseType	ILGV Eenvoudig	7
DocumentType	ILGV Eenvoudig	7
AreaOfLaw	ILGV Eenvoudig	7

interface	analysis	FP
file documents	NO	0
assign cases	NO	0

Chapter 7

Data structure

The requirements, which are listed in chapter 2, have been translated into the data model in figure 7.2. There are 9 data sets, 12 associations, no generalisations, and no aggregations. VIRO has a total of 16 concepts.

Figure 7.1: Classification of VIRO

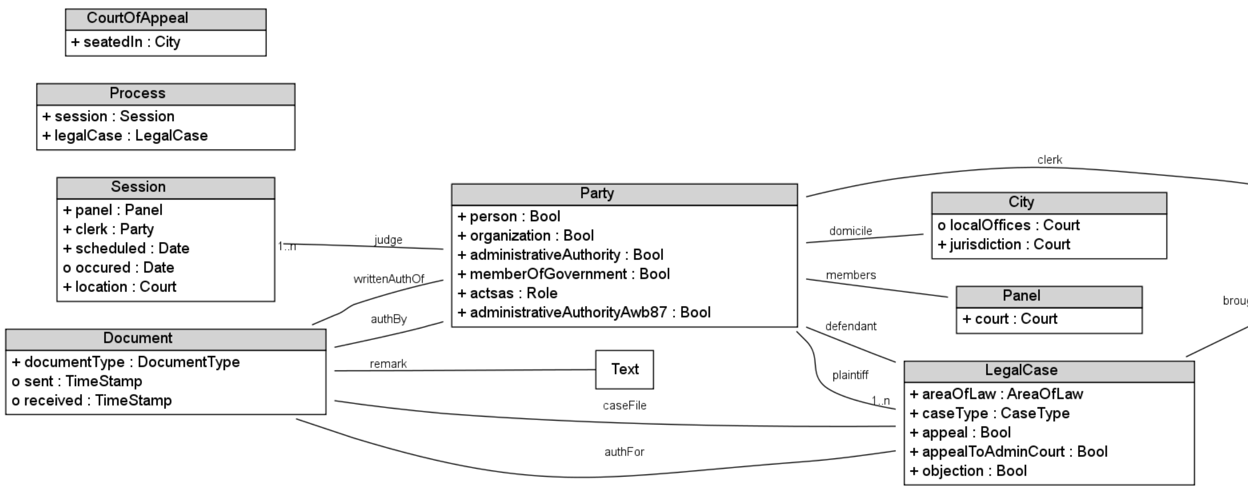


Figure 7.2: Data model of VIRO

VIRO has the following associations and multiplicity constraints.

relation	total	surjective
plaintiff		√
judge		√

$Additionally, theendorelationscomewiththefollowingproperties:$

relation	Rfx	Irf	Trn	Sym	Asy	Prop
objection				√	√	√
appealToAdminCourt				√	√	√
appeal				√	√	√
objection				√	√	√
appealToAdminCourt				√	√	√
appeal				√	√	√
organization				√	√	√
person				√	√	√
administrativeAuthority				√	√	√
memberOfGovernment				√	√	√
administrativeAuthorityAwb87				√	√	√

7.1 Party

The attributes in Party have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Party	√	√
person	Bool	√	
organization	Bool	√	
administrativeAuthority	Bool	√	
memberOfGovernment	Bool	√	
actsas	Role	√	
administrativeAuthorityAwb87	Bool	√	

Within this data set, the following integrity rules shall be true at all times.

- $$I_{[Party]} \vdash (person \cup organization \cup administrativeAuthority) \cap \overline{(person \cap organization \cap administrativeAutho}$$

- $$memberOfGovernment \vdash administrativeAuthority$$

- $$administrativeAuthorityAwb87 \vdash administrativeAuthority$$

The following rule defines the integrity of data within this data set. It must remain true at all times.

$$\overline{I_{[Party]}} \cup (person \cup organization \cup administrativeAuthority) \cap (\overline{person} \cup \overline{organization} \cup \overline{administrativeAuthority})$$

7.2 Session

The attributes in Session have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Session	✓	✓
panel	Panel	✓	
clerk	Party	✓	
scheduled	Date	✓	
occured	Date		
location	Court	✓	

Within this data set, the following integrity rule shall be true at all times.

$$occured \vdash scheduled$$

7.3 LegalCase

The attributes in LegalCase have the following multiplicity constraints.

attribute	type	mandatory	unique
key	LegalCase	✓	✓
areaOfLaw	AreaOfLaw	✓	
caseType	CaseType	✓	
appeal	Bool	✓	
appealToAdminCourt	Bool	✓	
objection	Bool	✓	

Within this data set, the following integrity rule shall be true at all times.

$$I_{[LegalCase]} \vdash (appeal \cup appealToAdminCourt \cup objection) \cap \overline{(appeal \cap appealToAdminCourt \cap objection)}$$

The following rules define the integrity of data within this data set. They must remain true at all times.

- $$\overline{I_{[LegalCase]}} \cup (appeal \cup appealToAdminCourt \cup objection) \cap (\overline{appeal} \cup \overline{appealToAdminCourt} \cup \overline{objection})$$
- $$\overline{I_{[LegalCase]}} \cup legalCase\ ; legalCase$$

7.4 Document

The attributes in Document have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Document	√	√
documentType	DocumentType	√	
sent	TimeStamp		
received	TimeStamp		

The following rule defines the integrity of data within this data set. It must remain true at all times.

$$\overline{I_{[Document]}} \cup caseFile; caseFile \ \cup authFor; authFor$$

7.5 Court

The attributes in Court have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Court	√	√
seatedIn	City	√	
district	CourtOfAppeal	√	

7.6 Process

The attributes in Process have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Process	√	√
session	Session	√	
legalCase	LegalCase	√	

7.7 City

The attributes in City have the following multiplicity constraints.

attribute	type	mandatory	unique
key	City	√	√
localOffices	Court		
jurisdiction	Court	√	

7.8 CourtOfAppeal

The attributes in CourtOfAppeal have the following multiplicity constraints.

attribute	type	mandatory	unique
key	CourtOfAppeal	√	√
seatedIn	City	√	

7.9 Panel

The attributes in Panel have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Panel	√	√
court	Court	√	

7.10 plaintiff

The attributes in plaintiff have the following multiplicity constraints.

attribute	type	mandatory	unique
key	LegalCase	√	
Party	Party	√	

The following rules define the integrity of data within this data set. They must remain true at all times.

- $$\overline{I_{[LegalCase]}} \cup (appeal \cup appealToAdminCourt \cup objection) \cap (\overline{appeal} \cup \overline{appealToAdminCourt} \cup \overline{objection})$$
- $$\overline{I_{[LegalCase]}} \cup legalCase\ ; legalCase$$

7.11 defendant

The attributes in defendant have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Party	√	
LegalCase	LegalCase	√	

7.12 domicile

The attributes in domicile have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Party	√	
City	City	√	

7.13 writtenAuthOf

The attributes in writtenAuthOf have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Document	√	
Party	Party	√	

7.14 authFor

The attributes in authFor have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Document	√	
LegalCase	LegalCase	√	

7.15 authBy

The attributes in authBy have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Document	√	
Party	Party	√	

7.16 caseFile

The attributes in caseFile have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Document	√	
LegalCase	LegalCase	√	

7.17 members

The attributes in members have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Party	✓	
Panel	Panel	✓	

7.18 judge

The attributes in judge have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Session	✓	
Party	Party	✓	

7.19 clerk

The attributes in clerk have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Court	✓	
Party	Party	✓	

7.20 broughtBefore

The attributes in broughtBefore have the following multiplicity constraints.

