Title	Safeguarding Trusted eBusiness Transactions of Lifecycles for Cross-Enterprize Collaboration
Author(s)	Norta, Alex
Citation	Norta , A 2012 , Safeguarding Trusted eBusiness Transactions of Lifecycles for Cross-Enterprize Collaboration , Department of Computer Science, Series of Publications C , no. C-2012-1 , University of Helsinki, Department of Computer Science , Helsinki .
Date	2012
URL	http://hdl.handle.net/10138/30150

Safeguarding Trusted eBusiness Transactions of Lifecycles for Cross-Enterprise Collaboration

TECHNICAL REPORT: C-2012-1

Trusted Business Transactions (TBT): 129117, Academy of Finland
Project leader: Prof. Lea Kutvonen

authored by

Alex Norta, PhD.

CINCO, Collaborative and Interoperable Computing,
Department of Computer Science, University of Helsinki,
Gustaf Hällströmin katu 2b, FI-00014, Helsinki

List of Abbreviations

ATC	Abstract Transaction Construct
BNM	Business Network Model
BNMA	Business Network Model Agent
BPEL	Business Process Execution Language
BPML	Business Process Modeling Language
BTF	Business Transaction Framework
BTML	Business Transaction Model Language
CC	Cloud Computing
CE	Conceptual External
CEC	Cross–Enterprise Collaboration
CPN	Colored Petri–Net
CI	Conceptual Internal
DB	Data Base
DGI	Distributed Governance Infrastructure
DIBPM	Dynamic Inter–Organizational Business Process Management
eBT	eBusiness Transaction
ECML	Electronic Contracting Markup Language
FIAT	Fluency; Interferability; Alternation; Transparency
eSML	eSourcing Markup Language
eSRA	eSourcing Reference Architecture
HTTP	Hypertext Transfer Protocol
IKW	Inter– and Intra–organizational Knowledge Worker
OEM	Original Equipment Manufacturer
OWL–S	Ontology Web Language **for** Services
PNK	Petri Net Kernel
PNML	Petri Net Markup Language
QoS	Quality of Service
SLA	Service–Level Agreement
SME	Small– and Medium–Sized Enterprise
SOBI	Service–Oriented Business Integration
SOA	Service–oriented architecture
SOAP	Simple Object Access Protocol
SOC	Service–oriented computing
SOCC	Service–oriented cloud computing
TBT	Trusted Business Transaction
Tx	Transaction
TxQoS	Transactional Quality of Service
UDDI	Universal Description Discovery and Integration
UML	Unified Modelling Language
WFMS	Worklfow Management System
WISE	Workflow Based Internet Services
WS–C	Web Service Coordination

WS–CDL	Web Services Choreography Description Language
WSCI	Web Service Choreography Interface
WSCL	Web Service Composition Language
WSDL	Web Service Definition Language
WSFL	Web Services Flow Language
WS–T	Web Service Transaction
XML	Extensible Markup Language
XPDL	XML Processing Description Language
XRL	eXchangeable Routing Language
XSLT	Extensible Stylesheet Language Transformations
XTC	eXecution of Transactional Contracted electronic services

Contents

1 Collaboration context for trusted business transactions 1

1.1 Introduction 1
1.2 Conceptual collaboration positioning 2
1.3 Research Design 5

2 Transactionality context 9

2.1 Introduction 10
2.2 Transaction types 10
2.3 Features of electronic business transactions 13
2.4 A transactional architecture for collaboration 17
2.5 Conclusion 19

3 Transaction-lifecycle formalization 21

3.1 Introduction 22
3.2 CPN-notation and state-space analysis 24
3.3 Top-level eBT-lifecycle protocol 25
3.4 Token colors of the eBT-lifecycle 26
3.5 Results of model checking 27
3.6 Conclusion 29

4 create 33

4.1 Introduction 33
4.2 BNM selection 34
4.3 populate 35
4.4 negotiate 36
4.5 Conclusion 37

5 perform - rollout 47

5.1 Introduction 47
5.2 contract establishment 49
5.3 operate 50
5.4 Conclusion 52

6 perform - rollback 65

6.1 Introduction 65
6.2 disruptively reset 67
6.3 nondisruptively choose 67
6.4 nondisruptively change 70
6.5 nondisruptively manage 73

6.6 Conclusion 73

7 Outlook: aligning transaction quality with business goals 77

7.1 Introduction 78
7.2 Specifying transactional quality of service 79
7.3 Nesting Abstract Transaction Constructs 81
7.4 Industry initiatives for e-business transactions 84
7.5 Conclusion 86

8 Conclusion 89

8.1 Research summary 89
8.2 Contributions 90
8.3 Limitations and future work 93

A BNM selection 95

A.1 Statistics 95
A.2 Boundedness Properties 95
A.3 Home Properties 100
A.4 Liveness Properties 100
A.5 Fairness Properties 100

B populate 101

B.1 Statistics 101
B.2 Boundedness Properties 101
B.3 Home Properties 105
B.4 Liveness Properties 105
B.5 Fairness Properties 106

C negotiate with forced agreement 107

C.1 Statistics 107
C.2 Boundedness Properties 107
C.3 Home Properties 114
C.4 Liveness Properties 114
C.5 Fairness Properties 114

D negotiate with forced counteroffer 117

D.1 Statistics 117
D.2 Boundedness Properties 117
D.3 Home Properties 123
D.4 Liveness Properties 123
D.5 Fairness Properties 124

E negotiate with forced disagree 125

E.1 Statistics 125
E.2 Boundedness Properties 125
E.3 Home Properties 132
E.4 Liveness Properties 132
E.5 Fairness Properties 132

F governance distribution and extraction 135
F.1 Statistics 135
F.2 Boundedness Properties 135
F.3 Home Properties 140
F.4 Liveness Properties 140
F.5 Fairness Properties 140

G prepare 141
G.1 Statistics 141
G.2 Boundedness Properties 141
G.3 Home Properties 145
G.4 Liveness Properties 146
G.5 Fairness Properties 146

H enact 147
H.1 Statistics 147
H.2 Boundedness Properties 147
H.3 Home Properties 150
H.4 Liveness Properties 150
H.5 Fairness Properties 150

I terminate 153
I.1 Statistics 153
I.2 Boundedness Properties 153
I.3 Home Properties 157
I.4 Liveness Properties 157
I.5 Fairness Properties 157

List of Figures

1.1 A conceptual business-collaboration model. 2
1.2 A three-layer business process framework [27]. 3
1.3 An extended SOA [53]. 4
1.4 Design-science research framework for the domain of information systems [33]. 5

2.1 Simplified informal models of distributed and nested transactions. . . 12
2.2 Simplified informal models of a chained transaction and Saga. 12
2.3 The levels and phases of the eBT-framework. 15
2.4 High-level figure of an eCommunity lifecycle for a cloud infrastructure. 16
2.5 Highest layer of eSRA. 17

3.1 Hierarchy of the eBT-lifecycle. 22
3.2 Distributed governance infrastructure. 23
3.3 Creation line for the DGI. 24
3.4 Creating and performing a cross-enterprise community transaction. . . 26

4.1 A collaboration pyramid in B2B (*create*). 34
4.2 Selection of a Business Network Model (*BNM selection*). 35
4.3 Adding roles to a BNM (*repository accessing*). 35
4.4 Populating a BNM with service offer, roles, and checking interoperability (*populate*). 39
4.5 Checking channels for interoperability requirements and data semantics (*interoperability checking*). 40
4.6 Finalizing the BNM-population (*finalize*). 41
4.7 Negotiating an eContract (*negotiate*). 42
4.8 Extracting an eContract-proposal for negotiation (*contract extraction*). 43
4.9 Finalizing an agreement on an eContract (*agreement finalizing*). . . . 44
4.10 Disagreement handling for an eContract (*disagreeing*). 45
4.11 Finalizing an eContract disagreement (*disagreement finalizing*). . . . 46

5.1 Protocol of the rollout service with business-semantics rollback (*perform*). 48
5.2 Establishing a DGI and operating it (*rollout*). 53
5.3 Distributing local contracts, policies and populating with electronic services (*contract establishment*). 54
5.4 Policy extraction from local contract copies and assignment of monitors and BNMAs (*governance distribution*). 55

5.5 Extracting from local contract copies the amount of required policies (*extraction*). 56
5.6 Electronic service choosing and communication endpoint creation (*prepare*). 57
5.7 Exception handling for service preparation (*preparation error*). 58
5.8 Protocol for enacting and terminating an eContract (*operate*). 59
5.9 Enacting an eContract and triggering business-semantics rollbacks or a termination (*enact*). 60
5.10 Terminating an eCommunity (*terminate*). 61
5.11 Removing policies and electronic services that are enacted (*policy service removal*). 62
5.12 Removing BNMAs, communication endpoints and local contracts (*observance removal*). 62
5.13 Finalizing an eContract disagreement (*roles removal*). 63

6.1 Options for rolling back violations in an eContract enactment (*business-semantics rollback*). 66
6.2 Establishing a distributed governance infrastructure and operating it (*disruptively reset*). 68
6.3 Distributing local contracts, policies and populating with local electronic services (*nondisruptively choose*). 69
6.4 Policy extraction from local contract copies and assignment of monitors and BNMAs (*remove*). 70
6.5 Policy extraction from local contract copies and assignment of monitors and BNMAs (*reinsert*). 71
6.6 Protocol for enacting and terminating an eContract (*nondisruptively change*). 72
6.7 Terminating an eCommunity (*removal*). 74
6.8 Enacting an eContract and triggering business-semantics rollbacks or a termination (*insertion*). 75
6.9 Electronic service choosing and communication endpoint creation (*nondisruptively manage*). 76

7.1 Trustworthiness injection in the eBT-lifecycle [64]. 79
7.2 An example for a nested ATC [64]. 82
7.3 Hierarchy of ATCs [64]. 83

List of Tables

2.1 Mapping the eBT-lifecycle into eSRA. 18

3.1 Data exchange of top-level protocol. 27
3.2 Acronyms, names and description of token colors. 30
3.3 Results from model checking. 31

5.1 Categorization of operate protocol. 50

Abstract

Corporations are pressured to engage ever more into agile business-networks of collaborations, which changes the requirements for computing systems. In a service-oriented and cloud-computing environment, the supporting infrastructure for cross-enterprise computing (CEC) needs to support the lifecycle of loosely-coupled, eContract-governed business networks. This lifecycle needs to support the selection of autonomously administered business services provided and used by corporations, contract negotiations, and monitoring of the contract behavior during enactment with the potential for breach management.

In this context, a new type of business transaction is required to safeguard the CEC across the entire lifecycle that relaxes the traditional properties of atomicity, consistency, isolation and durability for business-semantics induced consistency rules. This transactional safeguarding involves breach detection during CEC and recovery aspects. Thus, the research question addressed in this manuscript is how to safeguard the CEC-automation for several organizations in a heterogeneous system environment paying attention to the semantic complexities of trusted, electronic business transactions (eBT)? These semantic complexities differentiate an eBT from traditional database transactions in that it also incorporates separation levels of concern to tackle the collaboration complexity of pragmatic, semantic and technical aspects in a heterogeneous system environment.

To fill the detected gap, we define a eBT-lifecycle that spans the entire lifecycle as described above. Since the definition of this metamodel is based on a formal notation that comprises the definition structural and behavioral properties in a semantically deterministic way, we are able to perform model checking for correctness and performance. The formal modeling-notation allows the precise exploration of exception catching during a CEC and subsequent business-semantics business-semantics business-semantics rollback or termination so that the metamodel does not break down under an exploding mount of clutter that comes into existence along the lifecycle of a CEC. The result is a top-level eBT for a eBT-lifecycle that resembles a Saga-transaction and is at the same time an open-nested transaction that allows the incorporation of lower-level transactionalities. A Saga-transaction is long lived such as in a business process. Committed data may need a business-semantics business-semantics business-semantics rollback because of an occurring exception. However, unlike conventional business-semantics business-semantics business-semantics rollbacks, specific business logic is required to roll back a long lived transaction and restore the system to its original state.

The advantage of using a semantically deterministic modeling notation not only allows for simulation and verification of the eBT-lifecycle, exception- and compensation-management with a precise business-semantics business-semantics business-semantics rollback study to keep the metamodel free of clutter and sustainable. The approach also ensures metamodel-extensibility for conducting, e.g., further embedded explorations into eBT-related trust and reputation management, extensions for studying the nature of the breeding ecosystem for setting up CEC in a service-oriented and cloud-computing based environment, additional exception- and compensation-management measures. Finally, the realization of the eBT-lifecycle and safeguarding transactionality in an evaluation prototype is made feasible because of the semantic clarity of the

chosen modeling notation. Such evaluations are made easy as the eBT-lifecycle comprises nested services with clearly specified exchange protocols of known data sets.

To show the feasibility of the described eBT framework, industry initiatives such as business-transaction standards, are checked for eBT compatible characteristics. Since realizing an eBT framework raises many tricky issues and existing frameworks and standards do not meet CEC-requirements to safeguard, the manuscript finally maps out important research areas that require scientific attention for future work.

Chapter 1

Collaboration context for trusted business transactions

Contents

1.1 Introduction 1
1.2 Conceptual collaboration positioning 2
1.2.1 Collaboration-complexity management 3
1.2.2 Technological automation context 4
1.3 Research Design 5
1.3.1 Research Approach 6
1.3.2 Research question 6

With the emergence of service-oriented and cloud computing, companies embrace new ways of carrying out business transactions electronically. Since the parties involved in an electronic business transaction (eBT) manage a heterogeneous information-systems infrastructure within their organizational domains, the collaboration complexity is considerable and safeguarding a cross-enterprise collaboration (CEC) with an eBT is difficult, but of high significance. The conceptual framework of an eBT must pay attention to the complexities involved and differentiating characteristics that go further than traditional database transactions. Since an eBT comprises separate levels, the exploration of pre-existing transaction concepts is relevant for populating the respective levels.

1.1 Introduction

The emergence of electronic business promises for companies a sustainable market advantage that comprises an integration and coordination of information flow and product flow between heterogeneous information-system infrastructures. Such information flow that bridges different organizations, includes the linking of business elements into an integrated whole. Automating cross-enterprise collaboration (CEC) enhances efficiency and effectiveness. For the first case, it means a lowering of costs, savings in time budgets and resources such as personnel, machines, energy, and so on. For the latter case of effectiveness, the objective is to provide the quality of service that is in high market demand.

For CEC, a loose coupling of information systems is a requirement as a tight coupling of information systems results in too many agreement details and too much shared context has to be revealed to the business counterpart. In CEC, the registration of business transactions is of major legal importance for organizations. A business transaction [42] is a consistent change in the state of a business relationship that is driven by a well-defined business function. Each party in a business transaction holds its own business transaction. For CEC, a transaction concept is important to ensure reliability.

To facilitate a loose coupling and highly dynamic establishment of business collaboration, the service oriented computing (SOC) paradigm is increasingly important. Services are self-describing, logical manifestations of physical resources that are grouped as a process, i.e., as a set of actions [67] that an organization is prepared to execute and expose to the web.

With the complexity involved in CEC, no single transaction model is able to meet all requirements. Instead, it is necessary to inter-organizationally establish transaction frameworks in a way that does not force companies into disclosing an undesirable amount of business internals. In this paper, a conceptual model of an electronic business transaction (eBT) is put forward, based on an investigation of features that also incorporates business aspects and in which collaborating organizations safeguard their business internals. Note that an eBT needs to safeguard the legally binding contractual relationships between collaborating parties that dictate responsibilities and the consequences of behavior. The importance of business semantics in an eBT also has consequences for the nature of the atomicity, consistency, isolation, and durability properties.

1.2 Conceptual collaboration positioning

Observing business collaborations [44] in the business cases of this document, reveals characteristic features where an original equipment manufacturer (OEM) organizes the creation of value in an in-house business process that is decomposable into different perspectives, e.g., control flow of tasks, information flow, personnel management, allocation of production resources, and so on.

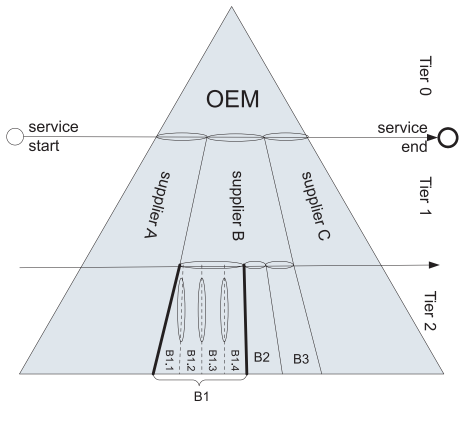


Figure 1.1: A conceptual business-collaboration model.

Figure 1.1 depicts conceptually a complex service of an OEM [48] with optional tangible elements, of which several need to be acquired from suppliers. The reasons for acquiring parts externally are manifold, e.g., the OEM cannot produce with the same quality, or an equally low price per piece, the production capacity is not available, required special know-how is lacking, and so on.

The horizontal ellipses in Figure 1.1 denote the client/server integration of outsourced in-house-process parts to lower-level clients who provide services to the vertically adjacent higher tier of a supply chain [49]. The outsourced business processes receive refinements by the respective suppliers. The refinements remain opaque to the service consumer and the supplier only has awareness of the OEM's outsourced respective process but the remaining in-house process remains opaque.

Vertical ellipses in Figure 1.1, depict a peer-to-peer (P2P) collaboration within a cluster of small and medium sized enterprises (SME). If several SMEs form a composed service in a P2P way [27], they become a supplier for a higher-level service consumer.

1.2.1 Collaboration-complexity management

In order to manage the complex business and conceptual collaboration and the heterogeneous technological environment in a supply chain, a three-layer framework [27] is a suitable model. The bottom of Figure 1.2 shows the internal layer that caters towards a heterogeneous system environment. Often organizations support their business processes by containing them in a hard-coded way in legacy systems. Examples of such legacy systems are applications for enterprise-resource planning, databases, accounting systems, applications for human-resource management, and so on. If the business processes of an organization are known and modeled, they are directly enactable by process-management applications, e.g., by intra-organizational workflow-management systems. Companies are reluctant to directly link their internal-layer legacy systems cross-organizationally to safeguard their information infrastructure and because of the fear they could disclose business internals that result in a loss of competitive advantages.