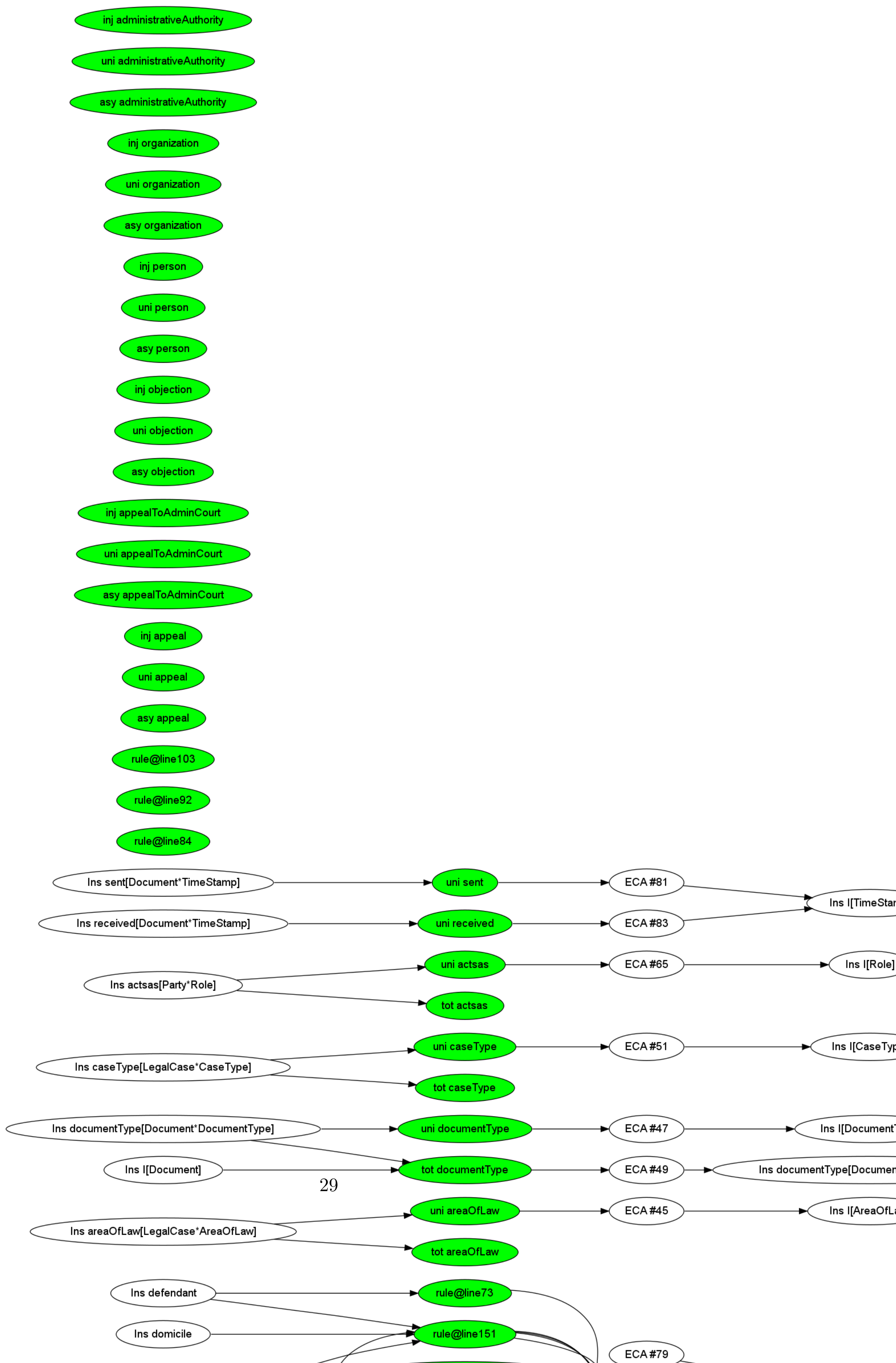
attribute	type	mandatory	unique
key	LegalCase	✓	
Court	Court	✓	

7.21 remark

The attributes in remark have the following multiplicity constraints.

attribute	type	mandatory	unique
key	Document	✓	
Text	Text	✓	

Figure 7.3 shows the switchboard diagram.This is used in designing the database functionality.



inj administrativeAuthority
uni administrativeAuthority
asy administrativeAuthority
inj organization
uni organization
asy organization
inj person
uni person
asy person
inj objection
uni objection
asy objection
inj appealToAdminCourt
uni appealToAdminCourt
asy appealToAdminCourt
inj appeal
uni appeal
asy appeal
rule@line103
rule@line92
rule@line84
Ins sent[Document*TimeStamp]
uni sent
ECA #81
Ins I[TimeStar
Ins received[Document*TimeStamp]
uni received
ECA #83
Ins actsas[Party*Role]
uni actsas
ECA #65
Ins I[Role]
tot actsas
Ins caseType[LegalCase*CaseType]
uni caseType
ECA #51
Ins I[CaseTyp
tot caseType
Ins documentType[Document*DocumentType]
uni documentType
ECA #47
Ins I[Document
Ins I[Document]
tot documentType
ECA #49
Ins documentType[Documen
Ins areaOfLaw[LegalCase*AreaOfLaw]
uni areaOfLaw
ECA #45
Ins I[AreaOfLa
tot areaOfLaw
Ins defendant
rule@line73
Ins domicile
rule@line151
ECA #79

Chapter 8

file documents

For what purpose activity ‘file documents’ exists remains undocumented. Activity ‘file documents’ must be performed by a user with role Archivist. During that activity, rule ‘rule@line84’ will be maintained without intervention of a user. The following table shows which edit actions invoke which function.

action	relation	rule
Ins	authFor[Document*LegalCase]	error: rule ‘rule@line84’
Del	authFor[Document*LegalCase]	error: rule ‘rule@line84’
Ins	caseFile[Document*LegalCase]	error: rule ‘rule@line84’
Del	caseFile[Document*LegalCase]	error: rule ‘rule@line84’
Ins	I	ECA rule 49
Del	I	error: rule ‘tot documentType’

Figure 8.1 shows the knowledge graph of this interface.

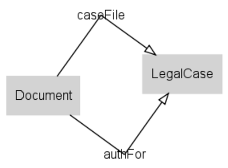


Figure 8.1: Language diagram of file documents

Every section in this chapter describes one activity. While performing an activity, users will insert or delete population in various relations. This may potentially violate invariants. (An invariant is a business rule rules that must remain true at all times.) The software to maintain the truth of invariant rules is generated automatically. The structure of that software is illustrated by a so called switchboard diagram, the first of which you will find in figure X. Each switchboard diagram consists of three columns: Invariant rules are drawn in the middle, and relations occur on the (right and left hand) sides. An arrow on the left hand side points from a relation that may be edited to a rule that may be violated as a consequence thereof. Each arrow on the right hand side of a rule represents an edit action that is required to restore its truth. It points to the relation that is edited for that purpose. If that arrow is labeled '+', it involves an insert event; if labeled '-' it refers to a delete event. This yields an accurate perspective on the way in which invariants are maintained.

Figure 8.2 shows the switchboard diagram of this interface.

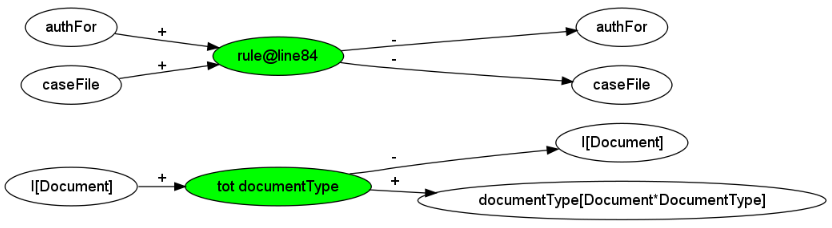


Figure 8.2: Switchboard of file documents

Chapter 9

assign cases

For what purpose activity ‘assign cases’ exists remains undocumented. Activity ‘assign cases’ must be performed by a user with role Scheduler. During that activity, rule ‘rule@line92’ will be maintained without intervention of a user. The following table shows which edit actions invoke which function.

action	relation	rule
Ins	I	error: rule ‘asy appeal’, ‘asy appealToAdminCourt’, and ‘asy obje
Del	I	error: rule ‘rule@line92’, ‘sur plaintiff’, ‘tot areaOfLaw’, and ‘tot
Ins	legalCase[Process*LegalCase]	ECA rule 57
Del	legalCase[Process*LegalCase]	error: rule ‘uni legalCase’

Figure 9.1 shows the knowledge graph of this interface.

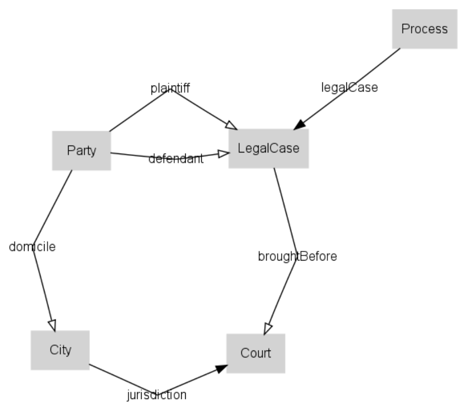


Figure 9.1: Language diagram of assign cases

Figure 9.2 shows the switchboard diagram of this interface.

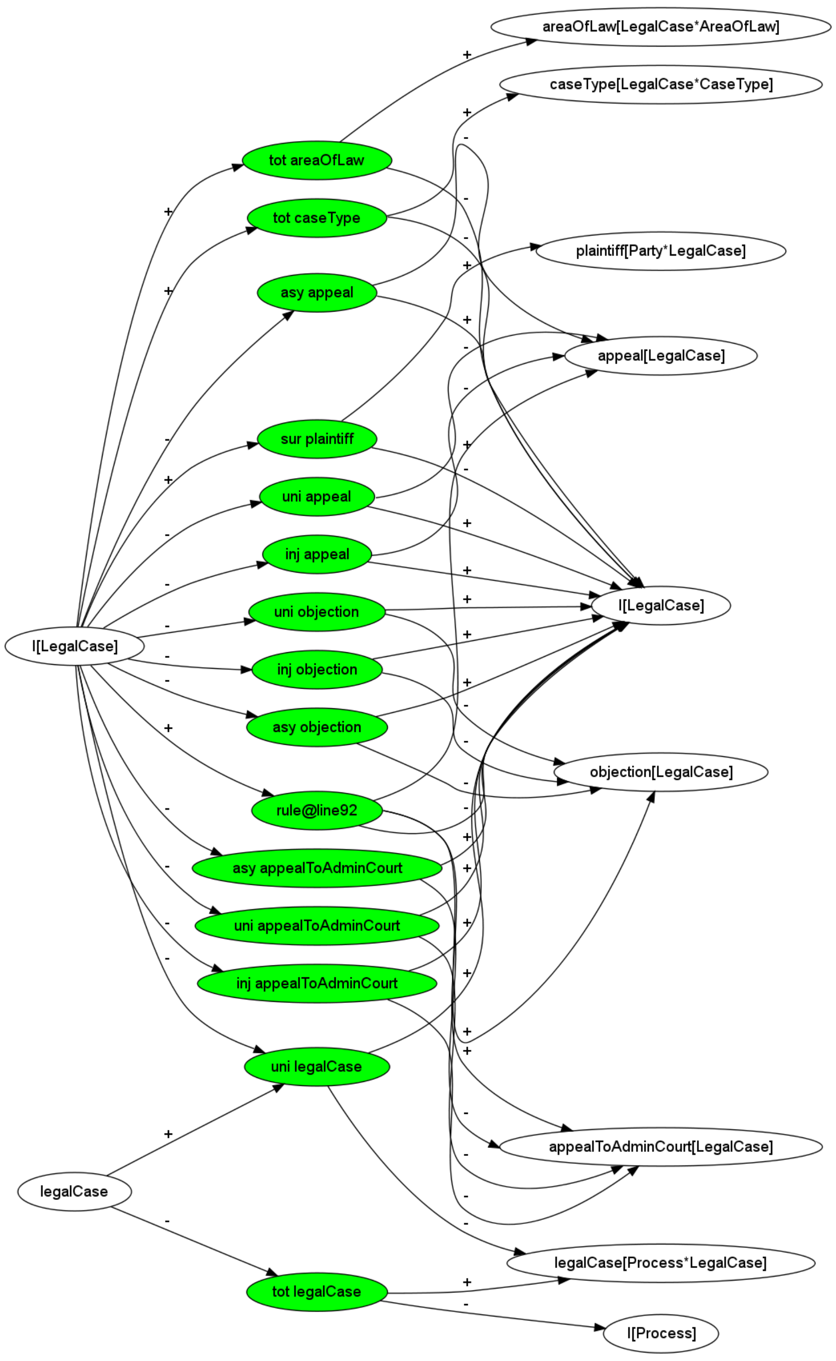


Figure 9.2: Switchboard of assign cases

Chapter 10

ECA rules

This chapter lists the ECA rules. ECA rules:
temporarily not documented

```
ON INSERT Delta IN I[Party] EXECUTE    -- (ECA rule 1)
BLOCK
(CANNOT CHANGE -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FROM R3
(CANNOT CHANGE -(defendant;defendant~) \/ -I[Party] \/ administrativeAuthority FROM R
(CANNOT CHANGE -person \/ -person \/ I[Party] FROM R0)
(CANNOT CHANGE -organization \/ -organization \/ I[Party] FROM R0)
(CANNOT CHANGE -administrativeAuthority \/ -administrativeAuthority \/ I[Party] FROM
(CANNOT CHANGE -memberOfGovernment \/ -memberOfGovernment \/ I[Party] FROM R0)
(CANNOT CHANGE -administrativeAuthorityAwb87 \/ -administrativeAuthorityAwb87 \/ I[Pa

ON DELETE Delta FROM I[Party] EXECUTE    -- (ECA rule 2)
BLOCK
(CANNOT CHANGE -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FROM R3
(CANNOT CHANGE -I[Party] \/ (person \/ organization \/ administrativeAuthority)/\(-pe
(CANNOT CHANGE -I[Party] \/ actsas;actsas~ FROM R0)

ON INSERT Delta IN plaintiff EXECUTE    -- (ECA rule 3)
ONE of INSERT INTO person SELECTFROM
         (plaintiff;plaintiff~ \/ plaintiff;Delta~ \/ Delta;plaintiff~ \/ Delta;Delta
       (TO MAINTAIN -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       INSERT INTO organization SELECTFROM
         (plaintiff;plaintiff~ \/ plaintiff;Delta~ \/ Delta;plaintiff~ \/ Delta;Delta
       (TO MAINTAIN -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       CREATE x:Party;
         ALL of INSERT INTO person SELECTFROM
                  'x'[Party];V[Party*LegalCase];((plaintiff~ \/ Delta~)/\-(plaintiff~
                INSERT INTO plaintiff~ SELECTFROM
                  -(plaintiff~;person)/\-(Delta~;person)/\-(plaintiff~;organization)/
         (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(plaintiff~);
         INSERT INTO person SELECTFROM
           'x'[Party];V[Party*LegalCase];((plaintiff~ \/ Delta~)/\-(plaintiff~;person
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(person~);
         INSERT INTO plaintiff~ SELECTFROM
           -(plaintiff~;person)/\-(Delta~;person)/\-(plaintiff~;organization)/\-(Delt
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(plaintiff~);
         INSERT INTO organization SELECTFROM
           'x'[Party];V[Party*LegalCase];((plaintiff~ \/ Delta~)/\-(plaintiff~;person
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(organization~);
         INSERT INTO plaintiff~ SELECTFROM
           -(plaintiff~;person)/\-(Delta~;person)/\-(plaintiff~;organization)/\-(Delt
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(person);
         INSERT INTO plaintiff SELECTFROM
           -(person;plaintiff)/\-(person;Delta)/\-(organization;plaintiff)/\-(organiz
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(plaintiff~);
         INSERT INTO person SELECTFROM
           ((plaintiff \/ Delta)/\-(person;plaintiff)/\-(person;Delta)/\-(organizatio
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(organization);
         INSERT INTO plaintiff SELECTFROM
           -(person;plaintiff)/\-(person;Delta)/\-(organization;plaintiff)/\-(organiz
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
       SELECT x:Party FROM codomain(plaintiff~);
         INSERT INTO organization SELECTFROM
           ((plaintiff \/ Delta)/\-(person;plaintiff)/\-(person;Delta)/\-(organizatio
       (MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FR
(MAINTAINING -(plaintiff;plaintiff~) \/ -I[Party] \/ person \/ organization FROM R3)

ON INSERT Delta IN defendant EXECUTE    -- (ECA rule 5)
ALL of ONE of INSERT INTO administrativeAuthority SELECTFROM
                (defendant;defendant~ \/ defendant;Delta~ \/ Delta;defendant~ \/ Delta
              (TO MAINTAIN -(defendant;defendant~) \/ -I[Party] \/ administrativeAutho
              INSERT INTO I[Party] SELECTFROM
                (administrativeAuthority;defendant;defendant~ \/ administrativeAuthor
              (TO MAINTAIN -(defendant;defendant~) \/ -I[Party] \/ administrativeAutho
              INSERT INTO I[Party] SELECTFROM
                (defendant;defendant~;administrativeAuthority \/ defendant;Delta~;adm
              (TO MAINTAIN -(defendant;defendant~) \/ -I[Party] \/ administrativeAutho
       (MAINTAINING -(defendant;defendant~) \/ -I[Party] \/ administrativeAuthority F
       ONE of INSERT INTO broughtBefore SELECTFROM
                (appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction
              (TO MAINTAIN -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
              CREATE x:Court;
                ALL of INSERT INTO jurisdiction~ SELECTFROM
```

```
                         V[Court*LegalCase];((appeal;defendant~;administrativeAuthori
                       INSERT INTO broughtBefore SELECTFROM
                         ((appeal;defendant~;administrativeAuthorityAwb87;domicile \/
                (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicil
              (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
              SELECT x:Court FROM codomain(broughtBefore);
                INSERT INTO jurisdiction~ SELECTFROM
                  V[Court*LegalCase];((appeal;defendant~;administrativeAuthorityAwb87
              (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
              SELECT x:Court FROM codomain(jurisdiction);
                INSERT INTO broughtBefore SELECTFROM
                  ((appeal;defendant~;administrativeAuthorityAwb87;domicile \/ appeal
              (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
       (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
(MAINTAINING -(defendant;defendant~) \/ -I[Party] \/ administrativeAuthority FROM R4)
(MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction) 

ON DELETE Delta FROM defendant EXECUTE    -- (ECA rule 6)
BLOCK
(CANNOT CHANGE -(defendant;defendant~) \/ -I[Party] \/ administrativeAuthority FROM R
(CANNOT CHANGE -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction

ON INSERT Delta IN authFor EXECUTE    -- (ECA rule 7)
BLOCK
(CANNOT CHANGE -authFor \/ -caseFile FROM R5)

ON DELETE Delta FROM authFor EXECUTE    -- (ECA rule 8)
BLOCK
(CANNOT CHANGE -authFor \/ -caseFile FROM R5)