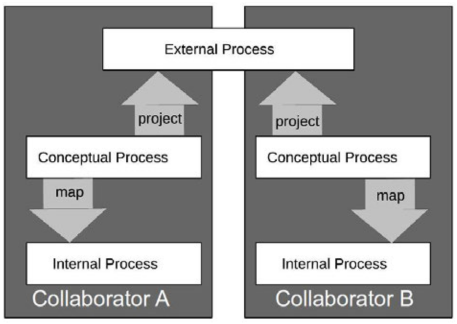


Figure 1.2: A three-layer business process framework [27].

At the conceptual layer, the business processes are designed independent from infrastructure and collaboration specifics. Conceptual processes are mapped to their respective internal layer for enactment. If a service-oriented architecture supports the conceptual-layer processes, their enactment allows the orchestration of Web-service wrapped legacy systems that are located on the internal layer. For the conceptual layer, it is important that collaborating parties can use a common denominator for cross-enterprise collaboration harmonization.

The external layer stretches across the domains of the collaborating organizations. Parts of the conceptual processes are projected to the external layer and compared by the collaborating parties. That way, the parties investigate the demands of service consumption and the ability of service provisioning. Since it can suffice to project a subset of process details to the external layer, an organization can determine which business internals should remain hidden from the counterpart. The process-based collaboration is automated and dynamically forged.

1.2.2 Technological automation context

Contemporary automation means are service-oriented computing (SOC) and increasingly cloud computing (CC). SOC [26] is a design paradigm to build computer software in the form of services, which refers to a set of related software functionalities that can be reused for different purposes, together with the policies that control its usage. Stateless services wrap legacy information systems such as for enterprise resource planning, customer relationship management, and so on, in a way that these legacy systems exchange messages through the Internet protocol. Stateful services orchestrate the ports of stateless services or choreograph stateful services. Service-orientation provides a governing approach to automate business logic as distributed systems. CC [47] is the delivery of computing as a service, whereby shared resources, software, and information are provided to computers and other devices as a utility over the Internet.

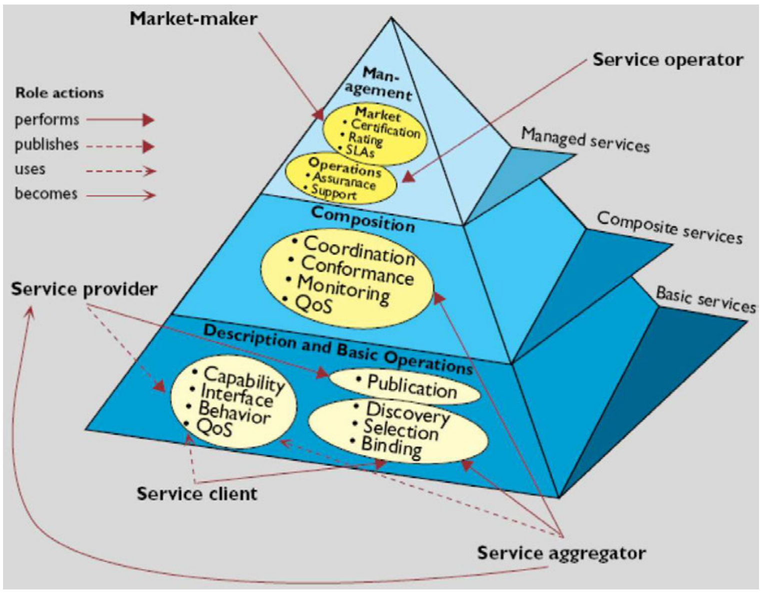


Figure 1.3: An extended SOA [53].

Service-oriented architecture (SOA) addresses IT-asset reuse and standardize communications between parties. The main idea is that collaborating organizations find each other and form a complex service together. Such a simple approach is not sufficiently suitable for the complexities involved in eBT. Thus, Figure 1.3 depicts a broader scenario that takes into account multiple service roles and their interactions. With respect to XML-based specification languages, the currently available set of standards is either not suitable for CEC and eBT or does not exist at all. Besides well established WSDL [16] for specifying the ports of services and SOAP [6] for delivering messages via the Internet protocol, BPEL [1] is a standard for defining stateful services. However, certainly the capstone of Figure 1.3 is only partially populated with suitable specification languages.

Next, after introducing the context for this research, we pose the research question for this manuscript that we split up into several sub-questions so that the complexity becomes manageable. The subsequent explanation details the meaning of the research questions.

1.3 Research Design

In this thesis, design-science research in the domain of information systems is followed as a research methodology. The design-science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts [33] that are broadly defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations(implemented and prototype systems).

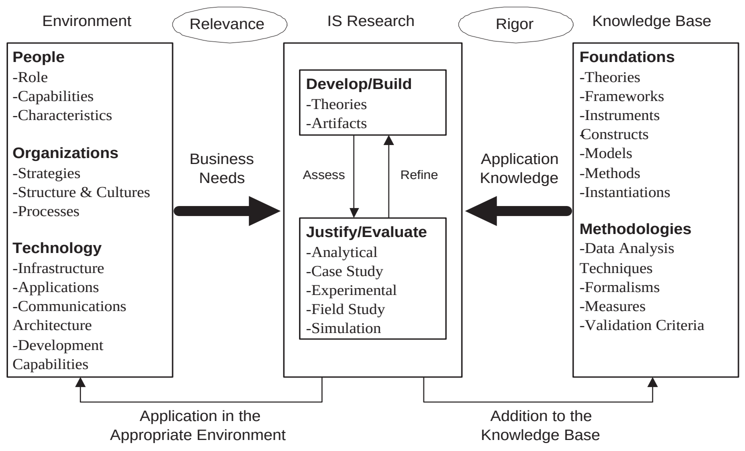


Figure 1.4: Design-science research framework for the domain of information systems [33].

In Figure 1.4, the essence of information-systems research framework is depicted. To the left, the environment pillar defines the problem space in which phenomena of interest reside consisting of people, organizations, and technology. Design-science research achieves relevance by building artifacts that address the business needs evolving from the environment.

To the right of Figure 1.4, the knowledge base delivers foundations and methodologies from and through which IS research is accomplished. Rigor is achieved by applying foundations in the develop/build phase and methodologies during the justify/evaluate phase. The results of design-science research are assessed to the business need in an appropriate environment and contribute to the content of the knowledge base for further research and practice. The next section shows how this thesis follows guidelines for conducting design-science research.

1.3.1 Research Approach

For conducting design-science research that adheres to the framework in Figure 1.4, guidelines exist [33] that are followed as a research approach in this thesis.

- *Problem Relevance*: The importance of safeguarding CEC with trusted business transactions (TBT) for the future competitiveness of companies is explained in this introduction chapter. For developing the concept of a transactional lifecycle, the business needs of CEC are relevant input.
- *Design as an Artifact*: This thesis proposes a formalization of the eCommunity lifecycle, equipped with trusted business transactionality that allows CEC without directly linking the respective information infrastructure of collaborating business parties. Furthermore, the formalized eCommunity concept allows to safeguard business privacy[1] and ensures flexibility for internally organizing tasks without violating the externally agreed-upon business collaboration agreements.
- *Research Rigor*: To understand in further detail, a conceptual exploration is used that results in a TBT-framework. As an advantage of a conceptual exploration, it is technology independent and it establish a uniform vocabulary of the universe of discourse. For TBT exploration, Petri nets [56, 57] are an integral part. Petri nets [56, 57] have been widely used to provide underlying semantics for business process models and the design of complex distributed systems. These formal semantics serve as an analysis technique for TBT that create a foundation for subsequent automation with means of cloud computing.
- *Design as a Search*: The developed eSourcing concept is part of a broader lifecycle of peer-to-peer TBT. We explore what comprises this broader notion of TBT and how the eCommunity concept model is embedded in it.
- *Design Evaluation*: With formal verification methods, an evaluation of TBT is performed by using Petri-net based tool support for soundness- and performance checking. Additionally, a feasibility study explores how existing technology from service-oriented and cloud computing is employable for realizing TBT.

The research approach we express in a concrete form through a number of specific research questions that result in answers for the research questions. The list of steps is as follows:

1.3.2 Research question

For the detected gap of a lacking transactionality framework, we pose the following research question. How can the automation of CEC be safeguarded in a heterogeneous system environment so that it pays attention to the semantic complexities of a trusted business transaction? Based on this main research question, we deduce five sub-questions.

[1]We define privacy as the rights of people, organizations or groups of organizations to determine for themselves for whom, for what purpose, to what extent and how information about them or information held by them is communicated to others [60].

- How is the eBT-lifecycle defined in terms of its collaboration pipeline for transactionality including its contained data flow?
- How is the orderly partial or complete termination of a lifecycle for an electronic business transaction ensured without leaving behind abandoned processes and database entries that lead to a cloud-collapse?
- Assuming a cloud-computing environment that caters for elastic resource assignment, where are performance spikes in the eBT-lifecycle?
- How is a system architecture specified for privacy protection so that it supports the automation of the eBT-lifecycle?
- How must the architecture be extended with features for ensuring the dependability of an eBT for cross-enterprise collaboration?

The approach for answering the posed research questions takes into account principles of semiotics [59] that differentiate the representation of signs and symbols in an information system by a pragmatic-, semantic-, and syntactic view. The first level focuses on the usage of symbols taking into account contextual issues including the characteristics of the person using the symbols, the task they are engaged in and the organizational context. The semantic level focuses on the meaning of symbols and finally, the syntactic level defines the form of symbols rather than their meaning. We term the latter as $technical$ level in the sequel as it captures better its nature in TBT.

With respect to the phases of a TBT, a broad classification comprises a setup phase with a maturation environment for services, a main enactment phase with transactional business-semantics rollback or compensation steps, and a post-enactment phase where events are logged that elude on the trustworthiness and reputation record of services and their assigned partners and affect later decision-making [58]. In order to support this business-level decision-making, the monitored transaction quality must be aligned with with business-level goals; we define dependability as the match between business-level service quality agreements with the observations from operational time monitoring of transactional quality.

For TBT, we assume a peer-to-peer collaboration exists where several enterprises engage in coopetition, i.e., a collaborative form of competition for the creation of complex services. Such a situation Figure 1.1 depicts for the case of $B1$ on Tier 2, e.g., equivalent to a cluster of small- and medium-sized enterprises (SME). On a larger scale, hyper-complex products such as the construction of the Airbus A380 is impossible without OEMs behaving on a larger scale as SMEs in a cluster.

Semantics plays an essential role to ensure the collaborating parties understand precisely with which services and partners they interact at what stage of a TBT-lifecycle. Thus, to answer the research questions posed above, a design notation is required that uses formal semantics that is deterministic to clearly specify the structure of collaboration phases and also to specify the data-exchange protocols. Simulation and verification means must accompany the chosen design notation to ensure model correctness. Only that way it is possible to ensure the business semantics of a TBT is controllable for managing exceptional collaboration situations. Without clarity on formal semantics, it is not possible to automate neither CEC nor the dependable transactionalities required for safeguarding business collaboration.

The remainder of the manuscript is structured as follows. Section 2 provides additional information relevant to understand the context for electronic business transactions. This includes a conceptual description of a TBT-lifecycle that requires transactional safeguarding and also a description of the system architecture for automating the setup-, enactment- and post-enactment phases of that lifecycle. Section 3 shows the top-level of the collaboration lifecycle using CPN-notation for formalization. As a result, service protocols are visible with their data-exchanges. As the lifecycle is formalized with nested services, the data flow exists on varying refinement levels. Furthermore, Section 3 lists the results from the tool-supported CPN-model checking that shows correctness and performance bottlenecks in the collaboration-lifecycle. Section 4 focuses on the formalization of the transactional lifecycle responsible for a distributed governance infrastructure that comes into existence when an agreed upon eContract is . Section 5 shows the lifecycle for establishing the contract on a technical level that consequently are enacted until a termination brings an eCommunity of transacting parties to a final state. Such a termination does not create computational waste Also Section 6 shows when and how Saga-resembling business-semantics rollbacks and compensations safeguard the lifecycle in a way where no waste occurs. Section 7 presents a conceptual outlook for aligning transaction quality with business goals in order to inject dependability in the prior formalized eBT-lifecycle. The outlook offers scope for future work into many directions. Finally, Section 8 concludes this manuscript by first summarizing the research work, secondly, giving the contributions achieved and finally, showing directions for future work.

Chapter 2

Transactionality context

Contents

2.1 Introduction 10

2.2 Transaction types 10

2.2.1 Internal-level transactions 10

2.2.2 Advanced transactions for the conceptual and external level of an eBT 11

2.2.3 Other related transaction-work 13

2.3 Features of electronic business transactions 13

2.3.1 Coordinating electronic business transactions 14

2.3.2 eBT-lifecycle for cross-enterprise collaboration 15

2.4 A transactional architecture for collaboration 17

2.4.1 The eSourcing Reference Architecture 17

2.4.2 Mapping the transactional lifecycle into eSRA 18

2.5 Conclusion 19

Taking into account the need for safeguarding cross-enterprise collaboration (CEC) and for building a service-oriented cloud-computing infrastructure, we first characterize the nature of electronic business transactionality (eBT) that reaches across organizational boundaries. With respect to the research question posed in Chapter 1, this section addresses the nature of the eBT-lifecycle, which is initially given conceptually and without data-flow. The eBT-lifecycle we assign to services in a system-architecture specification that ensures privacy protection of collaborating parties, which addresses another important issue of the research question. Accordingly, the structure of this section is as follows. After an introduction, we give in Section 2.2 an overview of the spectrum of traditional transaction features and order them according to categories. Section 2.3 conceptually presents a transaction framework with CEC business semantics and explains the behavior with incorporated business-semantics rollback-, and compensation mechanisms for handling exceptional collaboration situations. Section 2.4 uses the CEC-findings from the CrossWork project [44] for putting forward a conceptual system-application architecture into which we map the eBT-lifecycle. Finally, Section 2.5 gives a conclusion.

2.1 Introduction

The complexity of transactions that span multiple organizations rises in loosely coupled distributed computer networks that are enabled by service-oriented [48] and cloud computing. Here, business processes that use database systems must be cross-organizationally integrated. The transactional parts of a business process are referred to as a business transaction. By linking business processes cross-organizationally with electronic means, collaborating parties hope to safe time and money during the setup, enactment, and post-enactment of supply chains.

For electronic business transactions (eBT), that take place in a highly dynamic environment, applying conventional transaction mechanisms is not sufficient, as data from resources that are at the back-end of web services would need to be locked in order to assure atomicity and isolation. However, locking data for isolation in long running transactions for eBT is unrealistic as this might block resources that are consequently not available for others. For example, locking tables for selling a product blocks other potential customers, results in lower turn-over, and prevents other organizations from participating in the business process.

The synchronization of business processes between organizations must be part of a wider business-coordination protocol that defines the publicly agreed business interactions between business parties. Additionally, well founded possibilities are missing to compose eBT out of several transaction models [68] to support loosely coupled, long running transactions for heterogeneous systems. Thus, we conceptually address the research questions, how is the eBT-lifecycle defined in terms of its collaboration pipeline for transactionality and, how is a system architecture specified for privacy protection so that it supports the automation of the eBT-lifecycle?

2.2 Transaction types

Pre-existing transaction concepts from the database domain are available and assessed in this paper for eBT-management on different levels of concern. The remainder of this section is structured as follows. First, Section 2.2.1 discusses transaction management for the cross-organizational level of an eBT, followed by Section 2.2.2 that comprises advanced transactions for the conceptual and external level. Finally, Section 2.2.3 gives an overview of remaining miscellaneous transaction concepts.

2.2.1 Internal-level transactions

To traditionally manage data in a sound way, transactions must fulfill the following requirements. Atomicity states a transaction executes completely or not at all, consistency means a transaction preserves the cross-organizational consistency of the database, isolation infers a transaction executes as it were running alone with no other transactions, and finally, durability demands the results of a transaction are not be lost in a failure. These ACID-properties of flat transactions are instrumental for handling exceptions in transaction management that shield from applications running on top of databases.

Flat transactions still dominate the database world because of their simple structures and they are easily implementable. ACID-properties are suitable for the legacy systems of the cross-organizational level of an eBT as depicted in Figure 2.3. However, from a technical point of view, web-service composition faces the transactional challenges of relaxed atomicity, i.e., a situation where intermediate results may be kept without business-semantics rollback despite the failure to complete the overall execution of a composite service.

2.2.2 Advanced transactions for the conceptual and external level of an eBT

Advanced transaction models are extensions to flat transactions that release one or more ACID-constraints to meet with specific requirements. Two strategies have been adopted for extension purposes to achieve different structures inside a transaction, namely the modularization of a complex transaction with hierarchies, and the decomposition of a long-lasting transaction into shorter sub-transactions. In the sequel, the first type is referred to as distributed and nested transactions and the latter as chained transactions and Sagas.

A supporting concept for advanced transactions is the mechanism of savepoints [3] that enables a transaction business-semantics rollback to a intermediate state for recovery. Savepoints are important for supporting recoveries in distributed and nested transactions. Furthermore, the use of checkpoints in transaction logs is instrumental in chained transactions to indicate a point until which a business-semantics rollback does not result in an inconsistent database state. The sequel of this chapter explains details.

Distributed and nested transactions

A distributed transaction facilitates the integration of several database systems in a bottom-up way that reside on different servers in different geographic locations. As depicted in Figure 2.1, distributed transactions consist of sub-transactions that may access multiple local database systems and comprise two types of transactions [7], namely local transactions and global ones. Local transactions execute under the control of the local database management system (DBMS), while the multi-database system (MDBS) is in charge of global transactions. Hence, local and global integrity constraints need to align. Also the atomicity and isolation is managed globally when the whole transaction is aborted if any sub-transaction fails. The most influential example of distributed transactions is the X/Open Distributed Transaction Processing (X/Open DTP) software architecture [70] that is a standard for the two-phase commit (2PC) protocol. In combination with ACID transactions, a multiphase protocol like 2PC ensures sound database-state changes.

A nested transaction is a generalization of savepoints [24], which is suitable for complex-structured applications and adopts a top-down method to decompose a complex transaction into sub-transactions or child transactions according to their functionalities [46]. In nested transactions, parts of a transaction may fail without aborting the entire transaction. Sub-transactions compose in a hierarchical manner and only the leaf sub-transactions perform database operations while others function as coordinators. As Figure 2.1 shows, a sub-transaction is atomic and when it aborts, the parent may trigger another sub-transaction as an alternative without necessarily violating the database consistency.

Multilevel transactions are a variation of a nested transaction that are also called layered transactions [68] and their generalization is called open nested transactions [39]. In multilevel transactions, a transaction tree has its layers corresponding to those of the underlying system architecture. Here, a pre-commit concept allows an early commit-

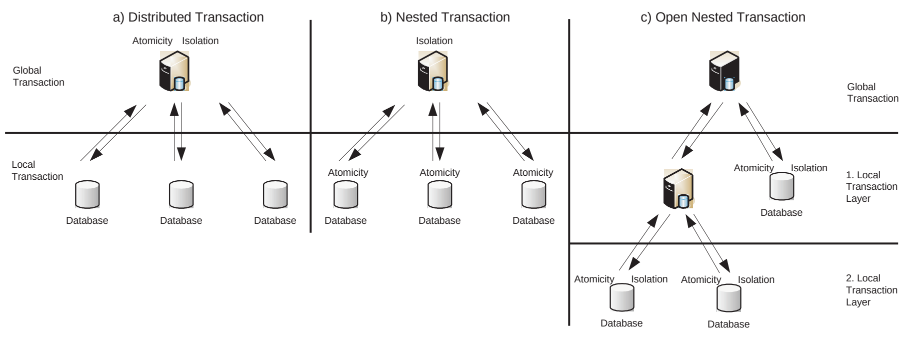

a) Distributed Transaction
b) Nested Transaction
c) Open Nested Transaction
Global Transaction
Local Transaction
Atomicity
Isolation
Database
1. Local Transaction Layer
2. Local Transaction Layer

Figure 2.1: Simplified informal models of distributed and nested transactions.

ment of a sub-transaction before the root transaction actually commits, which requires a sub-transaction to semantically undo the committed one.

Multilevel transactions evolve to open nested transactions if the structure of the transaction tree is no longer restricted to layering, i.e., leaves in different layers are allowed (Figure 2.1). Open nested transactions relax the ACID-properties compared to achieve a higher level of concurrency. The latter guarantee global-level isolation, which means in open nested transactions, the intermediate results of committed sub-transactions in nested transactions are invisible to other concurrently executing ones.

Chained transactions and Sagas

Differently to a nested transaction and its extensions, a chained transaction is appropriate for a time-consuming application with long-lasting transaction processes. As depicted in Figure 2.2, a chained transaction is a variation of savepoints [25] that decompose long running transactions into small, sequentially executing sub-transactions that roughly correspond to savepoint intervals.

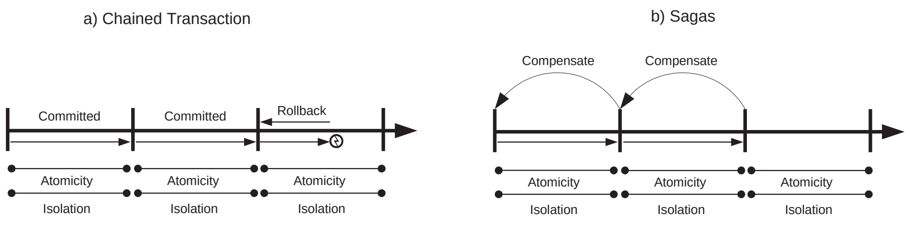


Figure 2.2: Simplified informal models of a chained transaction and Saga.

The difference is that each sub-transaction is atomic, while each interval between every two save points is part of an atomic transaction. In the chain, a sub-transaction triggers the next upon commit until the whole chained transactions commit. When encountering a failure, the previously committed sub-transactions have already durably changed the database so that only the results of the currently executing sub-transaction are lost. This way the business-semantics rollback only returns the system to the beginning of the most recently-executing sub-transaction.

Since the atomicity and isolation properties are relaxed in a chained transaction, this leads to aborting problems of the whole chain in the middle of execution as all the committed sub-transactions cannot be undone, other concurrent transactions see intermediate results generated during the execution of the chain.

Sagas [23, 19] adopt the idea of chained transactions of including a compensation mechanism to roll back. As shows Figure 2.2, Sagas divide a long lasting transaction into sequentially executed atomic sub-transactions with ACID-properties and each sub-transaction, except the last one, has its own compensating sub-transaction. When any failure arises, the committed sub-transactions are undone by compensating sub-transactions. Unlike chained transactions, Sagas can return the whole transaction back to the very beginning with compensations.

2.2.3 Other related transaction-work

Additional cross-enterprise business transaction concepts stem from the area of workflow-oriented research. Hence, the two most relevant research projects are described below. In [30], a two-layer transaction model, known as the WIDE transaction model, is presented. The model combines the concept of savepoints with Sagas [23] to offer that more flexibility in compensation paths in case of exceptions. The bottom layer consists of local transactions with a nested structure that conform to the ACID-properties [5]. The upper layer is based on Sagas that roll back the completed sub-transactions using the compensation mechanism, thus, relaxing the requirement of atomicity [63]. The semantics of the upper layer is formalized using simple set and graph theory [29]. The local transaction layer is designed to model low-level, short-living business processes, while the global transaction models high-level and long-living business processes.

The CrossFlow project [62] adopts the flexible approach of WIDE and develops the more comprehensive X-transaction model. The X-transaction model is a three-level, compensation-based transaction model to support cross-organizational workflow management, namely, the outsourcing level, the contract level and the internal level, each with a different visibility to the consumer or the provider organization. The X-transaction model views an entire workflow process as a transaction. For intra-organizational processes, smaller I-steps divide X-steps that adhere to ACID-properties. Each I-step has a compensating step in case of failure.

Next, we specifically focus on transactionalities enriched with business semantics.

2.3 Features of electronic business transactions

An eBT is a consistent change in the state of the business that is carried out with electronic means and that is driven by a well-defined business function. An eBT is automated, complex, long running and may involve multiple cross-organizational and external parties. Additionally, an eBT requires commitments to the transaction that needs to be negotiated by the participating organizations [42]. Further features of an eBT are: support for the formation of contracts, shipping and logistics, tracking, varied payment instruments and exception handling. Compared to traditional database transactions, eBTs have several distinguishing characteristics. Firstly, they extend the scope of traditional transaction processing as they may encompass classical transactions which they combine with non-transactional processes. Secondly, they group both classical transactions as well as non-transactional processes together into a unit of work that reflects the semantics and behavior of their underlying business task. Thirdly, they are governed by unconventional types of atomicity, e.g., payment atomicity, goods atomicity, delivery atomicity, contract atomicity. In [71], these unconventional atomicities are described in further detail.

Unconventional behavioral features [52] of an eBT are specified as follows: Generic characteristics tackle issues like who is involved in the transaction, what is being transacted, the destination of payment and delivery, the transaction time frame of permissible operations. Examples for special purpose characteristics are links to other transactions, receipts and acknowledgments, identification of money transferred outside national boundaries. Furthermore, advanced characteristics are the ability to support reversible and repaired transactions, the ability to reconcile and link transactions with other transactions, to specify contractual agreements, liabilities and dispute resolution policies, transactions that guarantee the integrity of information, confidentiality and non-repudiation; the ability for transactions to be monitored logged and recovered.

In the remainder, Section 2.3.1 first shows a conceptual framework for eBT-coordination followed by Section 2.3.2 that equally conceptually describes a Saga-stile eBT for the top-transaction level of that framework. The Saga-stile eBT we use for equipping a CEC system architecture with behavior and in the sequel we refine the first with a formal modeling notation.

2.3.1 Coordinating electronic business transactions

The integrated heterogeneous systems of a cross-enterprise collaboration need to be loosely coupled because of different reliability requirements that exist within long running eBTs. Different reliability requirements result from the properties of an eBT such as the phase the transaction is in and the level in which the transaction is taking place.

In [51], a phased model distinguishes between pre-transaction, main transaction and post-transaction phases in a collaborative business process. In [31], the need for a three-level process framework identifies companies as not willing to directly connect their legacy systems. Enabling interoperability between systems of different organizations is not just a matter of coupling systems, as this introduces interoperability problems, like semantic differences, autonomy, non-disclosures, company secrets, etc. Presenting the backend systems applications as services, i.e., wrapping them as web services is a valid solution for most of the problems mentioned.

Given the complex features of eBT as described, the conceptual model of Figure 2.3 represents an integration of the separate solution concepts [31, 51] that permits a manageable separation of concerns. In Figure 2.3, the organizational domains of a eBT-participants are bridged by an external level where companies cross-organizationally harmonize their business collaboration and, hence, their business transaction. Along a time line, the external-level phases of an eBT visualizes the need of coordination with the eBT phases on the conceptual level of an organization. Finally, the conceptual level coordinates the legacy system of the intra-organizational level, e.g., ERP systems, workflow systems, database systems, and so on. The latter give technical feedback to the higher level to inform about the success or failure of a transaction. Likewise, the conceptual level releases coordination information to the external level for aligning an eBT with the domain of the collaborating counterpart.

To control an eBT as depicted in Figure 2.3, spheres of control are a useful vehicle [48] to demarcate process parts of a collaborating party. The theory of spheres of control [4] originates from the domain of traditional database transactions. So called workflow spheres [40] expand the transaction theory into the dynamic world of complex business processes. Those concepts are applied in [34] for analyzing atomicity

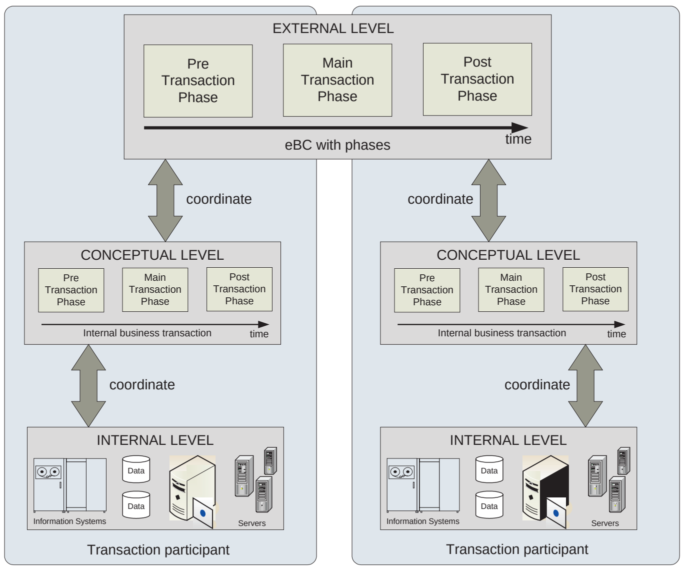


Figure 2.3: The levels and phases of the eBT-framework.

criteria dependencies and atomicity spheres without relating the workflow concepts of highly dynamic cross-organization processes. In the work of [61] a substantial emphasis is put on the characteristic atomicity properties of e-business. These unconventional atomicities for spheres in electronic business transactions (eBT) are explored and related [55] to each other along the categories system-level atomicity, business-interaction atomicity, and operational-level atomicity. The unconventional atomicities need to be part of a transaction model that pays attention to the business realities that form the context of CEC.

2.3.2 eBT-lifecycle for cross-enterprise collaboration

The proposed cloud-computing infrastructure supports the lifecycle of an eCommunity from inception to termination. Note that the high-level depiction of Figure 2.4 sums up a far more detailed model that we designed using CPN Tools[1] for designing Colored Petri-Nets. A CPN is a graphical oriented language for the design, specification, simulation and verification of systems. It is in particular well-suited for systems that consist of a number of processes which communicate and synchronize. Typical examples of application areas are communication protocols, distributed systems, automated production systems, or work flow analysis.

The eBT-lifecycle extends the eCommunity concept [38] and starts with the creation of a business-network model (BNM) that contains service offers which are first validated with service types, and additionally roles are assigned to the services. Concrete collaborating partners fill these roles during the eContract negotiation. Here, the partners that slip into roles, must vote on agreeing or rejecting an eContract proposal that is based on a picked BNM. Rejection terminates the eCommunity while having all partners agree, results in a consensual eContract passed on to the next service. A third

[1]http://wiki.daimi.au.dk/cpntools/cpntools.wiki

option during the negotiation phase is the proposal of a contract alternative.

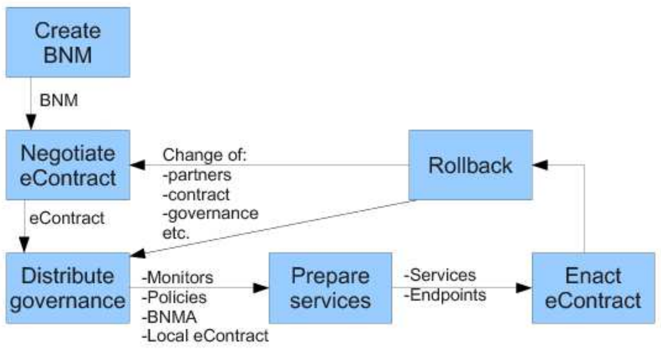


Figure 2.4: High-level figure of an eCommunity lifecycle for a cloud infrastructure.

In the service for distributing control governance, all partners who form an eCommunity receive a local copy of the agreed upon eContract. Next, all distributions receive local policies, monitors and BNM agents (BNMA) with facts deduced from the contract. The policies guide the monitored behavior of partners during enactment.

Once the governance structure is set up, technically realizing the behavior demanded in the local copies of the contracts requires concrete local electronic services. After picking these services follows a creation of communication endpoints so that the services of the partners are able to communicate with each other. The final step of the preparation is a liveness check of the channel-connected services. The enactment carries out the tasks contained in the local electronic services by an engine that propels the eCommunity business collaboration technically. Several alternative situations may occur during the enactment, namely, a total enactment termination, partner replacement, or policy violation.

To make the eCommunity-lifecycle a transactional one, the business-semantics rollback service concretely manages the alternative enactment situations. For termination, the entire distributed governance structure for an eContract is removed and the eCommunity is brought into a final ending state from where reuse is not possible. Partner replacement may either be disruptive in a sense that the governance infrastructure must be removed and a contract negotiation started from scratch. We assume a memory unit is available to notify the remaining partners to engage again in the formation of eCommunity. Non-disruptive partner replacement means the governance infrastructure remains entirely intact and a new eCommunity-partner slips into the existing local setup to replace an old partner. A policy violation may be treated with a reconciliation, ignoring it, replacing the partner, or replacing the policy. Finally, a local contract change means the removal of the entire local governance infrastructure and the respective partner has the chance to perform either a partial or entirely different local governance-infrastructure re-configuration within the framework of the overall eContract.

Next, we map the eBT-lifecycle into a system architecture for CEC. Note that CEC is widely addressed as sourcing by practitioners and thus, the we adopt the term $eSourcing$ for expressing the CEC-nature of the subsequent system-architecture discussion.

2.4 A transactional architecture for collaboration

The specification of the so-called eSourcing reference architecture (eSRA) [48] we closely relate to the earlier stipulated three-layer framework of Figure 1.2. eSRA is designed in accordance with the principle of functional decomposition of a system. This decomposition is also known as ”separation operation” and based on the part-whole principle. Thus, at each refinement level of eSRA, the identified services provide functionalities that do not overlap with the remaining services located on the same level. To achieve completeness, eSRA is designed in a top-down way, i.e., the services on the first level are decomposed into detailing services.

Initially, the first eSRA specification [48] is a facilitator for multiple organizations from industry that intend to take advantage from research results for designing their own cross-enterprise business-collaboration architectures. After the scenario-based evaluation [50], eSRA is concrete with respect to the services, interfaces and requirements. The following eSRA specification is semi-formal for easy understandability but it still provides a clear specification. The specification is abstract so that organizations that instantiate eSRA into their own architectures, have the freedom to use their respective concrete technologies.

2.4.1 The eSourcing Reference Architecture

The highest abstraction layer in Figure 2.5 shows two parties that contain the same set of services distributed across an external, conceptual, and cross-organizational layer [27]. The gray shaded boxes represent services and arcs depict data exchanges between them.

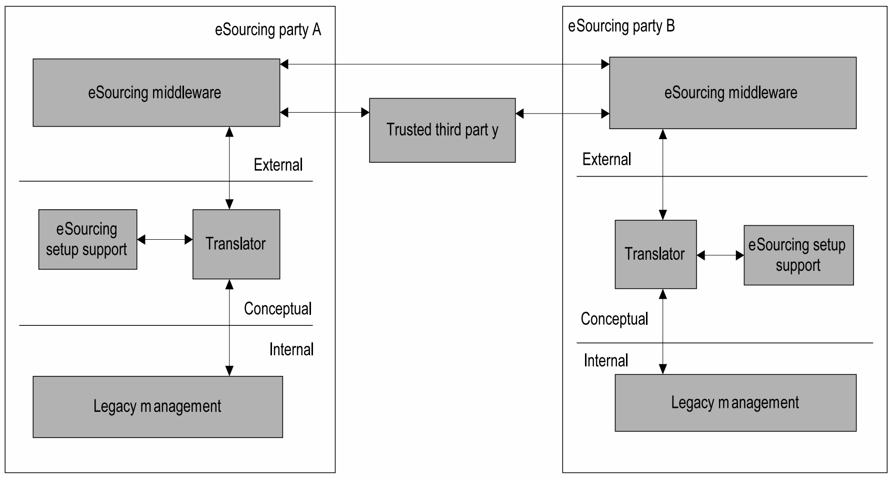


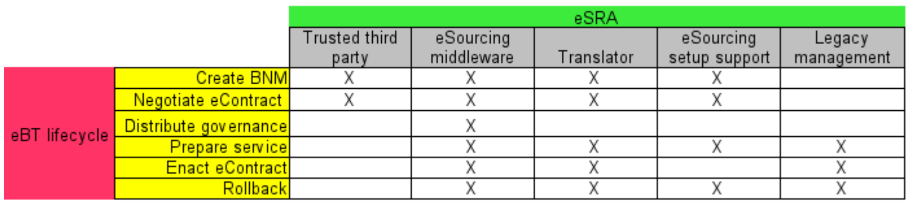
Figure 2.5: Highest layer of eSRA.