ON INSERT Delta IN caseFile EXECUTE    -- (ECA rule 9)
BLOCK
(CANNOT CHANGE -authFor \/ -caseFile FROM R5)

ON DELETE Delta FROM caseFile EXECUTE    -- (ECA rule 10)
BLOCK
(CANNOT CHANGE -authFor \/ -caseFile FROM R5)

ON INSERT Delta IN I[LegalCase] EXECUTE    -- (ECA rule 11)
BLOCK
(CANNOT CHANGE -appeal \/ -appeal \/ I[LegalCase] FROM R0)
(CANNOT CHANGE -appealToAdminCourt \/ -appealToAdminCourt \/ I[LegalCase] FROM R0)
(CANNOT CHANGE -objection \/ -objection \/ I[LegalCase] FROM R0)

ON DELETE Delta FROM I[LegalCase] EXECUTE    -- (ECA rule 12)
BLOCK
(CANNOT CHANGE -I[LegalCase] \/ (appeal \/ appealToAdminCourt \/ objection)/\(-appeal
(CANNOT CHANGE -I[LegalCase] \/ plaintiff~;plaintiff FROM R0)
(CANNOT CHANGE -I[LegalCase] \/ areaOfLaw;areaOfLaw~ FROM R0)
(CANNOT CHANGE -I[LegalCase] \/ caseType;caseType~ FROM R0)
```

```
ON INSERT Delta IN objection EXECUTE    -- (ECA rule 13)
BLOCK
(CANNOT CHANGE -I[LegalCase] \/ (appeal \/ appealToAdminCourt \/ objection)/\(-appeal

ON DELETE Delta FROM objection EXECUTE    -- (ECA rule 14)
BLOCK
(CANNOT CHANGE -objection \/ -objection \/ I[LegalCase] FROM R0)
(CANNOT CHANGE -(objection;objection) \/ I[LegalCase] FROM R0)

ON INSERT Delta IN appealToAdminCourt EXECUTE    -- (ECA rule 15)
BLOCK
(CANNOT CHANGE -I[LegalCase] \/ (appeal \/ appealToAdminCourt \/ objection)/\(-appeal

ON DELETE Delta FROM appealToAdminCourt EXECUTE    -- (ECA rule 16)
BLOCK
(CANNOT CHANGE -appealToAdminCourt \/ -appealToAdminCourt \/ I[LegalCase] FROM R0)
(CANNOT CHANGE -(appealToAdminCourt;appealToAdminCourt) \/ I[LegalCase] FROM R0)

ON INSERT Delta IN appeal EXECUTE    -- (ECA rule 17)
BLOCK
(CANNOT CHANGE -I[LegalCase] \/ (appeal \/ appealToAdminCourt \/ objection)/\(-appeal

ON DELETE Delta FROM appeal EXECUTE    -- (ECA rule 18)
BLOCK
(CANNOT CHANGE -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction
(CANNOT CHANGE -appeal \/ -appeal \/ I[LegalCase] FROM R0)
(CANNOT CHANGE -(appeal;appeal) \/ I[LegalCase] FROM R0)

ON INSERT Delta IN administrativeAuthority EXECUTE    -- (ECA rule 19)
BLOCK
(CANNOT CHANGE -I[Party] \/ (person \/ organization \/ administrativeAuthority)/\(-pe

ON DELETE Delta FROM administrativeAuthority EXECUTE    -- (ECA rule 20)
BLOCK
(CANNOT CHANGE -administrativeAuthority \/ -administrativeAuthority \/ I[Party] FROM 
(CANNOT CHANGE -(administrativeAuthority;administrativeAuthority) \/ I[Party] FROM R0

ON INSERT Delta IN organization EXECUTE    -- (ECA rule 21)
BLOCK
(CANNOT CHANGE -I[Party] \/ (person \/ organization \/ administrativeAuthority)/\(-pe

ON DELETE Delta FROM organization EXECUTE    -- (ECA rule 22)
BLOCK
(CANNOT CHANGE -organization \/ -organization \/ I[Party] FROM R0)
(CANNOT CHANGE -(organization;organization) \/ I[Party] FROM R0)

ON INSERT Delta IN person EXECUTE    -- (ECA rule 23)
BLOCK
(CANNOT CHANGE -I[Party] \/ (person \/ organization \/ administrativeAuthority)/\(-pe
ON DELETE Delta FROM person EXECUTE    -- (ECA rule 24)
BLOCK
(CANNOT CHANGE -person \/ -person \/ I[Party] FROM R0)
(CANNOT CHANGE -(person;person) \/ I[Party] FROM R0)

ON INSERT Delta IN memberOfGovernment EXECUTE    -- (ECA rule 25)
ALL of INSERT INTO (memberOfGovernment \/ Delta)/\(memberOfGovernment \/ -I[Party])/\
         (memberOfGovernment \/ Delta)/\(memberOfGovernment \/ -I[Party])/\(-I[Party]
       (TO MAINTAIN -memberOfGovernment \/ -memberOfGovernment \/ I[Party] FROM R0)
       (TO MAINTAIN -(memberOfGovernment;memberOfGovernment) \/ I[Party] FROM R0)
       ONE of INSERT INTO administrativeAuthority SELECTFROM
                (memberOfGovernment \/ Delta)/\(memberOfGovernment \/ -administrative
              (TO MAINTAIN -memberOfGovernment \/ administrativeAuthority FROM R8)
              INSERT INTO I[Party] SELECTFROM
                (administrativeAuthority;memberOfGovernment \/ administrativeAuthorit
              (TO MAINTAIN -memberOfGovernment \/ administrativeAuthority FROM R8)
              INSERT INTO I[Party] SELECTFROM
                (memberOfGovernment;administrativeAuthority \/ Delta;administrativeAu
              (TO MAINTAIN -memberOfGovernment \/ administrativeAuthority FROM R8)
       (MAINTAINING -memberOfGovernment \/ administrativeAuthority FROM R8)
(MAINTAINING -memberOfGovernment \/ administrativeAuthority FROM R8)
(MAINTAINING -memberOfGovernment \/ -memberOfGovernment \/ I[Party] FROM R0)
(MAINTAINING -(memberOfGovernment;memberOfGovernment) \/ I[Party] FROM R0)

ON DELETE Delta FROM memberOfGovernment EXECUTE    -- (ECA rule 26)
BLOCK
(CANNOT CHANGE -memberOfGovernment \/ administrativeAuthority FROM R8)
(CANNOT CHANGE -memberOfGovernment \/ -memberOfGovernment \/ I[Party] FROM R0)
(CANNOT CHANGE -(memberOfGovernment;memberOfGovernment) \/ I[Party] FROM R0)

ON INSERT Delta IN scheduled EXECUTE    -- (ECA rule 27)
ALL of INSERT INTO I[Session] SELECTFROM
         panel;panel~/\location;location~/\(scheduled;scheduled~ \/ scheduled;Delta~ \
       (TO MAINTAIN -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled
       INSERT INTO I[Date] SELECTFROM
         (scheduled~;scheduled \/ scheduled~;Delta \/ Delta~;scheduled \/ Delta~;Delt
       (TO MAINTAIN -(scheduled~;scheduled) \/ I[Date] FROM R0)
(MAINTAINING -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled~) \/ I
(MAINTAINING -(scheduled~;scheduled) \/ I[Date] FROM R0)

ON DELETE Delta FROM scheduled EXECUTE    -- (ECA rule 28)
BLOCK
(CANNOT CHANGE -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled~) \/
(CANNOT CHANGE -(scheduled~;scheduled) \/ I[Date] FROM R0)

ON INSERT Delta IN location EXECUTE    -- (ECA rule 29)
ALL of INSERT INTO I[Session] SELECTFROM
         panel;panel~/\(location;location~ \/ location;Delta~ \/ Delta;location~ \/ D
       (TO MAINTAIN -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled
       INSERT INTO I[Court] SELECTFROM
         (location~;location \/ location~;Delta \/ Delta~;location \/ Delta~;Delta)/\
       (TO MAINTAIN -(location~;location) \/ I[Court] FROM R0)
(MAINTAINING -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled~) \/ I
(MAINTAINING -(location~;location) \/ I[Court] FROM R0)

ON DELETE Delta FROM location EXECUTE    -- (ECA rule 30)
BLOCK
(CANNOT CHANGE -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled~) \/
(CANNOT CHANGE -(location~;location) \/ I[Court] FROM R0)