The *eSourcing middleware* is replicated on the respective external layers of collaborating parties. This service is the main enabler of interoperability and direct information exchange exists between the *eSourcing middleware* of each collaborating party to synchronize the respective services. Between the collaborating counterparts, a service is located termed *Trusted third party* that exchanges business-relevant information with the *eSourcing middleware*. A *Trusted third party* is necessary for several reasons. Firstly, collaborating parties expose service requests or service offerings to the *Trusted third party* for public evaluation. Secondly, the *Trusted third party* performs a verification of services and checks quality features of eSourcing configurations before enactment. If collaborating parties perform verifications and checks of eSourcing configurations themselves, they would need to reveal competitive secrets to each other, which is undesirable.

The conceptual layer of Figure 2.5 depicts two services, namely the *translator* and the *eSourcing setup support*. The *Translator* service exchanges information and translates it between the services located on the external and cross-organizational layer. The *eSourcing setup support* contains tools for modeling business rules and processes. Finally, the cross-organizational layer depicts a *Legacy management* service that interfaces with the translator service of the conceptual layer.

2.4.2 Mapping the transactional lifecycle into eSRA

The behavior in the eSRA-services of Figure 2.5 we determine by mapping in the eBT-lifecycle. Table 2.1 shows the highest-level assignment between services and the lifecycle stages from Figure 2.4.



		eSRA				
		Trusted third party	eSourcing middleware	Translator	eSourcing setup support	Legacy management
eBT lifecycle	Create BNM	X	X	X	X	
	Negotiate eContract	X	X	X	X	
	Distribute governance		X			
	Prepare service		X	X	X	X
	Enact eContract		X	X		X
	Rollback		X	X	X	X

Table 2.1: Mapping the eBT-lifecycle into eSRA.

BNM creation: The BNM is a blueprint for setting up an eCommunity. It emerges from a breeding ecosystem with a broker that serves as a rendezvous place for interested parties. Having a broker as a *Trusted third party* available reduces the communication overhead compared to a scenario where every respective party must be contacted separately for finding a service.

We assume a party with tools in the *eSourcing setup support* designs cross-organizationally the BNM-specification. The *Translator* channels the resulting specifications to the *eSourcing middleware*. The latter service transfers the BNM-specification to the *Trusted third party*.

Negotiate eContract: To exchange contract proposals, the *Trusted third party* serves as a deposition location. Such proposals the respective parties specify and modify cross-organizationally with tools from the *eSourcing setup support*. Via the *Translator*, the contract proposals are channeled to the *eSourcing middleware* where the distributed contracting negotiations take place.

Distribute governance: For establishing a governance infrastructure, the agreed upon eContract is replicated locally by the *eSourcing middleware*, policies are extracted, monitors and business network management agents.

Prepare service: The respective local governance infrastructure requires on a technical level a population with concrete electronic services. The latter are internally specified

with tools of the *eSourcing setup support* and mapped onto the *legacy management* via the *translator*-service that connects the electronic services with the distributed governance infrastructure in the external collaboration level.
Enact eContract: The essential services are the process- and rules engines in the *eSourcing middleware*- and the *legacy management*-service respectively. During the actual enactment, the *translator* converts data between the first two services so that the respective process- and rules-engines communicate successfully with each other.
Rollback: If exceptional situations occur and culminate in compensating business-semantics rollbacks, the most disruptive scenario is a return to the negotiation step. In that case all services are needed with the exception of the *Trusted third party*.

2.5 Conclusion

This chapter explains the evolution of traditional low-level database transactionality to advanced Saga-type transactions that are open and nested. Next, we specify a conceptual coordination framework for collaborative transactions that permit to integrate business semantics. This lifecycle of electronic business transactions we model and explain incorporating business-semantics rollback and compensation for exceptional events.

The transactional lifecycle populates the services of a privacy-preserving collaboration architecture. Privacy is ensured by using a layer pattern to protect business secrets that constitute competitive advantages and also to secure the legacy information-system infrastructure from corruption.

The collaboration architecture aligns with the specified eBT-framework as they both have a compatible layer architecture in common. That way, system implementers have a conceptual indication of service functionality and runtime behavior. The transactional lifecycle indicates business semantics and resembles an open, nested Saga transaction.

While the conceptual specifications succeed in organizing eBT-safeguarded collaboration, electronic automation requires further formalization. Thus, open issues are the semantically precise collaboration control-, and data-flow. That way precise features of transactionality are clarified in subsequent chapters.

Chapter 3

Transaction-lifecycle formalization

Contents

3.1 Introduction 22

3.1.1 Distributed governance infrastructure 23

3.2 CPN-notation and state-space analysis 24

3.3 Top-level eBT-lifecycle protocol 25

3.3.1 Input- and output states between main services 25

3.4 Token colors of the eBT-lifecycle 26

3.5 Results of model checking 27

3.5.1 Loops 28

3.5.2 Performance peaks 28

3.5.3 Markings 28

3.5.4 Liveness 29

3.6 Conclusion 29

The starting point for explaining the eBT-lifecycle is an overview of the services hierarchy and their data exchange in Section 3.1. The lifecycle leads to the creation of a distributed governance infrastructure for service enactment and safeguarding. We give a brief overview of the chosen formal design notation in Section 3.2, namely Colored Petri-Nets (CPN). An important advantage of CPN is that it allows for semantically deterministic design of system structure and also behavior that is verifiable for correctness and performance tests with tool support. Next, we show in Section 3.3 the top-level services designed in CPN and their protocol, i.e., input- and output states with data-flow. The token colors exchanged between the top-level services we explain in Section 3.4. Performance tests with the model-checking results listed in Section 3.5 indicate where in a cloud-computing implementation elastic resource assignment are important to prevent bottlenecks. Section 3.6 summarizes this chapter and answers the research questions.

3.1 Introduction

For allowing a fast comprehension of how the eBT-lifecycle formalizations relate to each other, the depiction in Figure 3.1 conceptually relates all services. The root is a circle in Figure 3.1 labeled *eCommunity lifecycle* has lower-level services depicted as rounded rectangles connected with arcs. The service-nesting direction is from the left side to the right side and the main lifecycle direction progresses from the top to the bottom. service pairs connect with directed arcs that show which services share exchange-protocols.

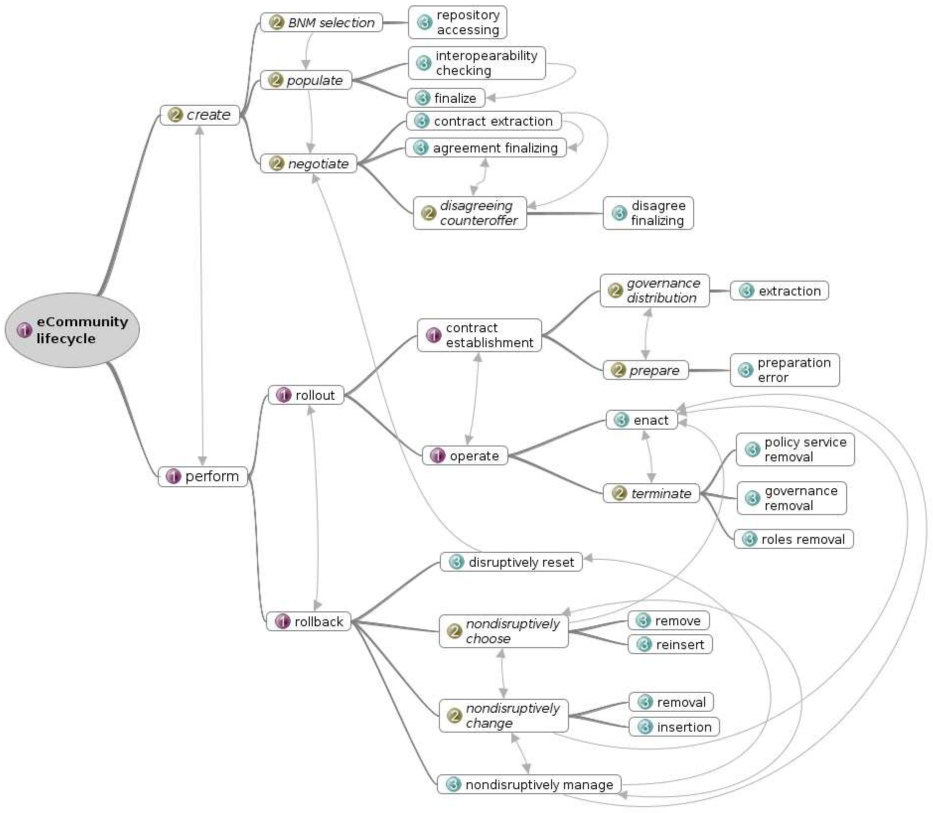


Figure 3.1: Hierarchy of the eBT-lifecycle.

Note that the $business - semanticsrollback$-branch has directed arcs connecting to services of other branches. Thus, the subsequent CPN-formalizations cater for exceptional situations that occur during enactment. In non-disruptive cases, the response is an orderly consolidation that leads to commenced enactment. In the disruptive case, the distributed governance infrastructure for eContract enactment is again orderly removed and the business-semantics rollback leads back to the negotiation stage. With orderly, we imply that the lifecycle is emptied so that no respective legacy-tokens related to the exceptional situation are left behind in the eBT-lifecycle that clutter to the point of collapse.

Based on the mapped out set of services for the eBT-lifecycle, this chapter addresses a subset of the research questions. Firstly, how is the semantic clarity of a trusted business transaction ensured? Once a way for developing semantic clarity is found, how is the eBT-lifecycle specified in detail including the data that flows through the service-protocol? What are the lifecycle features from a formal model-checking perspective and where are lifecycle-performance peaks?

Next, we show the distributed governance infrastructure (DGI) that is created for enactment in the eBT-lifecycle.

3.1.1 Distributed governance infrastructure

The depiction of the DGI in Figure 3.2 shows the hierarchy of elements for carrying out an electronic business transaction for the agreed upon eContract in which a set of eCommunity partners participate. In this DGI, the eContract that the eCommunity partners agree upon is a coordinating agent for the respective local copies. The latter have each a set of policies assigned that govern a partner's behavior. The additionally assigned monitors check for local behavior compliance and the business network model agent (BNMA) ensures compliance within the eCommunity.

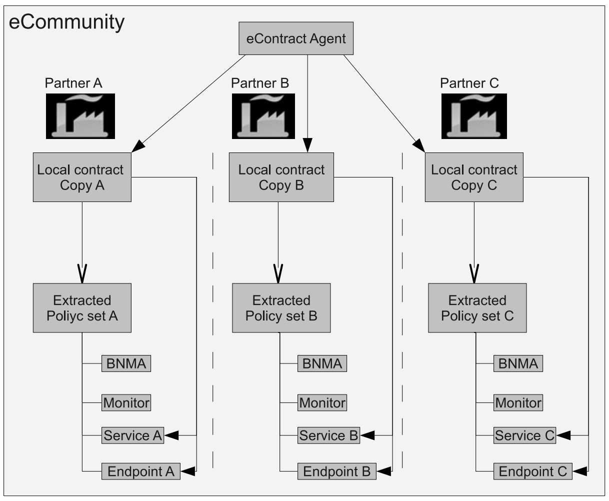


Figure 3.2: Distributed governance infrastructure.

In a time-line depicted in Figure 3.3, we show the creation sequence for the DGI-elements. If all eCommunity partners agree during the $negotiate$ stage, the eContract comes into existence that serves as a DGI-coordinating agent. In the *governance distribution*, local contracts come into existence for every eCommunity partner together with BNMAs and monitors. The $extract$ stage creates from the local contracts sets of policies and assigns each a BNMA and monitor. Finally, the $prepare$ stage populates the lowest technical DGI-level with matching services and corresponding endpoints for communication channels.

For dismantling the DGI, the $terminate$ stage removes the entire infrastructure step by step. Next, we explain the chosen CPN-notation for the eBT-lifecycle formalization.

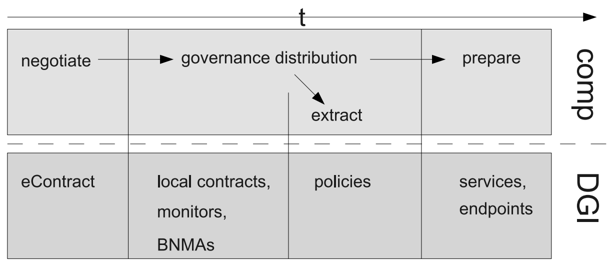


Figure 3.3: Creation line for the DGI.

3.2 CPN-notation and state-space analysis

Colored Petri Nets [35] is a graphical oriented language for design, specification, simulation and verification of systems. It is in particular well-suited for systems that consists of a number of processes that communicate and synchronize. Typical examples of application areas are communication protocols, distributed systems, automated production systems, workflow analysis.

CPN has an intuitive, graphical representation that consists of a set of modules (pages) each containing a network of places, transitions and arcs. The modules interact with each other through a set of well-defined interfaces in a similar way as known from many modern programming languages. Places may hold multiple tokens that carry color, i.e., attributes with values. Transitions fire when all input places hold the required sets of tokens and produce condition-adhering tokens into output places.

The formalized eBT-lifecycle of this section is translated into so-called $state\ spaces$ for analysis. The basic idea underlying state spaces is to compute all reachable states and state changes of the CPN-model and represent these as a directed graph where nodes represent states and arcs represent occurring events. Next, the state-space graph is translated into a *strongly connected service graph* (SCC-graph). The nodes in the SCC-graph are subgraphs called strongly connected services (SCCs) and informally explained, free of loops that may be contained in the state-space graph. The structure of the SCC-graph comprises useful information about the overall behavior of the model being analyzed.

Following the state-space analysis reports in the appendices, the checked properties we informally explain as follows. If the number of nodes in the state space and SCC-graph is equal, it means the state space is free of circles that would result in the model not terminating. The $boundedness\ properties$ tell how many (and which) tokens a place may hold considering all reachable markings. The best $upper\ integer\ bound$ of a place specifies the maximal number of tokens that can reside on a place in any reachable marking. The $best\ lower\ integer\ bounds$ for a place specifies the minimal number of tokens that can reside on the place in any reachable marking. The *best upper multi-set bound* of a place specifies for each color in the color set of this place, the maximal numbers of tokens that is present on this place with the given color in any reachable marking. The *best lower multi-set bound* of a place specifies for each color in the color set of a place the minimal number of tokens that is present on this place with the given color in any reachable marking.

The home properties tell us that there exists a single *home marking* M_{home}, which is reachable from any marking. This means that it is impossible to have an occurrence sequence which cannot be extended to reach M_{home}. In other words, it is not possible to end up in a situation that makes it impossible to reach M_{home} while that does not infer a guarantee.

The *liveness properties* cover several aspects. A transition is $live$ if from any reachable marking we can always find an occurrence sequence containing the transition. If every transition is live then a CPN is live in its entirety. A *dead marking* is part of the liveness properties, which is a marking where no binding elements are enabled. A dead marking can be a home marking because any marking can be reached from itself by means of the trivial occurrence sequence of length zero. A transition is $dead$ if there are no reachable markings in which it is enabled. If a model has dead transitions, it corresponds to parts of the model that can never be activated. Hence, we can remove dead transitions from the model without changing the behavior of it.

The motivation for the fairness property is to detect the transitions in a CPN-model that can not fire infinitely often while being enabled infinitely often. There are four fairness notions, namely, $impartial$ if a transition occurs infinitely often in every infinite run of a CPN-model. A transition is $fair$ if it occurs infinitely often in every infinite run of the model where the transition is enabled infinitely often. A $just$ transition occurs infinitely often in every infinite run of the net where it is continuously enabled from a marking onward. Finally, a transition is $notfair$ if it is not just. $Impartial$ considers all infinite runs while $fair$ and $just$ only consider some infinite runs.

3.3 Top-level eBT-lifecycle protocol

The eBT-lifecycle has two main services on its highest level, namely the $create$- and the $perform$-service as depicted in Figure 3.4. The first comprises a service-breeding ecosystem for populating a business-network model (BNM) with service-types and roles assigned. The BNM then is populated with service-offers and concrete partners who form the eCommunity, and the ability to exchange via communication channels is checked. The resulting BNM is comparable to a proto-contract that enters a negotiation phase which turns into a concrete eContract when all partners agree. It is possible to propose counteroffers that commence new negotiations with voting or when only one partner disagrees, the eCommunity comes to a sudden end. Note, the $creat$-service has been extensively published about and in [37] the reader may learn more details.

The $perform$-service to the right of Figure 3.4, falls again in two parts, namely, first one for setting up a distributed governance infrastructure that culminates in an eContract enactment. This involves the distribution of local copies to every eCommunity-member, the extraction of local policies from the contracts, the assignment of monitors that observe policy adherence, and a business-network monitor agent (BNMA). Additionally, concrete electronic services that adhere to the service-offers in the local contract, populate the established distributed governance infrastructure. The second part of the $perform$-service comprises several transactional business-semantics rollback steps that Section 3.1 describes.

3.3.1 Input- and output states between main services

Between the $create$- and the $perform$-services, the depicted top-level protocol of Figure 3.4 has as a starting state labeled *start lifecycle*. It comprises unique identification numbers for eCommunity lifecycles that are chosen when a BNM with service types and roles is ready for population with service offers and eCommunity partners. This

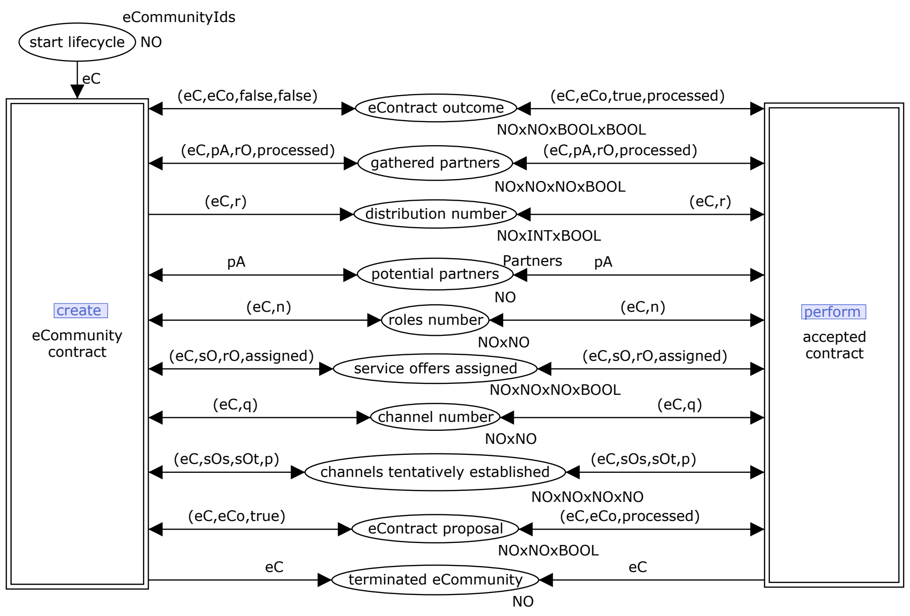


Figure 3.4: Creating and performing a cross-enterprise community transaction.

unique number identifies the entire lifecycle until it reaches the terminal state labeled *terminated eCommunity*.

The protocol between the services connects different lower-level services, as Table 3.1 shows. The left column lists the input- and output states between the $create$- and the $perform$-services that comprise the protocol. The next two columns to the right list the lower-level services to which the protocol-states between $create$ and $perform$ connect. The state *start lifecycle* is only connected to the $create$-service, which is the reason for an x placed in the $perform$ column. While the eCommunity identification from the *start lifecycle* is an input state for the lower-level $populate$-service, the remaining states are either also or exclusively connected to the $negotiate$-service. Finally, *terminated eCommunity* is the output state for the atomic transition labeled *close eCommunity*.

The $perform$ column shows that the second till forth row of protocol-states connect to the lower-level *governance distribution*. The second row and the fourth till seventh row in Table 3.1 show the protocol-states connect five times to $business - semanticsrollback$ services of which two cases are disruptive business-semantics rollbacks. Finally, the third and sixth till eleventh row connect within the $perform$-service to $termination$ of an eBT-lifecycle.

3.4 Token colors of the eBT-lifecycle

The token colors in the CPN-model for the eBT-lifecycle are available on different refinement levels. These colors are data types exchanged between respective services listed in Figure 3.1 in which the refinement direction is from left to right. Table 3.2 lists all colors with their hierarchic service-refinement availability mentioned in the

	services	
states	**create**	**perform**
start lifecycle	populate	x
eContract outcome	negotiate	governance distribution, terminate, nondisruptively change (rollback)
gathered partners	negotiate	governance distribution, terminate
distribution number	negotiate	governance distribution, nondisruptively change (rollback)
potential partners	negotiate	nondisruptively choose (rollback)
roles number	populate,negotiate	terminate, disruptively reset (rollback)
service offers assigned	populate,negotiate	terminate, disruptively reset (rollback)
channels tentatively established	populate,negotiate	terminate
channel number	populate,negotiate	terminate
eContract proposal	populate,negotiate	terminate
terminated eCommunity	close eCommunity (atomic)	terminate

Table 3.1: Data exchange of top-level protocol.

left column, meaning that 1 is the top-level and 6 the lowest refinement in Figure 3.1. Token colors listed for availability in a certain hierarchy level are present for all lower- but not for any higher hierarchy levels.

The fourth column of Table 3.2 textually explains the purpose of a token color for an eBT-lifecycle. The types of the token colors are either integer, string or boolean. In the first case, the integer mostly represents a token identification number and a string is either an eContract-negotiation outcome or an eContract-proposal extracted from a business-network model. Tokens of the type boolean represent decision points in an eBT-lifecycle. Since a CPN-model is a simplified version of a distributed system, these token colors of the model either represent in concrete application implementations more elaborate database tables, or XML schemata with more refining properties.

3.5 Results of model checking

From a developer perspective, the motivation for analyzing the eBT-lifecycle models in the sequel of this paper is to see if they terminate correctly, which is testable with a token game in CPN Tools. The listed services in Table 3.3 are pragmatically chosen with respect to their testability. Given the problem of state-space explosion and the relevance of test results for system developers, the listed services are either atomic refinement leaves without CPN-modules or comprise a module that itself only contain atomic transitions. Note, we term CPN-modules as services in this manuscript as it adheres to the service-oriented cloud-computing paradigm. The state space of the overall transaction lifecycle is too big for testing in a computationally feasible way. However, we employ a token-game simulation for evaluating the termination correctness of an eBT-lifecycle in the state *terminated eCommunity*.

A summary of the analysis results we provide in Table 3.3 where the first column lists the services of the eBT-lifecycle. For the $negotiate$-service there are three separate outcomes. Either, all eCommunity partners agree on a contract proposal, or a counteroffer is newly negotiated, or one partner disagrees entirely and terminates the proposal. For model checking, we separately generate results where only one respective outcome option is enabled and two remaining options disabled. Finally, the very right of Table 3.3 lists the appendices comprising respective model-checking results.

3.5.1 Loops

Detected loops in a model mean the system implementers must think carefully about enforceable termination criteria. Detected performance peaks mean, during runtime, provisions must be in place for elastic resource assignment, which is important in cloud-computing environments.

Loops exist when the state space has more nodes and arcs than the SCC-graph. If the boundedness properties reveal peaks in token numbers and the liveness properties of transitions show differences, performance peaks for respective transitions are given, which is indicated with a corresponding transition label. Further performance-peak indicators stem from fairness properties of transitions. i.e., we assume implementations of impartial transitions to perform most heavily and implementations of transitions that are not fair to perform lightly.

The checking results in Table 3.3 show that *loops* exist in the services for contract negotiation and enactment. In the first case, such loops are visible for extracting contracts until a copy exists for every eCommunity partner. Additionally, the finalization comprises loops for preparing the service-block for the subsequent eContract-proposal negotiation. Finally, a counteroffer triggers a loop, which the SCC-graph statistics in Appendix D show as the number of nodes is lower than in the state-space statistics. For the second case of enactment, a loop occurs for repeatedly stopping and starting the enactment of a service until either an exception occurs or the overall enactment enters a termination phase. The test results for remaining services in Table 3.3 show they do not contain loops.

3.5.2 Performance peaks

The column for *performance peaks* in Table 3.3 result from checking the boundedness- and fairness results in the appendices. Thus, we assume high numbers in tokens in the upper best integer bound of input- and output states in combination with enhanced transition fairness show performance peaks in respective services. With the exception of the services *BNM selection* and *terminate*, the remaining services in Table 3.3 have peaks that predictably require elastic resource assignment in a cloud-computing environment. For the $populate$-service, peaks occur not only for populating roles but also for checking if channel requirements and data-semantics match.

For all three $negotiate$ cases, contract extraction represents a peak and for the cases with forced agreement and a counteroffer, performance peaks occur in agreement finalizing when the $negotiate$-service is prepared for the next eContract negotiation. For negotiation with counteroffer, predictable performance peaks occur in the distribution of new contracts to eCommunity partners. For *governance distribution* and $extraction$ of policies (see Figure 4.8) are performance demanding. Finally, in the $preparation$-service, assigning electronic services is most performance intensive, followed by creating and publishing their endpoints and checking for operationality.

3.5.3 Markings

While no tested service has any home marking, in Table 3.3, the model-checking results for dead markings differ. We infer that having multiple dead markings and no home markings means the testing of implementations is more time- and resource intensive as only a big number of test cases ensure correctness. According to Table 3.3, testing the $populate$-service reduced to a counteroffer loop.

3.5.4 Liveness

The practical relevance of liveness checks for a service-oriented cloud-computing environment is that dead transitions are never used functionality in a service. Live transitions are functionalities of a service used at least sometimes. This means, system implementers must ensure for high runtime robustness of such functionality. If there is no consistent home marking and multiple dead transitions, developers should expect increased testing efforts of services.

The liveness column in Table 3.3 shows if dead (D) or life (L) transitions are present, i.e., ND means no dead transitions are present and NL means no live transitions are in a CPN-model. $D^{\star}$ in Table 3.3 means the model-checking results reveal a conditional dead marking. I.e., the model-checking results show the dead markings result from intentional disabling of marking paths for the purpose of focusing in specific marking paths under investigation. Right of columns for home- and dead markings, we mention the respective appendices with the model-checking results.

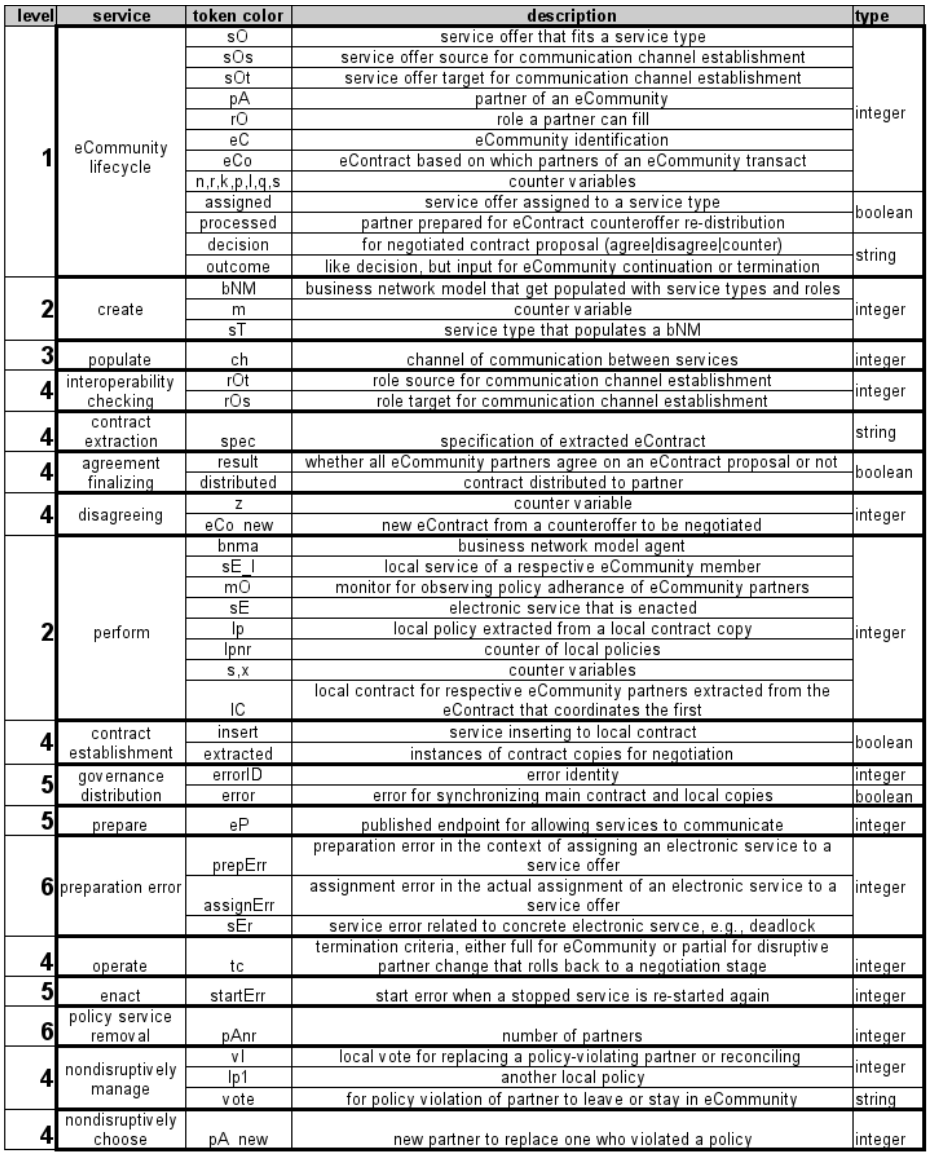
With respect to $liveness$, all checked services comprise no live transitions. That means no transition is always fired in any marking of a service. The test results for dead markings indicate no transition in a service is never fired in any marking (which we indicate as ND). For a subset of services in Table 3.3, we assign $D^{\star}$, which means test results show there exist intentionally dead transitions. When disabling certain marking paths, i.e., decision types in negotiate, to focus on marking parts in respective services. Thus, checking the model results in the respective appendices in detail for services with $D^{\star}$-liveness in Table 3.3, the results show dead transitions are from marking paths we deliberately disabled.

3.6 Conclusion

This chapter shows the full set of nested services for the eBT-lifecycle, their relationship to each other and data-exchange directions. For these data sets, a table shows their visibility at respective refinement levels of service nesting. The eBT-lifecycle establishes a distributed governance infrastructure for enacting eContracts. We explain the elements that comprise the DGI and show at which lifecycle stage what service creates and relates them into an infrastructure.

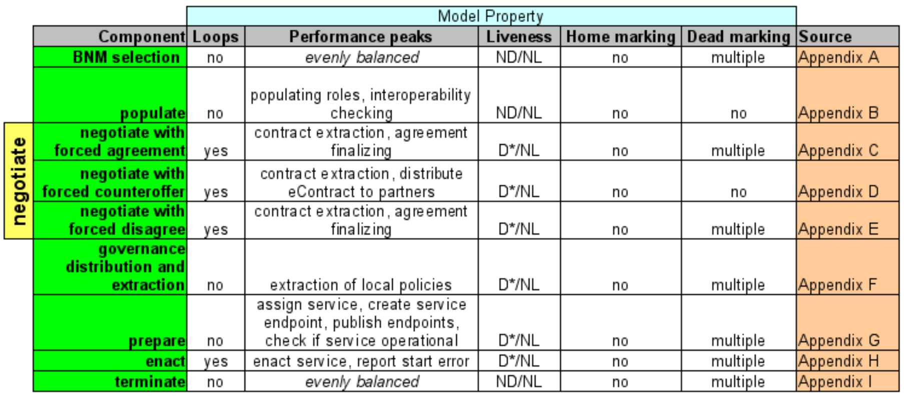
We formalize the eBT-lifecycle with Co loured Petri Nets and show the top-level services for the creation, enactment, business-semantics rollback and orderly termination of a DGI that is coordinated by an eContract for an eCommunity. The protocol between the two services show what color tuples assigned to tokens flow. Based on formalizations of nested services in subsequent chapters that refine the presented top-level of this chapter, we list at which hierarchy-visibility level these color tuples exist during a eBT-lifecycle.

The result of verification and model checking of leave services in the nesting hierarchy reveal their correctness and performance peaks that would require elastic resource allocation in a service-oriented cloud-computing environment. The checks reveal that performance peaks exist largely for populating roles with eCommunity partners, contract- and policy extraction, and enactment. The detailed formalizations of nested services of the eBT-lifecycle we discuss in subsequent chapters. The appendices of the manuscript show detailed model-checking and verification results that are input for the results of this chapter.



level	service	token color	description	type
1	eCommunity lifecycle	sO	service offer that fits a service type	integer
		sOs	service offer source for communication channel establishment	
		sOt	service offer target for communication channel establishment	
		pA	partner of an eCommunity	
		rO	role a partner can fill	
		eC	eCommunity identification	
		eCo	eContract based on which partners of an eCommunity transact	
		n,r,k,p,l,q,s	counter variables	
		assigned	service offer assigned to a service type	boolean
		processed	partner prepared for eContract counteroffer re-distribution	
		decision	for negotiated contract proposal (agree|disagree|counter)	string
		outcome	like decision, but input for eCommunity continuation or termination	
2	create	bNM	business network model that get populated with service types and roles	integer
		m	counter variable	
		sT	service type that populates a bNM	
3	populate	ch	channel of communication between services	integer
4	interoperability checking	rOt	role source for communication channel establishment	integer
		rOs	role target for communication channel establishment	
4	contract extraction	spec	specification of extracted eContract	string
4	agreement finalizing	result	whether all eCommunity partners agree on an eContract proposal or not	boolean
		distributed	contract distributed to partner	
4	disagreeing	z	counter variable	integer
		eCo_new	new eContract from a counteroffer to be negotiated	
2	perform	bnma	business network model agent	integer
		sE_l	local service of a respective eCommunity member	
		mO	monitor for observing policy adherance of eCommunity partners	
		sE	electronic service that is enacted	
		lp	local policy extracted from a local contract copy	
		lpnr	counter of local policies	
		s,x	counter variables	
		lC	local contract for respective eCommunity partners extracted from the eContract that coordinates the first	
4	contract establishment	insert	service inserting to local contract	boolean
		extracted	instances of contract copies for negotiation	
5	governance distribution	errorID	error identity	integer
		error	error for synchronizing main contract and local copies	boolean
5	prepare	eP	published endpoint for allowing services to communicate	integer
6	preparation error	prepErr	preparation error in the context of assigning an electronic service to a service offer	integer
		assignErr	assignment error in the actual assignment of an electronic service to a service offer	
		sEr	service error related to concrete electronic servce, e.g., deadlock	
4	operate	tc	termination criteria, either full for eCommunity or partial for disruptive partner change that rolls back to a negotiation stage	integer
5	enact	startErr	start error when a stopped service is re-started again	integer
6	policy service removal	pAnr	number of partners	integer
4	nondisruptively manage	vl	local vote for replacing a policy-violating partner or reconciling	integer
		lp1	another local policy	
		vote	for policy violation of partner to leave or stay in eCommunity	string
4	nondisruptively choose	pA_new	new partner to replace one who violated a policy	integer

Table 3.2: Acronyms, names and description of token colors.



		Model Property					
	Component	Loops	Performance peaks	Liveness	Home marking	Dead marking	Source
	BNM selection	no	*evenly balanced*	ND/NL	no	multiple	Appendix A
	populate	no	populating roles, interoperability checking	ND/NL	no	no	Appendix B
negotiate	negotiate with forced agreement	yes	contract extraction, agreement finalizing	D*/NL	no	multiple	Appendix C
negotiate	negotiate with forced counteroffer	yes	contract extraction, distribute eContract to partners	D*/NL	no	no	Appendix D
negotiate	negotiate with forced disagree	yes	contract extraction, agreement finalizing	D*/NL	no	multiple	Appendix E
	governance distribution and extraction	no	extraction of local policies	D*/NL	no	multiple	Appendix F
	prepare	no	assign service, create service endpoint, publish endpoints, check if service operational	D*/NL	no	multiple	Appendix G
	enact	yes	enact service, report start error	D*/NL	no	multiple	Appendix H
	terminate	no	*evenly balanced*	ND/NL	no	multiple	Appendix I

Table 3.3: Results from model checking.