ON INSERT Delta IN panel EXECUTE    -- (ECA rule 31)
ALL of INSERT INTO I[Session] SELECTFROM
         (panel;panel~ \/ panel;Delta~ \/ Delta;panel~ \/ Delta;Delta~)/\location;loc
       (TO MAINTAIN -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled
       INSERT INTO I[Panel] SELECTFROM
         (panel~;panel \/ panel~;Delta \/ Delta~;panel \/ Delta~;Delta)/\(panel~;pane
       (TO MAINTAIN -(panel~;panel) \/ I[Panel] FROM R0)
(MAINTAINING -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled~) \/ I
(MAINTAINING -(panel~;panel) \/ I[Panel] FROM R0)

ON DELETE Delta FROM panel EXECUTE    -- (ECA rule 32)
BLOCK
(CANNOT CHANGE -(panel;panel~) \/ -(location;location~) \/ -(scheduled;scheduled~) \/
(CANNOT CHANGE -(panel~;panel) \/ I[Panel] FROM R0)

ON INSERT Delta IN judge EXECUTE    -- (ECA rule 33)
ONE of CREATE x:Panel;
         ALL of INSERT INTO members~ SELECTFROM
                  V[Panel*Session];((judge \/ Delta)/\-(panel;members~)) \/ -members~
                INSERT INTO panel SELECTFROM
                  ((judge \/ Delta)/\-(panel;members~));V[Party*Panel] \/ -panel
         (MAINTAINING -judge \/ panel;members~ FROM R10)
       (MAINTAINING -judge \/ panel;members~ FROM R10)
       SELECT x:Panel FROM codomain(panel);
         INSERT INTO members~ SELECTFROM
           V[Panel*Session];((judge \/ Delta)/\-(panel;members~)) \/ -members~
       (MAINTAINING -judge \/ panel;members~ FROM R10)
       SELECT x:Panel FROM codomain(members);
         INSERT INTO panel SELECTFROM
           ((judge \/ Delta)/\-(panel;members~));V[Party*Panel] \/ -panel
       (MAINTAINING -judge \/ panel;members~ FROM R10)
       INSERT INTO members~ SELECTFROM
         (panel~;judge \/ panel~;Delta)/\(panel~;judge \/ -members~)/\(-members~ \/ p
       (TO MAINTAIN -judge \/ panel;members~ FROM R10)
(MAINTAINING -judge \/ panel;members~ FROM R10)

ON INSERT Delta IN clerk[Session*Party] EXECUTE    -- (ECA rule 35)
```

```
ALL of ONE of CREATE x:Court;
                ALL of INSERT INTO clerk[Court*Party] SELECTFROM
                         V[Court*Session];((clerk[Session*Party] \/ Delta)/\-(locatio
                       INSERT INTO location SELECTFROM
                         ((clerk[Session*Party] \/ Delta)/\-(location;clerk[Court*Par
                (MAINTAINING -clerk[Session*Party] \/ location;clerk[Court*Party] FRO
              (MAINTAINING -clerk[Session*Party] \/ location;clerk[Court*Party] FROM
              SELECT x:Court FROM codomain(location);
                INSERT INTO clerk[Court*Party] SELECTFROM
                  V[Court*Session];((clerk[Session*Party] \/ Delta)/\-(location;clerk
              (MAINTAINING -clerk[Session*Party] \/ location;clerk[Court*Party] FROM
              SELECT x:Court FROM codomain(clerk[Court*Party]~);
                INSERT INTO location SELECTFROM
                  ((clerk[Session*Party] \/ Delta)/\-(location;clerk[Court*Party]));V
              (MAINTAINING -clerk[Session*Party] \/ location;clerk[Court*Party] FROM
              INSERT INTO clerk[Court*Party] SELECTFROM
                (location~;clerk[Session*Party] \/ location~;Delta)/\(location~;clerk
              (TO MAINTAIN -clerk[Session*Party] \/ location;clerk[Court*Party] FROM
       (MAINTAINING -clerk[Session*Party] \/ location;clerk[Court*Party] FROM R11)
       INSERT INTO I[Party] SELECTFROM
         (clerk[Session*Party]~;I[Session];clerk[Session*Party] \/ clerk[Session*Part
       (TO MAINTAIN -(clerk[Session*Party]~;I[Session];clerk[Session*Party]) \/ I[Par
(MAINTAINING -clerk[Session*Party] \/ location;clerk[Court*Party] FROM R11)
(MAINTAINING -(clerk[Session*Party]~;I[Session];clerk[Session*Party]) \/ I[Party] FRO

ON DELETE Delta FROM clerk[Session*Party] EXECUTE    -- (ECA rule 36)
BLOCK
(CANNOT CHANGE -(clerk[Session*Party]~;I[Session];clerk[Session*Party]) \/ I[Party] F

ON INSERT Delta IN occured EXECUTE    -- (ECA rule 37)
ALL of ONE of INSERT INTO scheduled SELECTFROM
                (occured \/ Delta)/\(occured \/ -scheduled)/\(-scheduled \/ Delta)/\(
              (TO MAINTAIN -occured \/ scheduled FROM R12)
              INSERT INTO I[Date] SELECTFROM
                (scheduled~;occured \/ scheduled~;Delta)/\(scheduled~;occured \/ -I[D
              (TO MAINTAIN -occured \/ scheduled FROM R12)
       (MAINTAINING -occured \/ scheduled FROM R12)
       INSERT INTO I[Date] SELECTFROM
         (occured~;occured \/ occured~;Delta \/ Delta~;occured \/ Delta~;Delta)/\(occ
       (TO MAINTAIN -(occured~;occured) \/ I[Date] FROM R0)
(MAINTAINING -occured \/ scheduled FROM R12)
(MAINTAINING -(occured~;occured) \/ I[Date] FROM R0)

ON DELETE Delta FROM occured EXECUTE    -- (ECA rule 38)
BLOCK
(CANNOT CHANGE -occured \/ scheduled FROM R12)
(CANNOT CHANGE -(occured~;occured) \/ I[Date] FROM R0)

ON INSERT Delta IN administrativeAuthorityAwb87 EXECUTE    -- (ECA rule 39)
```

```
ALL of INSERT INTO (administrativeAuthorityAwb87 \/ Delta)/\(administrativeAuthorityAw
         (administrativeAuthorityAwb87 \/ Delta)/\(administrativeAuthorityAwb87 \/ -I
       (TO MAINTAIN -administrativeAuthorityAwb87 \/ -administrativeAuthorityAwb87 \/
       (TO MAINTAIN -(administrativeAuthorityAwb87;administrativeAuthorityAwb87) \/ I
       ONE of INSERT INTO administrativeAuthority SELECTFROM
                (administrativeAuthorityAwb87 \/ Delta)/\(administrativeAuthorityAwb87
              (TO MAINTAIN -administrativeAuthorityAwb87 \/ administrativeAuthority F
              INSERT INTO I[Party] SELECTFROM
                (administrativeAuthority;administrativeAuthorityAwb87 \/ administrati
              (TO MAINTAIN -administrativeAuthorityAwb87 \/ administrativeAuthority F
              INSERT INTO I[Party] SELECTFROM
                (administrativeAuthorityAwb87;administrativeAuthority \/ Delta;admini
              (TO MAINTAIN -administrativeAuthorityAwb87 \/ administrativeAuthority F
       (MAINTAINING -administrativeAuthorityAwb87 \/ administrativeAuthority FROM R13
       ONE of INSERT INTO broughtBefore SELECTFROM
                (appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction
              (TO MAINTAIN -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
              CREATE x:Court;