Chapter 4

create

Contents

4.1 Introduction **33**
4.2 BNM selection **34**
4.2.1 repository accessing 35
4.3 populate **35**
4.3.1 interoperability checking 36
4.3.2 finalize 36
4.4 negotiate **36**
4.4.1 contract extraction 36
4.4.2 agreement finalizing 37
4.4.3 disagreeing 37
4.5 Conclusion **37**

This chapter uses CPN-notation to specify the refined, nested elements of the $create$-service in Figure 3.4, including the data-flow protocol between them. The models continue answering the research questions relating to semantic clarity of the elements that comprise the eBT-lifecycle. After an introduction in Section 4.1 about the $create$-service, the remaining sections explain the nested refining behavior and services. Section 4.2 explains the breeding ecosystem for creating business network models (BNM). The $populate$-service as explained in Section 4.3 prepares the chosen BNMA into a proto-contract that requires the consensus of the eCommunity-partners, which the $negotiate$-service in Section 4.4 manages. Section 4.5 concludes the chapter.

4.1 Introduction

The functionality in the *create*-service of Figure 4.1 shows that a nested service labeled *BNM selection* is an ecosystem for breeding service types that become part of so-called business-network models (BNM). The latter is a cross-enterprise collaboration blueprint to insert service types and roles for the next step in the eBT-lifecycle, namely the population with service offers and eCommunity partners. The BNMs that emerge from the breeding ecosystem exist permanently for repeated use in the subsequent populating stage, i.e., conformance-validated service offers and BNMs. The

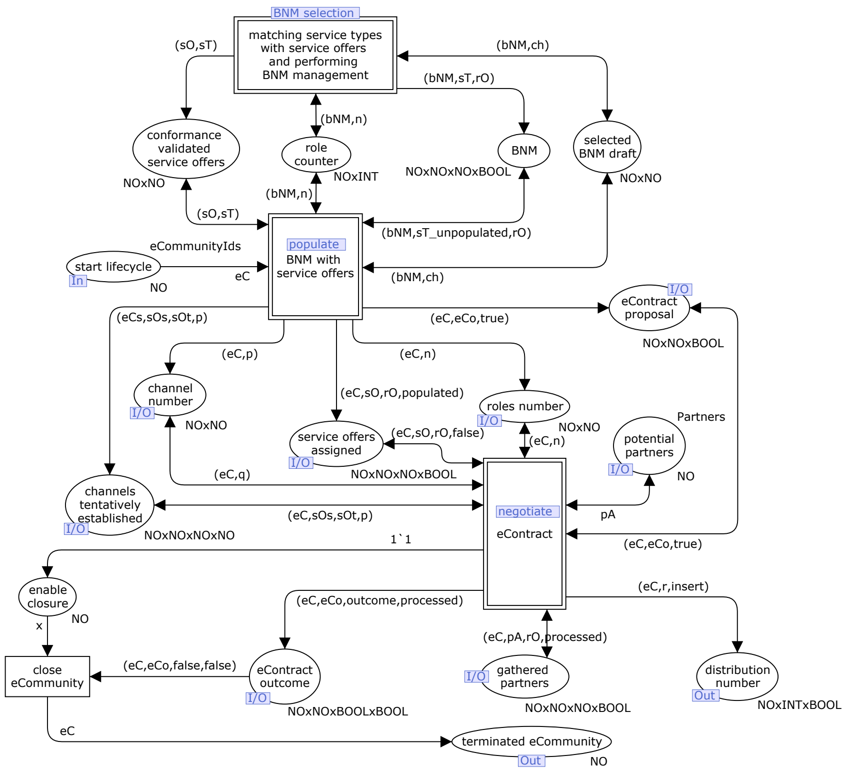


Figure 4.1: A collaboration pyramid in B2B (*create*).

populate-service validates the inserted service offers against the service types of the BNM as it emerges from the breeding ecosystem.

At the end of the *populate*-phase, a proto-contract exists for a *negotiate* step that is carried out by the eCommunity partners. The negotiation of the proto-contract has three different outcome options. An agreement of all eCommunity partners establishes the eContract for subsequent rollout and enactment; a counter-offer from only one eCommunity partner triggers a business-semantics rollback to the inception of the *negotiate*-service; finally, a disagreement of only one eCommunity partner results in a complete termination of not only the contract negotiation but additionally, the eBT-lifecycle also suddenly terminates with the identification ending in the state labeled *terminated eCommunity*.

4.2 BNM selection

This service functions as an ecosystem to breed BNMs. For that, a repository exists in the *BNM selection* of Figure 4.2 for which we assume users insert service types over time that they specify themselves. The same assumption holds for the repository of service offers in *BNM selection*, which is correspondingly a state in the CPN-model labeled *repository service offers*.

To be considered as a service offer for populating a BNM, beforehand the passing

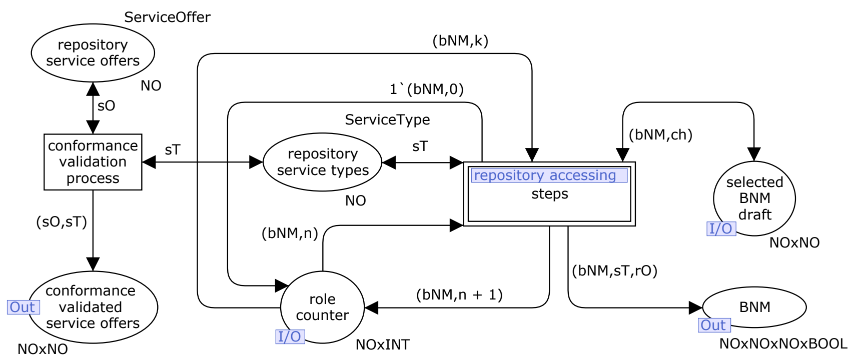


Figure 4.2: Selection of a Business Network Model (*BNM selection*).

of a conformance validation as depicted in Figure 4.3. The actual BNM selection involves choosing a BNM draft for adding validated service offers and roles to be filled subsequently with respective eCommunity partners.

4.2.1 repository accessing

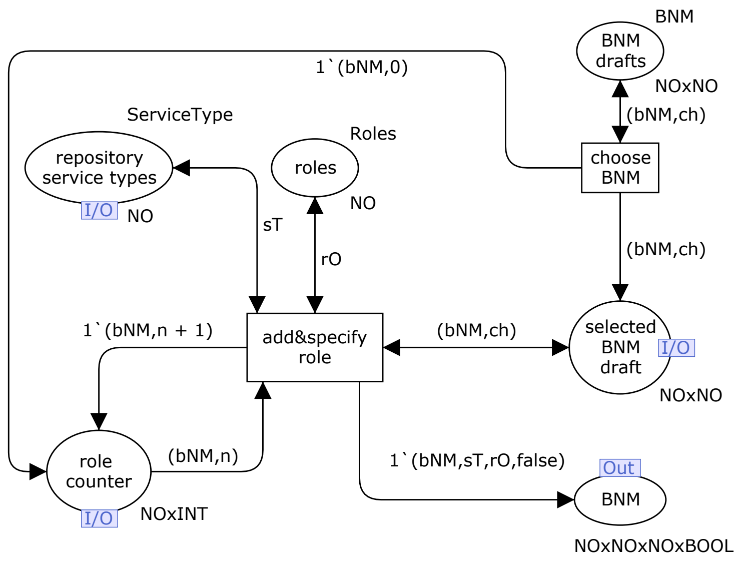


Figure 4.3: Adding roles to a BNM (*repository accessing*).

4.3 populate

In this service depicted in Figure 4.4, an unique eCommunity number identifies the entire eBT-lifecycle. The service types in the chosen BNM draft specification are populated with conformance-validated service offers. Next, an interoperability check of those service offers ensures the channels are capable of exchanging data that matches semantic expectations. When the proto-contract with tentatively established channels is ready, the population stage is ready for the next BNM population.

4.3.1 interoperability checking

The service offers chosen for populating a BNM are in the state labeled *populated*. In Figure 4.5, for the tentatively established channels between service offers, the ch token color contained in the BNM indicates the required amount of channels. Consequently, a chosen service offer becomes a channel source and another one a channel target that must adhere to given channel requirements and data-semantics matching. The interoperability checking ends when the number of required tentative channels exists, which enables the completion stage of BNM population.

4.3.2 finalize

To complete the population of BNMs, the *finalize*-service of Figure 4.6 empties the *populate*-service of remaining tokens. In an application-system implementation, this emptying corresponds to emptying an instantiation of not needed database entries and abandoned computing processes that lead to a collapse of the overall system. Such a collapse and restart is undesirable if many eCommunities are active to transact at the same time.

In Figure 4.6, finalizing the BNM population commences with an enabling of completion, continues with a reset and extracts important values for the subsequent proto-contract negotiation. The extracted values are the amount of roles, the proto-contract with tentative channels, a channel counter, assigned service offers. Eventually, the *finalize*-service concludes the BNM-populating stage.

4.4 negotiate

Once the *negotiate*-service of Figure 4.7 enables from a completed BNM population that generates a proto-contract, an eContract-proposal is extracted from that input from the *populate*-service. All eCommunity-partners receive an eContract copy who vote with three possible options.

As Figure 4.7 shows, the *negotiate*-service caters in an ideal case for an agree of all eCommunity partners, which establishes a consensus to make an eContract come into existence and triggers a preparation for the subsequent extraction of a DGI. Secondly, one eCommunity-partner decides on a modification and proposes a counteroffer for an eContract. The counteroffer instantaneously disables all voting options, removes the already casted votes and redistributes a copy of the modified contract for every respective eCommunity-partner. Thirdly, in the case of only one disagreement, the voting process is halted, again, it triggers the removal of already casted votes and terminates the entire eBT-lifecycle. Subsequently, the eCommunity identification comes to rest in the state labeled *terminated eCommunity*.

4.4.1 contract extraction

After choosing an eCommunity for which a business-network model (BNM) has been prepared, a pool of potential partners serves to fill the contained roles with concrete partners. Next, in the *contract extraction* service of Figure 4.8, the eContract-proposal emerges from the *population*-service, extending specifications of the tentatively created service-offer channels. The created eContract-proposal is ready for the subsequent steps of negotiation.

4.4.2 agreement finalizing

When all eCommunity-partners vote on an eContract-proposal, the objective is to achieve a consensus by all agreeing. In that case, the service *agreement finalizing* of Figure 4.9 empties the *negotiation* of eCommunity-partner instances and the offered services are reset for the next negotiation phase. Finally, the negotiation outcome is set to true.

4.4.3 disagreeing

This service depicted in Figure 4.10 activates in both cases of an eContract-proposal disagreement and also for the issuance of a counteroffer by an eCommunity-partner. For both cases, the calculated number of eCommunity-partners serves as input to empty the *negotiate*-service.

A disagreement leads to the removal of all eCommunity related tokens, the lifecycle termination and the identification token moves to its final state labeled *terminated eCommunity*. However, with a new counteroffer, the existing eCommunity-partners must cast votes again and as such, each partner must receive a contract-proposal copy of the counteroffer. All other eCommunity-related tokens remain in their respective states.

disagreement finalizing

When an eCommunity-partner casts a vote in a negotiation phase indicating a disagreement, this service performs the final clearance preparation for the subsequent DGI establishment, which is is depicted in the *disagreement finalizing* service of Figure 4.11. Setting the negotiation outcome to $false$ starts the removal of the eCommunity partners, followed by a removal of service offers and channels, which completes the disagreement finalization.

4.5 Conclusion

The formalization of the $create$-service with CPN-notation gives deep insight in its refinement with respect to control-flow, data-flow and protocol between nested services. The refinement specifies the elements of a breeding ecosystem for a business network model that inserts service-type conformance-validated service-offers. Additionally, in the breeding ecosystem the business network model receives roles. The ready BNM progresses into the $populate$-service for filling concrete eCommunity-partners with roles and tentatively establishing communication channels. The resulting proto-contract enters the $negotiate$-service where after an eContract-proposal extraction, all eCommunity-partners grant a vote. When one party disagrees, the entire lifecycle terminates instantaneously. A counteroffer results in canceling the ongoing voting procedure and a repeat of extracting an eContract-proposal for a new voting process. Only when all partners agree, a consensus creates a valid eContract as output of the $create$-service.

In Chapter 3, we list all token colors and their respective visibility in the refinement hierarchy that are relevant for the $create$-service. Furthermore, Chapter 3 also shows the model-checking and verification results of the $create$ CPN-models.

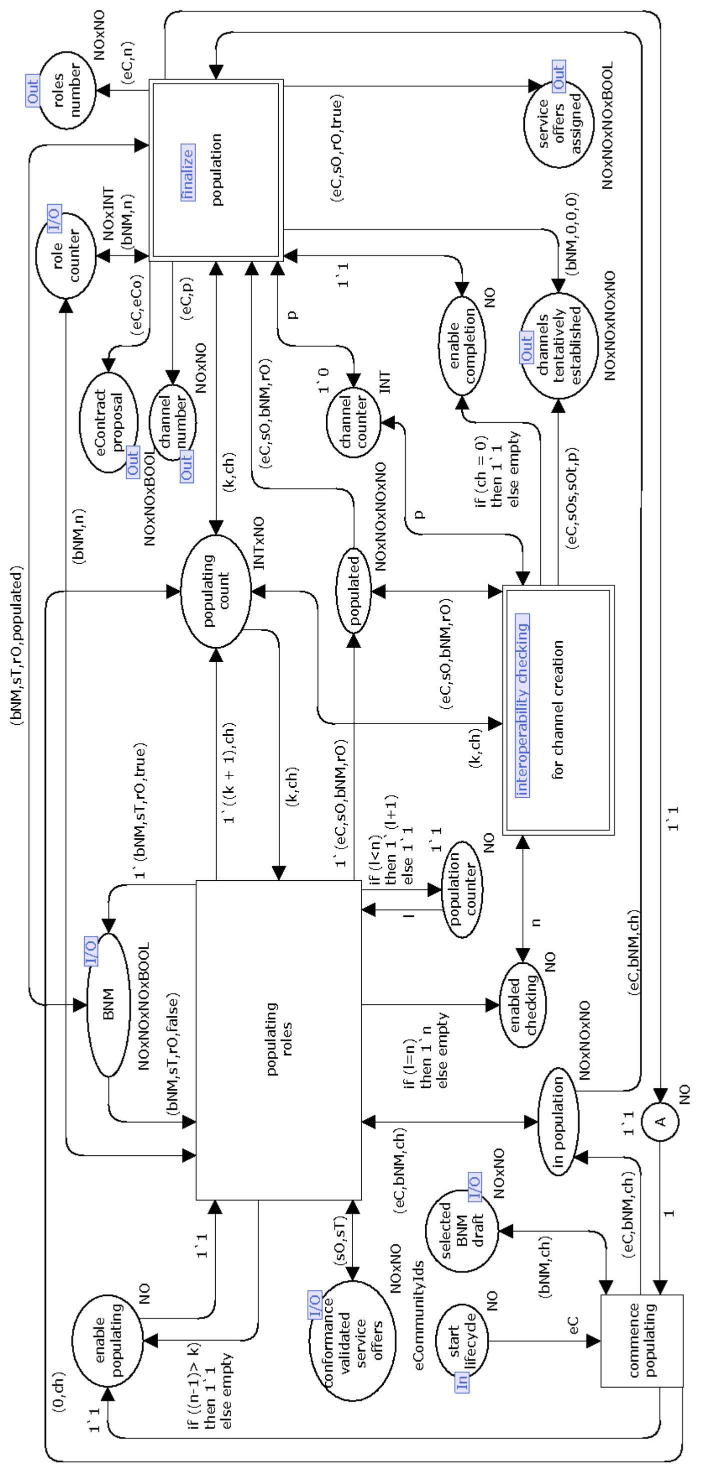


Figure 4.4: Populating a BNM with service offer, roles, and checking interoperability (*populate*).

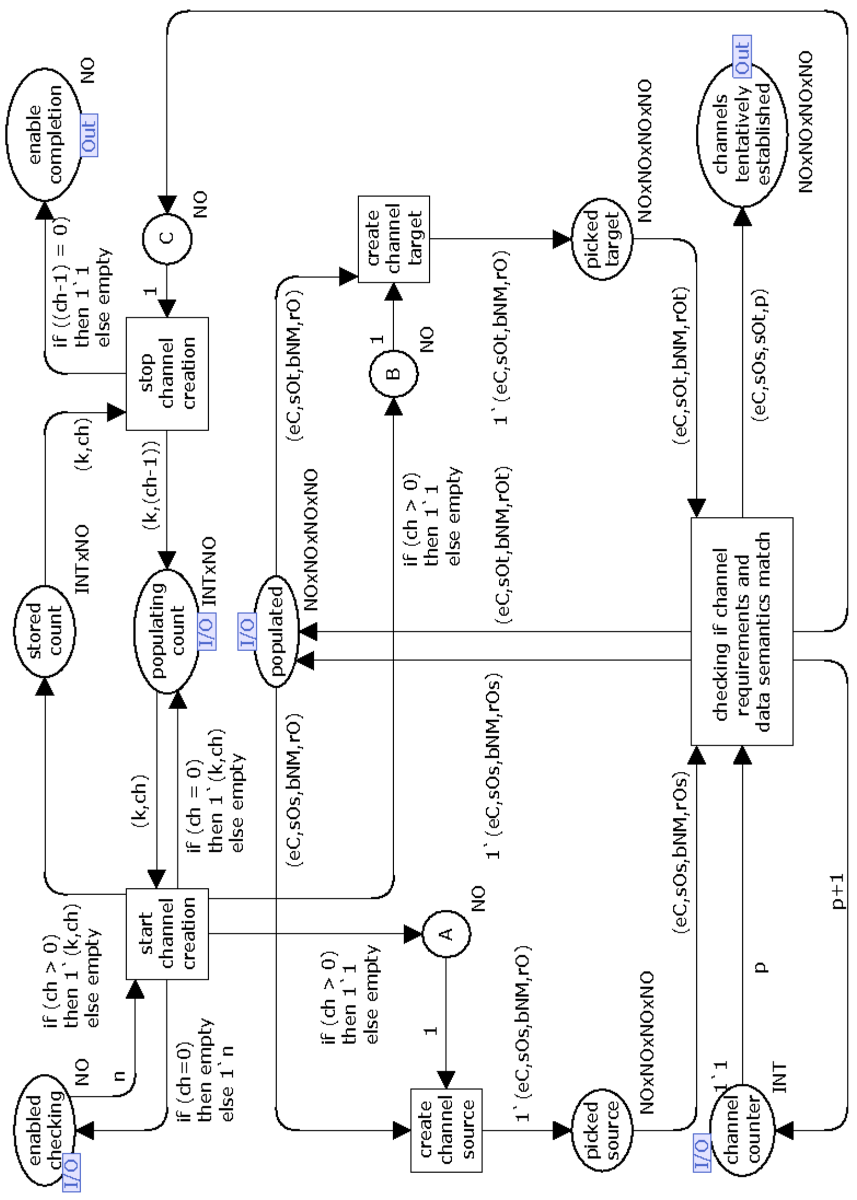


Figure 4.5: Checking channels for interoperability requirements and data semantics (*interoperability checking*).