                ALL of INSERT INTO jurisdiction~ SELECTFROM
                         V[Court*LegalCase];((appeal;defendant~;administrativeAuthori
                       INSERT INTO broughtBefore SELECTFROM
                         ((appeal;defendant~;administrativeAuthorityAwb87;domicile \/
                (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicil
              (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
              SELECT x:Court FROM codomain(broughtBefore);
                INSERT INTO jurisdiction~ SELECTFROM
                  V[Court*LegalCase];((appeal;defendant~;administrativeAuthorityAwb87
              (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
              SELECT x:Court FROM codomain(jurisdiction);
                INSERT INTO broughtBefore SELECTFROM
                  ((appeal;defendant~;administrativeAuthorityAwb87;domicile \/ appeal
              (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;
       (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
(MAINTAINING -administrativeAuthorityAwb87 \/ administrativeAuthority FROM R13)
(MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction)
(MAINTAINING -administrativeAuthorityAwb87 \/ -administrativeAuthorityAwb87 \/ I[Part
(MAINTAINING -(administrativeAuthorityAwb87;administrativeAuthorityAwb87) \/ I[Party]

ON DELETE Delta FROM administrativeAuthorityAwb87 EXECUTE    -- (ECA rule 40)
BLOCK
(CANNOT CHANGE -administrativeAuthorityAwb87 \/ administrativeAuthority FROM R13)
(CANNOT CHANGE -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction
(CANNOT CHANGE -administrativeAuthorityAwb87 \/ -administrativeAuthorityAwb87 \/ I[Pa
(CANNOT CHANGE -(administrativeAuthorityAwb87;administrativeAuthorityAwb87) \/ I[Part

ON INSERT Delta IN jurisdiction EXECUTE    -- (ECA rule 41)
ALL of INSERT INTO broughtBefore SELECTFROM
         (appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction \/ app
       (TO MAINTAIN -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
       INSERT INTO I[Court] SELECTFROM
         (jurisdiction~;jurisdiction \/ jurisdiction~;Delta \/ Delta~;jurisdiction \/
       (TO MAINTAIN -(jurisdiction~;jurisdiction) \/ I[Court] FROM R0)
(MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction)
(MAINTAINING -(jurisdiction~;jurisdiction) \/ I[Court] FROM R0)

ON DELETE Delta FROM jurisdiction EXECUTE    -- (ECA rule 42)
BLOCK
(CANNOT CHANGE -(jurisdiction~;jurisdiction) \/ I[Court] FROM R0)

ON INSERT Delta IN domicile EXECUTE    -- (ECA rule 43)
ONE of INSERT INTO broughtBefore SELECTFROM
         (appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction \/ app
       (TO MAINTAIN -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
       CREATE x:Court;
         ALL of INSERT INTO jurisdiction~ SELECTFROM
                  V[Court*LegalCase];((appeal;defendant~;administrativeAuthorityAwb87
                INSERT INTO broughtBefore SELECTFROM
                  ((appeal;defendant~;administrativeAuthorityAwb87;domicile \/ appeal
         (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;juris
       (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
       SELECT x:Court FROM codomain(broughtBefore);
         INSERT INTO jurisdiction~ SELECTFROM
           V[Court*LegalCase];((appeal;defendant~;administrativeAuthorityAwb87;domici
       (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
       SELECT x:Court FROM codomain(jurisdiction);
         INSERT INTO broughtBefore SELECTFROM
           ((appeal;defendant~;administrativeAuthorityAwb87;domicile \/ appeal;defend
       (MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdi
(MAINTAINING -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction)

ON DELETE Delta FROM domicile EXECUTE    -- (ECA rule 44)
BLOCK
(CANNOT CHANGE -(appeal;defendant~;administrativeAuthorityAwb87;domicile;jurisdiction

ON INSERT Delta IN areaOfLaw[LegalCase*AreaOfLaw] EXECUTE    -- (ECA rule 45)
INSERT INTO I[AreaOfLaw] SELECTFROM
  (areaOfLaw~;areaOfLaw \/ areaOfLaw~;Delta \/ Delta~;areaOfLaw \/ Delta~;Delta)/\(ar
(TO MAINTAIN -(areaOfLaw~;areaOfLaw) \/ I[AreaOfLaw] FROM R0)

ON DELETE Delta FROM areaOfLaw[LegalCase*AreaOfLaw] EXECUTE    -- (ECA rule 46)
BLOCK
(CANNOT CHANGE -(areaOfLaw~;areaOfLaw) \/ I[AreaOfLaw] FROM R0)

ON INSERT Delta IN documentType[Document*DocumentType] EXECUTE    -- (ECA rule 47)
INSERT INTO I[DocumentType] SELECTFROM
  (documentType~;documentType \/ documentType~;Delta \/ Delta~;documentType \/ Delta~
(TO MAINTAIN -(documentType~;documentType) \/ I[DocumentType] FROM R0)
```

```
ON DELETE Delta FROM documentType[Document*DocumentType] EXECUTE    -- (ECA rule 48)
BLOCK
(CANNOT CHANGE -(documentType~;documentType) \/ I[DocumentType] FROM R0)

ON INSERT Delta IN I[Document] EXECUTE    -- (ECA rule 49)
ONE of CREATE x:DocumentType;
         INSERT INTO documentType~ SELECTFROM
           V[DocumentType*Document];((I[Document] \/ Delta)/\-(documentType;documentT
       (MAINTAINING -I[Document] \/ documentType;documentType~ FROM R0)
       SELECT x:DocumentType FROM codomain(documentType);
         INSERT INTO documentType~ SELECTFROM
           V[DocumentType*Document];((I[Document] \/ Delta)/\-(documentType;documentT
       (MAINTAINING -I[Document] \/ documentType;documentType~ FROM R0)
(MAINTAINING -I[Document] \/ documentType;documentType~ FROM R0)

ON DELETE Delta FROM I[Document] EXECUTE    -- (ECA rule 50)
BLOCK
(CANNOT CHANGE -I[Document] \/ documentType;documentType~ FROM R0)

ON INSERT Delta IN caseType[LegalCase*CaseType] EXECUTE    -- (ECA rule 51)
INSERT INTO I[CaseType] SELECTFROM
  (caseType~;caseType \/ caseType~;Delta \/ Delta~;caseType \/ Delta~;Delta)/\(caseTy
(TO MAINTAIN -(caseType~;caseType) \/ I[CaseType] FROM R0)

ON DELETE Delta FROM caseType[LegalCase*CaseType] EXECUTE    -- (ECA rule 52)
BLOCK
(CANNOT CHANGE -(caseType~;caseType) \/ I[CaseType] FROM R0)

ON INSERT Delta IN session[Process*Session] EXECUTE    -- (ECA rule 53)
INSERT INTO I[Session] SELECTFROM
  (session~;session \/ session~;Delta \/ Delta~;session \/ Delta~;Delta)/\(session~;s
(TO MAINTAIN -(session~;session) \/ I[Session] FROM R0)

ON DELETE Delta FROM session[Process*Session] EXECUTE    -- (ECA rule 54)
BLOCK
(CANNOT CHANGE -(session~;session) \/ I[Session] FROM R0)

ON INSERT Delta IN I[Process] EXECUTE    -- (ECA rule 55)
ALL of ONE of CREATE x:Session;
                INSERT INTO session~ SELECTFROM
                  V[Session*Process];((I[Process] \/ Delta)/\-(session;session~)) \/
              (MAINTAINING -I[Process] \/ session;session~ FROM R0)
              SELECT x:Session FROM codomain(session);
                INSERT INTO session~ SELECTFROM
                  V[Session*Process];((I[Process] \/ Delta)/\-(session;session~)) \/
              (MAINTAINING -I[Process] \/ session;session~ FROM R0)
       (MAINTAINING -I[Process] \/ session;session~ FROM R0)
       ONE of CREATE x:LegalCase;
```

```
                INSERT INTO legalCase~ SELECTFROM
                  V[LegalCase*Process];((I[Process] \/ Delta)/\-(legalCase;legalCase~
              (MAINTAINING -I[Process] \/ legalCase;legalCase~ FROM R0)
              SELECT x:LegalCase FROM codomain(legalCase);
                INSERT INTO legalCase~ SELECTFROM
                  V[LegalCase*Process];((I[Process] \/ Delta)/\-(legalCase;legalCase~
              (MAINTAINING -I[Process] \/ legalCase;legalCase~ FROM R0)
       (MAINTAINING -I[Process] \/ legalCase;legalCase~ FROM R0)
(MAINTAINING -I[Process] \/ session;session~ FROM R0)
(MAINTAINING -I[Process] \/ legalCase;legalCase~ FROM R0)

ON DELETE Delta FROM I[Process] EXECUTE    -- (ECA rule 56)
BLOCK
(CANNOT CHANGE -I[Process] \/ session;session~ FROM R0)
(CANNOT CHANGE -I[Process] \/ legalCase;legalCase~ FROM R0)

ON INSERT Delta IN legalCase[Process*LegalCase] EXECUTE    -- (ECA rule 57)
INSERT INTO I[LegalCase] SELECTFROM
  (legalCase~;legalCase \/ legalCase~;Delta \/ Delta~;legalCase \/ Delta~;Delta)/\(le
(TO MAINTAIN -(legalCase~;legalCase) \/ I[LegalCase] FROM R0)

ON DELETE Delta FROM legalCase[Process*LegalCase] EXECUTE    -- (ECA rule 58)
BLOCK
(CANNOT CHANGE -(legalCase~;legalCase) \/ I[LegalCase] FROM R0)

ON INSERT Delta IN I[Session] EXECUTE    -- (ECA rule 59)
ALL of ONE of CREATE x:Panel;
                INSERT INTO panel~ SELECTFROM
                  V[Panel*Session];((I[Session] \/ Delta)/\-(panel;panel~)) \/ -panel
              (MAINTAINING -I[Session] \/ panel;panel~ FROM R0)
              SELECT x:Panel FROM codomain(panel);
                INSERT INTO panel~ SELECTFROM
                  V[Panel*Session];((I[Session] \/ Delta)/\-(panel;panel~)) \/ -panel
              (MAINTAINING -I[Session] \/ panel;panel~ FROM R0)
       (MAINTAINING -I[Session] \/ panel;panel~ FROM R0)
       ONE of CREATE x:Party;
                INSERT INTO judge~ SELECTFROM
                  V[Party*Session];((I[Session] \/ Delta)/\-(judge;judge~)) \/ -judge
              (MAINTAINING -I[Session] \/ judge;judge~ FROM R0)
              SELECT x:Party FROM codomain(judge);
                INSERT INTO judge~ SELECTFROM
                  V[Party*Session];((I[Session] \/ Delta)/\-(judge;judge~)) \/ -judge
              (MAINTAINING -I[Session] \/ judge;judge~ FROM R0)
       (MAINTAINING -I[Session] \/ judge;judge~ FROM R0)
       ONE of CREATE x:Party;
                INSERT INTO clerk[Session*Party]~ SELECTFROM
                  V[Party*Session];((I[Session] \/ Delta)/\-(clerk[Session*Party];cle
              (MAINTAINING -I[Session] \/ clerk[Session*Party];clerk[Session*Party]~ 
              SELECT x:Party FROM codomain(clerk[Session*Party]);
```

```
                INSERT INTO clerk[Session*Party]~ SELECTFROM
                  V[Party*Session];((I[Session] \/ Delta)/\-(clerk[Session*Party];cle
              (MAINTAINING -I[Session] \/ clerk[Session*Party];clerk[Session*Party]~ 
       (MAINTAINING -I[Session] \/ clerk[Session*Party];clerk[Session*Party]~ FROM R0
       ONE of CREATE x:Date;
                INSERT INTO scheduled~ SELECTFROM
                  V[Date*Session];((I[Session] \/ Delta)/\-(scheduled;scheduled~)) \/
              (MAINTAINING -I[Session] \/ scheduled;scheduled~ FROM R0)
              SELECT x:Date FROM codomain(scheduled);
                INSERT INTO scheduled~ SELECTFROM
                  V[Date*Session];((I[Session] \/ Delta)/\-(scheduled;scheduled~)) \/
              (MAINTAINING -I[Session] \/ scheduled;scheduled~ FROM R0)
       (MAINTAINING -I[Session] \/ scheduled;scheduled~ FROM R0)
       ONE of CREATE x:Court;
                INSERT INTO location~ SELECTFROM
                  V[Court*Session];((I[Session] \/ Delta)/\-(location;location~)) \/
              (MAINTAINING -I[Session] \/ location;location~ FROM R0)
              SELECT x:Court FROM codomain(location);
                INSERT INTO location~ SELECTFROM
                  V[Court*Session];((I[Session] \/ Delta)/\-(location;location~)) \/
              (MAINTAINING -I[Session] \/ location;location~ FROM R0)
       (MAINTAINING -I[Session] \/ location;location~ FROM R0)
(MAINTAINING -I[Session] \/ panel;panel~ FROM R0)
(MAINTAINING -I[Session] \/ judge;judge~ FROM R0)
(MAINTAINING -I[Session] \/ clerk[Session*Party];clerk[Session*Party]~ FROM R0)
(MAINTAINING -I[Session] \/ scheduled;scheduled~ FROM R0)
(MAINTAINING -I[Session] \/ location;location~ FROM R0)

ON DELETE Delta FROM I[Session] EXECUTE    -- (ECA rule 60)
BLOCK
(CANNOT CHANGE -I[Session] \/ panel;panel~ FROM R0)
(CANNOT CHANGE -I[Session] \/ judge;judge~ FROM R0)
(CANNOT CHANGE -I[Session] \/ clerk[Session*Party];clerk[Session*Party]~ FROM R0)
(CANNOT CHANGE -I[Session] \/ scheduled;scheduled~ FROM R0)
(CANNOT CHANGE -I[Session] \/ location;location~ FROM R0)

ON INSERT Delta IN court[Panel*Court] EXECUTE    -- (ECA rule 61)
INSERT INTO I[Court] SELECTFROM
  (court~;court \/ court~;Delta \/ Delta~;court \/ Delta~;Delta)/\(court~;court \/ co
(TO MAINTAIN -(court~;court) \/ I[Court] FROM R0)

ON DELETE Delta FROM court[Panel*Court] EXECUTE    -- (ECA rule 62)
BLOCK
(CANNOT CHANGE -(court~;court) \/ I[Court] FROM R0)

ON INSERT Delta IN I[Panel] EXECUTE    -- (ECA rule 63)
ONE of CREATE x:Court;
         INSERT INTO court~ SELECTFROM
           V[Court*Panel];((I[Panel] \/ Delta)/\-(court;court~)) \/ -court~
       (MAINTAINING -I[Panel] \/ court;court~ FROM R0)
       SELECT x:Court FROM codomain(court);
         INSERT INTO court~ SELECTFROM
           V[Court*Panel];((I[Panel] \/ Delta)/\-(court;court~)) \/ -court~
       (MAINTAINING -I[Panel] \/ court;court~ FROM R0)