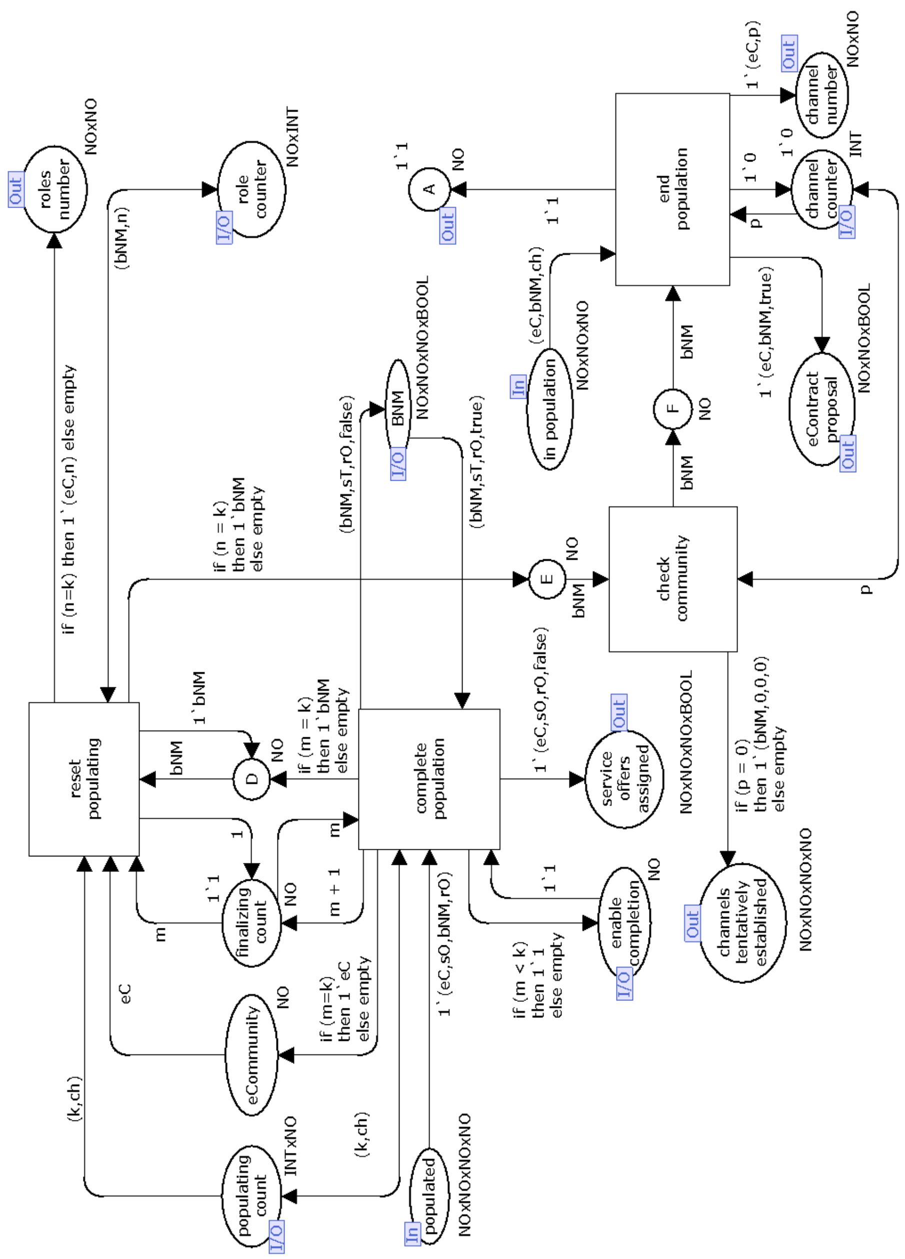


Figure 4.6: Finalizing the BNM-population (*finalize*).

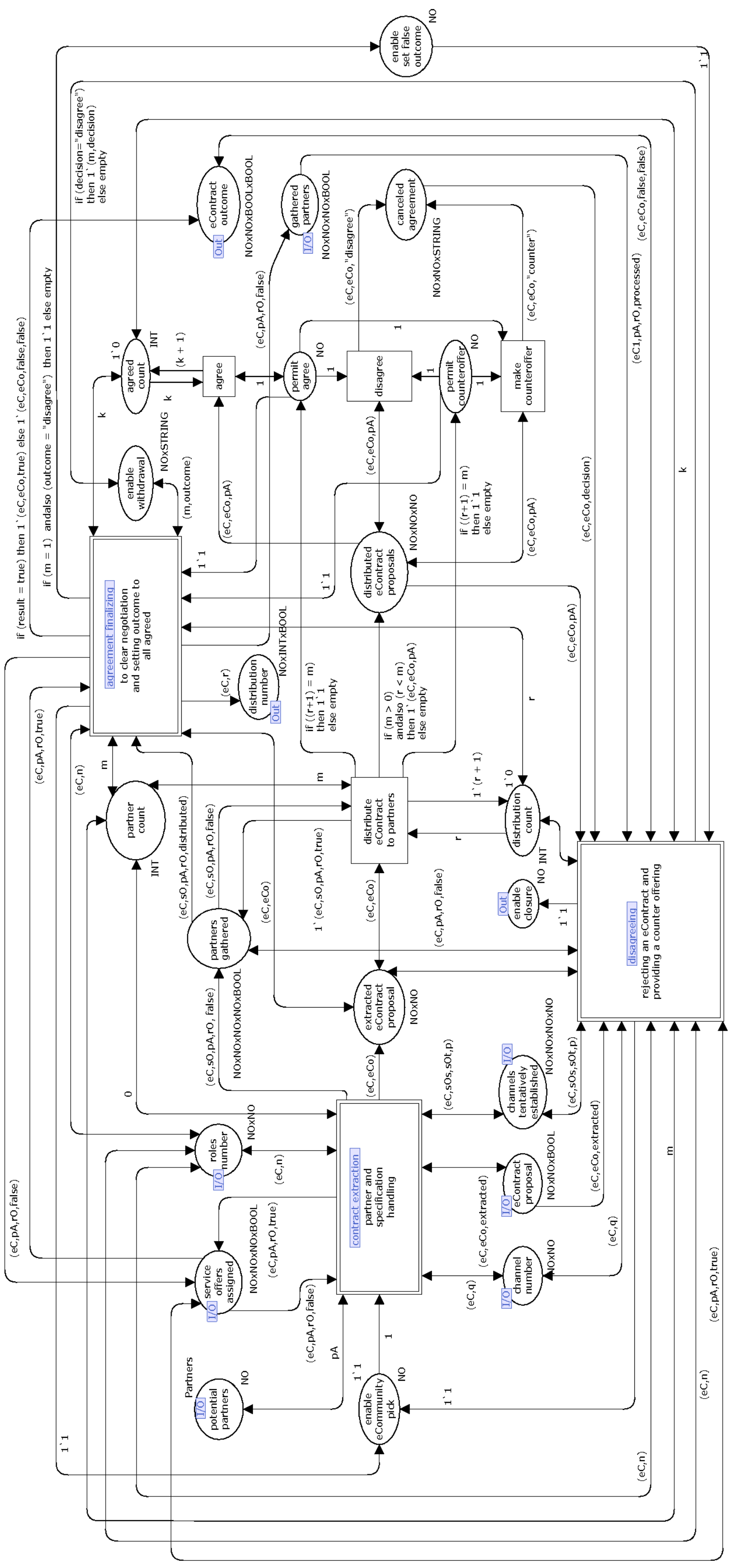


Figure 4.7: Negotiating an eContract (*negotiate*).

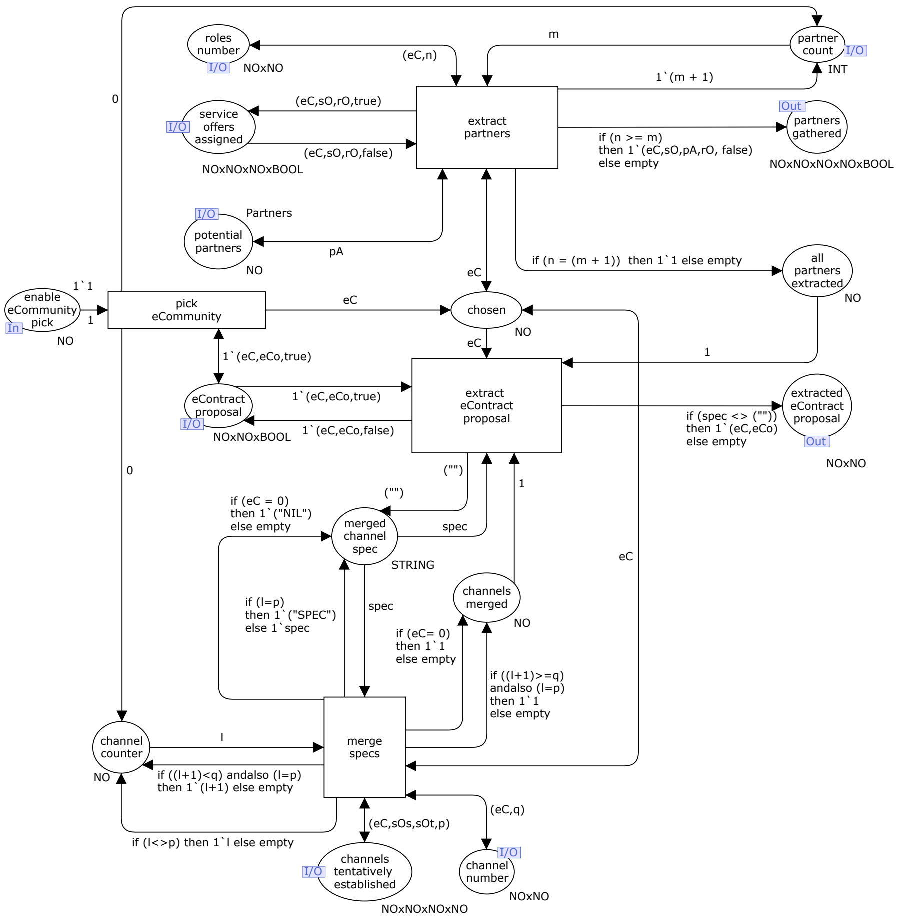


Figure 4.8: Extracting an eContract-proposal for negotiation (*contract extraction*).

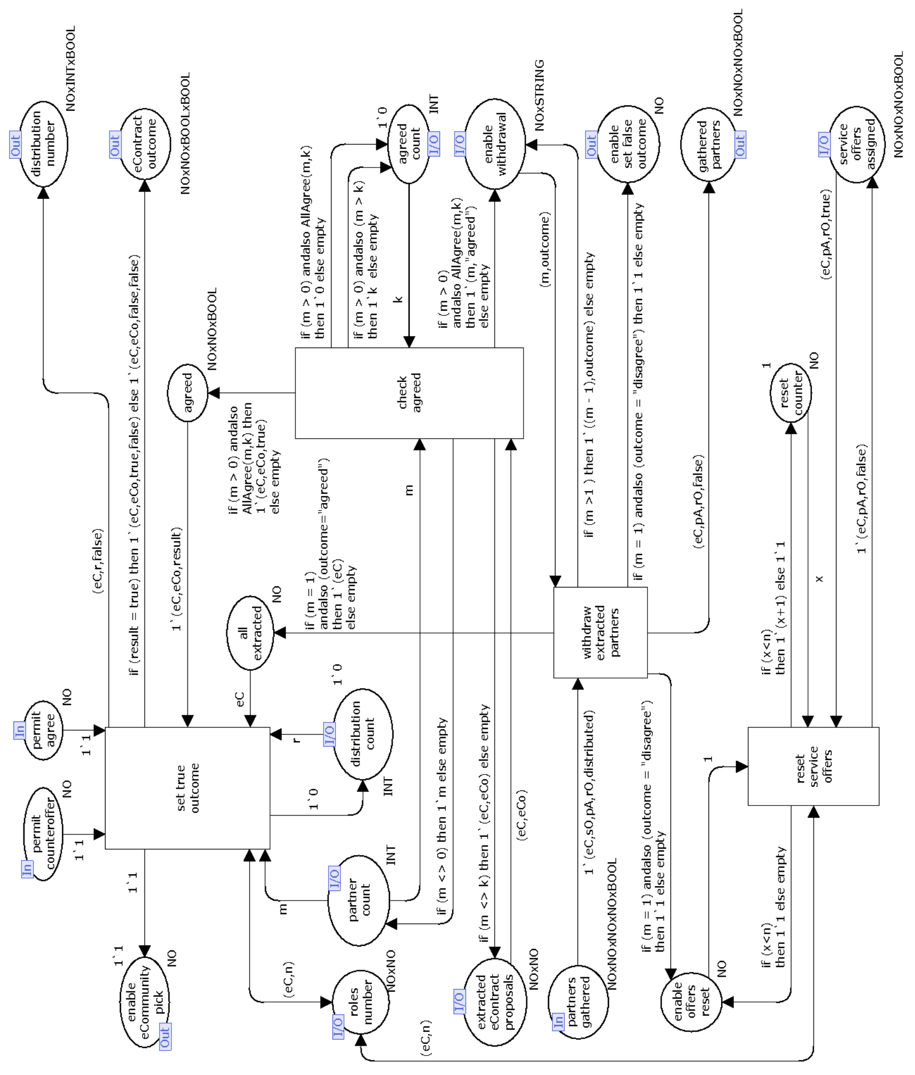

distribution number
NOxINTxBOOL
eContract outcome
NOxNOxBOOLxBOOL
agreed count
INT
enable withdrawal
NOxSTRING
enable set false outcome
NO
gathered partners
NOxNOxNOxBOOL
service offers assigned
NOxNOxNOxBOOL
agreed
NOxNOxBOOL
check agreed
reset counter
withdraw extracted partners
all extracted
set true outcome
distribution count
partner count
reset service offers
permit agree
permit counteroffer
enable eCommunity pick
roles number
NOxNO
extracted eContract proposals
partners gathered
enable offers reset
if (m > 0) andalso AllAgree(m,k) then 1`0 else empty
if (m > 0) andalso (m > k) then 1`k else empty
if (m > 0) andalso AllAgree(m,k) then 1`(m,"agreed") else empty
(m,outcome)
if (m >1) then 1`((m - 1),outcome) else empty
if (m = 1) andalso (outcome = "disagree") then 1`1 else empty
(eC,pA,rO,false)
(eC,pA,rO,true)
1`(eC,pA,rO,false)
if (x<n) then 1`(x+1) else 1`1
if (x<n) then 1`1 else empty
if (m = 1) andalso (outcome = "disagree") then 1`1 else empty
1`(eC,sO,pA,rO,distributed)
(eC,eCo)
if (m <> k) then 1`(eC,eCo) else empty
if (m <> 0) then 1`m else empty
if (m = 1) andalso (outcome="agreed") then 1`(eC) else empty
if (m > 0) andalso AllAgree(m,k) then 1`(eC,eCo,true) else empty
(eC,r,false)
if (result = true) then 1`(eC,eCo,true,false) else 1`(eC,eCo,false,false)
1`(eC,eCo,result)
(eC,n)

Figure 4.9: Finalizing an agreement on an eContract (*agreement finalizing*).

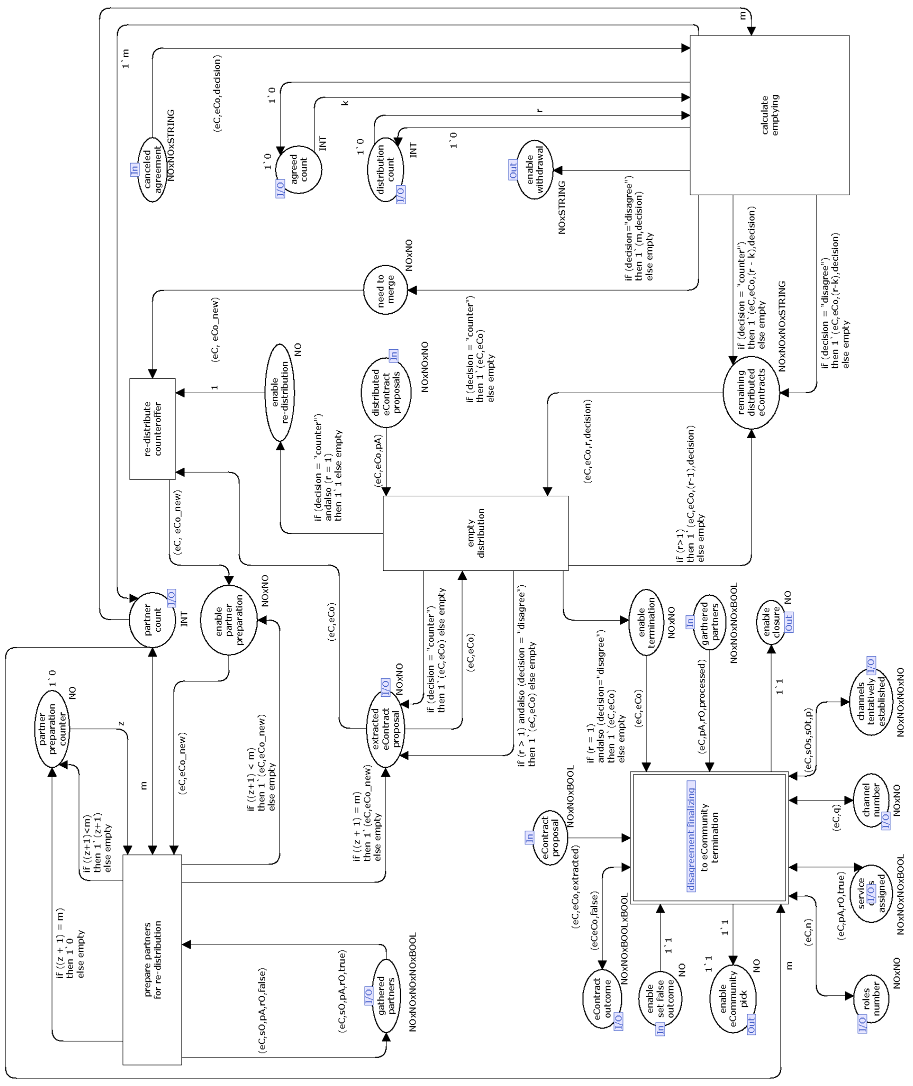


Figure 4.10: Disagreement handling for an eContract (*disagreeing*).