(MAINTAINING -I[Panel] \/ court;court~ FROM R0)

ON DELETE Delta FROM I[Panel] EXECUTE    -- (ECA rule 64)
BLOCK
(CANNOT CHANGE -I[Panel] \/ court;court~ FROM R0)

ON INSERT Delta IN actsas[Party*Role] EXECUTE    -- (ECA rule 65)
INSERT INTO I[Role] SELECTFROM
  (actsas~;actsas \/ actsas~;Delta \/ Delta~;actsas \/ Delta~;Delta)/\(actsas~;actsas
(TO MAINTAIN -(actsas~;actsas) \/ I[Role] FROM R0)

ON DELETE Delta FROM actsas[Party*Role] EXECUTE    -- (ECA rule 66)
BLOCK
(CANNOT CHANGE -(actsas~;actsas) \/ I[Role] FROM R0)

ON INSERT Delta IN seatedIn[Court*City] EXECUTE    -- (ECA rule 67)
INSERT INTO I[City] SELECTFROM
  (seatedIn[Court*City]~;I[Court];seatedIn[Court*City] \/ seatedIn[Court*City]~;Delta
(TO MAINTAIN -(seatedIn[Court*City]~;I[Court];seatedIn[Court*City]) \/ I[City] FROM R

ON DELETE Delta FROM seatedIn[Court*City] EXECUTE    -- (ECA rule 68)
BLOCK
(CANNOT CHANGE -(seatedIn[Court*City]~;I[Court];seatedIn[Court*City]) \/ I[City] FROM

ON INSERT Delta IN I[Court] EXECUTE    -- (ECA rule 69)
ALL of ONE of CREATE x:City;
                INSERT INTO seatedIn[Court*City]~ SELECTFROM
                  V[City*Court];((I[Court] \/ Delta)/\-(seatedIn[Court*City];seatedIn
              (MAINTAINING -I[Court] \/ seatedIn[Court*City];seatedIn[Court*City]~ FR
              SELECT x:City FROM codomain(seatedIn[Court*City]);
                INSERT INTO seatedIn[Court*City]~ SELECTFROM
                  V[City*Court];((I[Court] \/ Delta)/\-(seatedIn[Court*City];seatedIn
              (MAINTAINING -I[Court] \/ seatedIn[Court*City];seatedIn[Court*City]~ FR
       (MAINTAINING -I[Court] \/ seatedIn[Court*City];seatedIn[Court*City]~ FROM R0)
       ONE of CREATE x:CourtOfAppeal;
                INSERT INTO district~ SELECTFROM
                  V[CourtOfAppeal*Court];((I[Court] \/ Delta)/\-(district;district~))
              (MAINTAINING -I[Court] \/ district;district~ FROM R0)
              SELECT x:CourtOfAppeal FROM codomain(district);
                INSERT INTO district~ SELECTFROM
                  V[CourtOfAppeal*Court];((I[Court] \/ Delta)/\-(district;district~))
              (MAINTAINING -I[Court] \/ district;district~ FROM R0)
       (MAINTAINING -I[Court] \/ district;district~ FROM R0)
(MAINTAINING -I[Court] \/ seatedIn[Court*City];seatedIn[Court*City]~ FROM R0)
(MAINTAINING -I[Court] \/ district;district~ FROM R0)
```

```
ON DELETE Delta FROM I[Court] EXECUTE    -- (ECA rule 70)
BLOCK
(CANNOT CHANGE -I[Court] \/ seatedIn[Court*City];seatedIn[Court*City]~ FROM R0)
(CANNOT CHANGE -I[Court] \/ district;district~ FROM R0)

ON INSERT Delta IN seatedIn[CourtOfAppeal*City] EXECUTE    -- (ECA rule 71)
INSERT INTO I[City] SELECTFROM
  (seatedIn[CourtOfAppeal*City]~;I[CourtOfAppeal];seatedIn[CourtOfAppeal*City] \/ sea
(TO MAINTAIN -(seatedIn[CourtOfAppeal*City]~;I[CourtOfAppeal];seatedIn[CourtOfAppeal*

ON DELETE Delta FROM seatedIn[CourtOfAppeal*City] EXECUTE    -- (ECA rule 72)
BLOCK
(CANNOT CHANGE -(seatedIn[CourtOfAppeal*City]~;I[CourtOfAppeal];seatedIn[CourtOfAppea

ON INSERT Delta IN I[CourtOfAppeal] EXECUTE    -- (ECA rule 73)
ONE of CREATE x:City;
         INSERT INTO seatedIn[CourtOfAppeal*City]~ SELECTFROM
           V[City*CourtOfAppeal];((I[CourtOfAppeal] \/ Delta)/\-(seatedIn[CourtOfAppe
       (MAINTAINING -I[CourtOfAppeal] \/ seatedIn[CourtOfAppeal*City];seatedIn[CourtO
       SELECT x:City FROM codomain(seatedIn[CourtOfAppeal*City]);
         INSERT INTO seatedIn[CourtOfAppeal*City]~ SELECTFROM
           V[City*CourtOfAppeal];((I[CourtOfAppeal] \/ Delta)/\-(seatedIn[CourtOfAppe
       (MAINTAINING -I[CourtOfAppeal] \/ seatedIn[CourtOfAppeal*City];seatedIn[CourtO
(MAINTAINING -I[CourtOfAppeal] \/ seatedIn[CourtOfAppeal*City];seatedIn[CourtOfAppeal

ON DELETE Delta FROM I[CourtOfAppeal] EXECUTE    -- (ECA rule 74)
BLOCK
(CANNOT CHANGE -I[CourtOfAppeal] \/ seatedIn[CourtOfAppeal*City];seatedIn[CourtOfAppe

ON INSERT Delta IN district[Court*CourtOfAppeal] EXECUTE    -- (ECA rule 75)
INSERT INTO I[CourtOfAppeal] SELECTFROM
  (district~;district \/ district~;Delta \/ Delta~;district \/ Delta~;Delta)/\(distri
(TO MAINTAIN -(district~;district) \/ I[CourtOfAppeal] FROM R0)

ON DELETE Delta FROM district[Court*CourtOfAppeal] EXECUTE    -- (ECA rule 76)
BLOCK
(CANNOT CHANGE -(district~;district) \/ I[CourtOfAppeal] FROM R0)

ON INSERT Delta IN localOffices[City*Court] EXECUTE    -- (ECA rule 77)
INSERT INTO I[Court] SELECTFROM
  (localOffices~;localOffices \/ localOffices~;Delta \/ Delta~;localOffices \/ Delta~
(TO MAINTAIN -(localOffices~;localOffices) \/ I[Court] FROM R0)

ON DELETE Delta FROM localOffices[City*Court] EXECUTE    -- (ECA rule 78)
BLOCK
(CANNOT CHANGE -(localOffices~;localOffices) \/ I[Court] FROM R0)
```

```
ON INSERT Delta IN I[City] EXECUTE    -- (ECA rule 79)
ONE of CREATE x:Court;
         INSERT INTO jurisdiction~ SELECTFROM
           V[Court*City];((I[City] \/ Delta)/\-(jurisdiction;jurisdiction~)) \/ -juri
       (MAINTAINING -I[City] \/ jurisdiction;jurisdiction~ FROM R0)
       SELECT x:Court FROM codomain(jurisdiction);
         INSERT INTO jurisdiction~ SELECTFROM
           V[Court*City];((I[City] \/ Delta)/\-(jurisdiction;jurisdiction~)) \/ -juri
       (MAINTAINING -I[City] \/ jurisdiction;jurisdiction~ FROM R0)
(MAINTAINING -I[City] \/ jurisdiction;jurisdiction~ FROM R0)

ON DELETE Delta FROM I[City] EXECUTE    -- (ECA rule 80)
BLOCK
(CANNOT CHANGE -I[City] \/ jurisdiction;jurisdiction~ FROM R0)

ON INSERT Delta IN sent[Document*TimeStamp] EXECUTE    -- (ECA rule 81)
INSERT INTO I[TimeStamp] SELECTFROM
  (sent~;sent \/ sent~;Delta \/ Delta~;sent \/ Delta~;Delta)/\(sent~;sent \/ sent~;De
(TO MAINTAIN -(sent~;sent) \/ I[TimeStamp] FROM R0)

ON DELETE Delta FROM sent[Document*TimeStamp] EXECUTE    -- (ECA rule 82)
BLOCK
(CANNOT CHANGE -(sent~;sent) \/ I[TimeStamp] FROM R0)

ON INSERT Delta IN received[Document*TimeStamp] EXECUTE    -- (ECA rule 83)
INSERT INTO I[TimeStamp] SELECTFROM
  (received~;received \/ received~;Delta \/ Delta~;received \/ Delta~;Delta)/\(receiv
(TO MAINTAIN -(received~;received) \/ I[TimeStamp] FROM R0)

ON DELETE Delta FROM received[Document*TimeStamp] EXECUTE    -- (ECA rule 84)
BLOCK
(CANNOT CHANGE -(received~;received) \/ I[TimeStamp] FROM R0)
```

term	definition	source