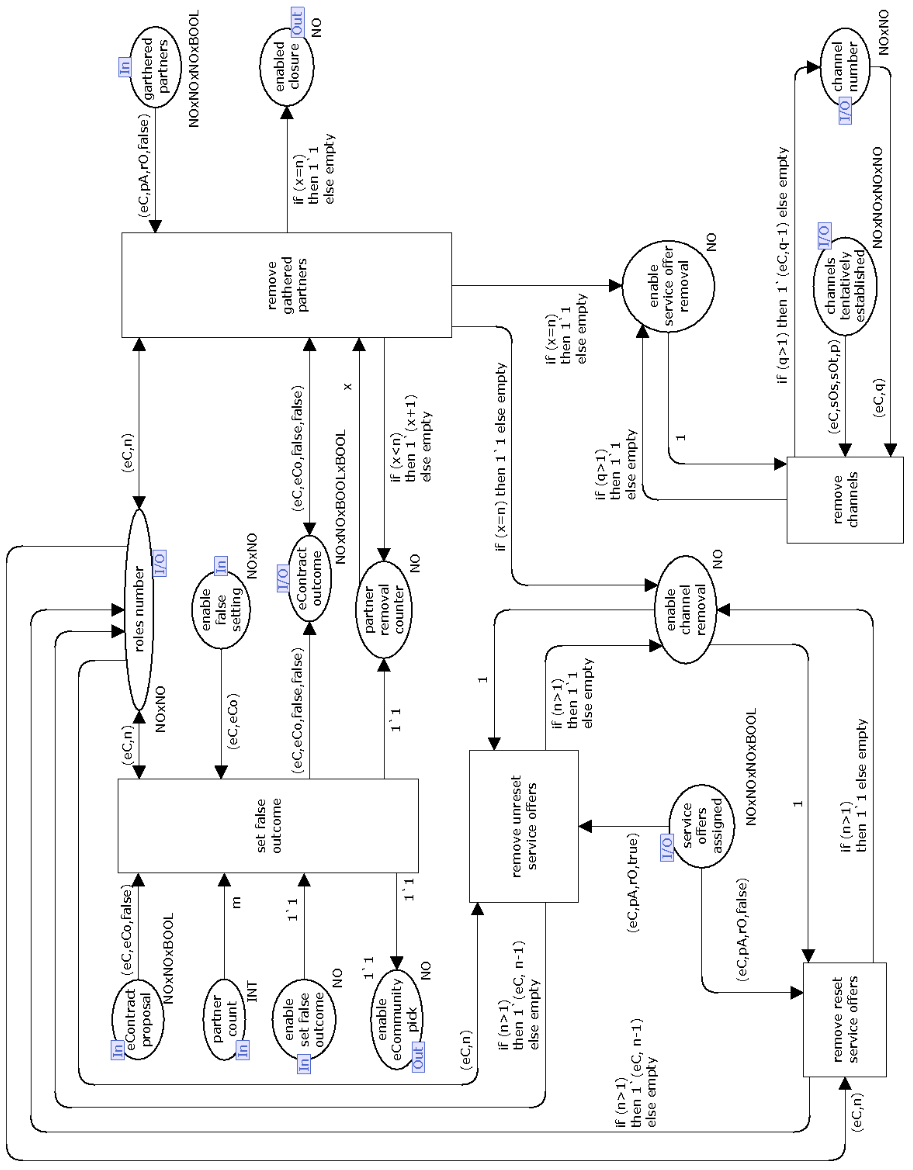


Figure 4.11: Finalizing an eContract disagreement (*disagreement finalizing*).

Chapter 5

perform - rollout

Contents

5.1 Introduction . . . 47
5.2 contract establishment . . . 49
5.2.1 governance distribution . . . 49
5.2.2 prepare . . . 49
5.3 operate . . . 50
5.3.1 enact . . . 51
5.3.2 terminate . . . 51
5.4 Conclusion . . . 52

This chapter continues the CPN-based refinement of the eBT-lifecycle presented conceptually in Chapter 3. This clarifies the semantics of not only the control-flow but also of that data-flow between services for the lifecycle-part towards the enactment of a distributed governance infrastructure. Furthermore, it shows how an eBT can be either partially or completely terminated without leaving behind abandoned processes and unneeded database entries to the point where an enacting application-system infrastructure is forced into restarting, which may inadvertently terminate countless ongoing eBTs. The structure of this chapter is as follows. In Section 5.1, an overview of the $perform$-service focuses on the embedded $rollout$. Note, while this chapter completes the main part of the lifecycle towards enactment that is a foundation for subsequent chapter about eBT-business-semantics rollbacks and compensations. Section 5.2 explains the creation of the distributed governance infrastructure (DGI) and population with concrete electronic, communicating services down to the lowest technical level. Section 5.3 shows how the DGI is enacted and either selectively or totally terminated. Section 5.4 concludes this chapter.

5.1 Introduction

Setting up the conditions for carrying out eContracts, involves first an establishment of a DGI (see Chapter 3). Briefly, every eCommunity-partner receives a local contract copy that is governed by the agreed upon eContract that latter functions as a controlling agent. Every local contract copy is the source to extract a respective set of policies, monitors, business network model agents (BNMA) for every eCommunity-partner.

Once a DGI is set up, the lowest technical level with locally enactable electronic services are machine enactable together with their service endpoints to enable communication. The business-semantics rollback and compensation options trigger exclusively from the enactment stage of an agreed upon eContract. Thus, during enactment, the scenarios may occur of a policy violation, disruptive or non-disruptive partner change. In the latter case, disruptive means there is a business-semantics rollback to the negotiation phase while non-disruptive means that the DGI remains intact and is taken over by a newly accepted eCommunity partner.

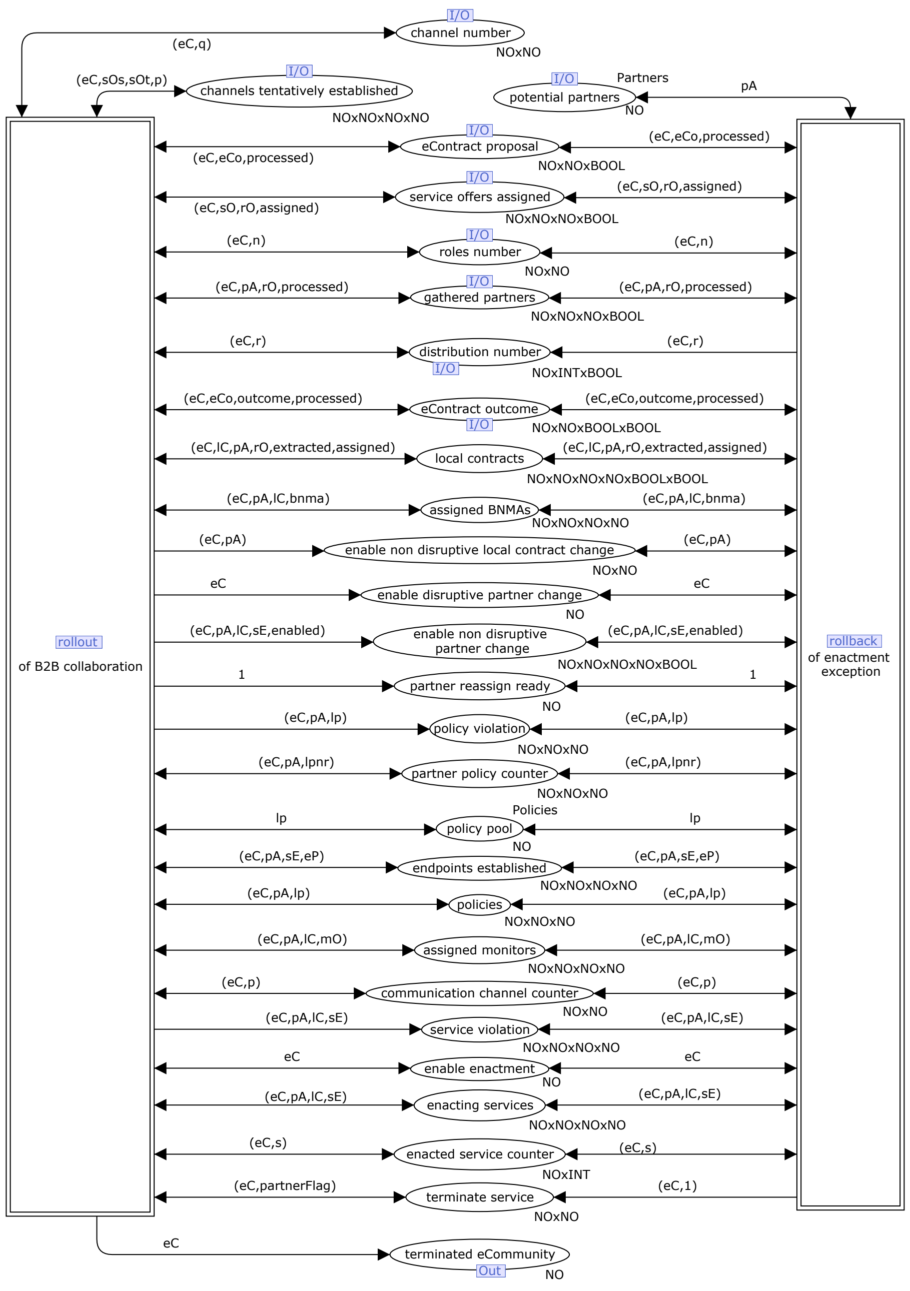


Figure 5.1: Protocol of the rollout service with business-semantics rollback (*perform*).

The contained sub-services of the $rollout$ in Figure 5.2 achieve on the one hand, governance distribution of an agreed upon eContract with an establishment of a DGI that is populated on a technical level with a set of communicating, enactable electronic services.

On the other hand, the second embedded $operate$-subservice enacts the established technical level of electronic services and terminates either non-disruptively the technical level or disruptively the entire DGI. However, since we focus in this section on setting up and enacting cross-enterprise collaboration, the business-semantics rollbacks that safeguard eBT-semantics are part of the subsequent chapter.

5.2 contract establishment

The contained subservice for *governance distribution* in Figure 5.3, starts with the arrival of an eContract all eCommunity-partners consent to. This starts the protocol population with extracted policies, monitors, BNMAs, and local contract copies for all respective eCommunity partners.

After the DGI establishment, the $prepare$ service starts the creation of a DGI-matching technical infrastructure for enactment.

5.2.1 governance distribution

The start of this service from Figure 5.4, constitutes choosing an agreed upon eContract, which first leads to the distribution of local contract copies and monitors for respective eCommunity-partners. From every local contract follows an extraction of local policies to which an eCommunity partner must adhere to.

The assigned monitors observe if policy violations occur. Every local contract has a business network model agent (BNMA) attached that utilizes the monitors to see if eCommunity-partners adhere to local policies and also policies for behavior-control of the entire eCommunity. Before the governance distribution completes, an error may be thrown if a synchronization check between the eContract and the local contracts fail.

extraction

This service creates sets of local behavior-limiting policies are extracted for every respective eCommunity-partner who is part of later DGI-enactment. Thus, the *extraction*-service of Figure 5.5 shows first the choosing of a local contract before enabling the policy extraction for respective eCommunity-partners.

5.2.2 prepare

The technical realization of a DGI involves the preparation of concrete local electronic services that fill the respective service offers. The service catches and reports exceptions during preparation and service assignment. For each electronic service, the *prepare*-service depicted in Figure 5.6 shows the establishment of communication endpoints precede a final check for the operationality of services.

During the preparation of an established DGI, errors may occur that the subservice named *preparation error* of Figure 5.7 catches. Error options are for the technical preparation of an electronic service, a failure of lifeness check and an assignment error between an electronic service and a local contract copy. Further refinements of Figure 5.7 are future work.

5.3 operate

The two sub-services of *operate* in Figure 5.8 comprise specific classes of respective input-and output places as a protocol. An enactment-ready set up DGI is the prerequisite for the *enact*-service to commence, which is given with corresponding tokens populating states as listed in Table 5.1.

With the DGI in place, a token arriving in the state *enable enactment* is a trigger for the actual enactment during which exceptional situations may occur that require business-semantics rollback. Additionally, an exception may occur during the enactment start of an electronic service. Since this is a technical issue that does not require complex business-semantics rollbacks, we omit designing exception management in detail.

		services	
purpose	state	enact	terminate
rollback enabling	enable non disruptive local contract change	x	
	enable non disruptive partner change	x	
	enable disruptive partner change	x	
	policy violation	x	
	service violation	x	
exception catching	service start error	x	
technical-level enactment of eContract and its termination	enable enactment	x	
	assigned BNMAs	x	x
	endpoints established	x	x
	enacted service counter	x	x
	local contracts	x	x
	enacting services	x	x
	policies	x	x
	assigned monitors	x	x
	terminate service	x	x
termination of distributed governance infrastructure	service offers assigned		x
	roles number		x
	gathered partners		x
	communication channel counter		x
	eContract proposal		x
	contract partner counter		x
	channels tentatively established		x
	channel number		x
	partner policy counter		x
	partner reassign ready		x
	eContract outcome		x
	terminated eCommunity		x

Table 5.1: Categorization of operate protocol.

The protocol from the *operate*-service in Table 5.1 at the very left column shows categorizations for business-semantics rollback-enabling states and for low-level exception catching that are exclusively triggered by the output nodes of the *enact*-service. Additionally, Table 5.1 shows an overlapping protocol category for the technical-level eContract enactment that is removable by the *terminate*-service. The bottom set of protocol states in Table 5.1 are for the deletion of a DGI and are equally states exclusively connected to the *terminate*-service.

5.3.1 enact

The electronic services that fill the service-offer templates in the eContract on the technical level, gather in the central state *enacting services* of the service in Figure 5.9. We assume, these services are discrete business-process specifications with a unique start state and tasks relating to each other in sequences or parallelisms that lead to a unique end state [48].

To perpetually enact respective services requires the involvement of the related BNMAs, monitors and policies. Respective services may stop for a period of time and restart again for enactment. Unless an orderly enactment culminates in a regular termination, eCommunity behavior that violates a policy results in triggering a corresponding violation assessment and business-semantics rollback that we explain in detail in the following chapter. Briefly, three additional business-semantics rollback scenarios exist that may either be disruptive or non-disruptive. In the first case, a business-semantics rollback to the eContract $negotiate$-phase eliminates the existing DGI, which is not occurring in the latter case where merely a modification in the existing DGI takes place.

5.3.2 terminate

The service unravels an eContract-enactment in three stages. As Table 5.1 shows, the $terminate$-service in Figure 5.10 not only eliminates the technical-level setup of electronic contracts but also the remainder of the distributed governance infrastructure. Correspondingly, first, the $terminate$-service removes the extracted policy sets that govern the behavior of all respective eCommunity-partners and also the enacted services that realize every local contract.

The second step of termination removes the observance part of the DGI, which involves BNMAs and monitors. Additionally, $terminate$ removes all local copies of eContracts earlier disseminated to respective eCommunity-partners together with their established communication endpoints. Finally, the removal of all channels and eCommunity-roles, completes the eContract termination.

policy service removal

This service depicted in Figure 5.11, commences with the termination of all electronic services from the technical level in enactment. Next, the counter consumption with the amount of eCommunity-partners triggers the removing of all policies that are prior extracted from the eContract.

observance removal

The elements for observing the enactment of an eContract are the monitors and BNMAs. As depicted in Figure 5.12, these infrastructure elements for observing the eCommunity-partner behavior are removed as part of the DGI together with related local contracts. Finally, it is possible to remove the communication channels represented by the established endpoints.

roles removal

This service depicted in Figure 5.13, first removes the roles that are part of the agreed upon eContract and all eCommunity-partners that populate these roles. Finally, the *roles removal* service consumes all tentatively established channels that are realized by communication endpoints on the technical level of electronic services.

5.4 Conclusion

The main eBT-lifecycle part of this chapter completes to the point of enacting the established distributed governance infrastructure. The enactment of the DGI may be either totally or partially terminated. In the first case, the entire DGI is stepwise dismantled and the lifecycle of the transacting eCommunity brought to its final logging state. The latter case performs a partial DGI dismantling but rolls the eCommunity identification back to the negotiation phase in which the proto-contract is repeated, during which the partnership of eCommunity partners may change that populate concrete roles. This lifecycle towards enactment we explain with respect to semantically deterministic control- and data flow. Besides inserted low-level exception handling, the enactment stage may trigger either disruptive or non-disruptive business-semantics rollback- and compensation steps for policy violations and partner changes. Disruptive means the semantic business-semantics rollback and compensation dismantles the distributed governance infrastructure while non-disruptive means merely changes take place in an existing distributed governance infrastructure where merely an eCommunity-partner replacement takes place. the

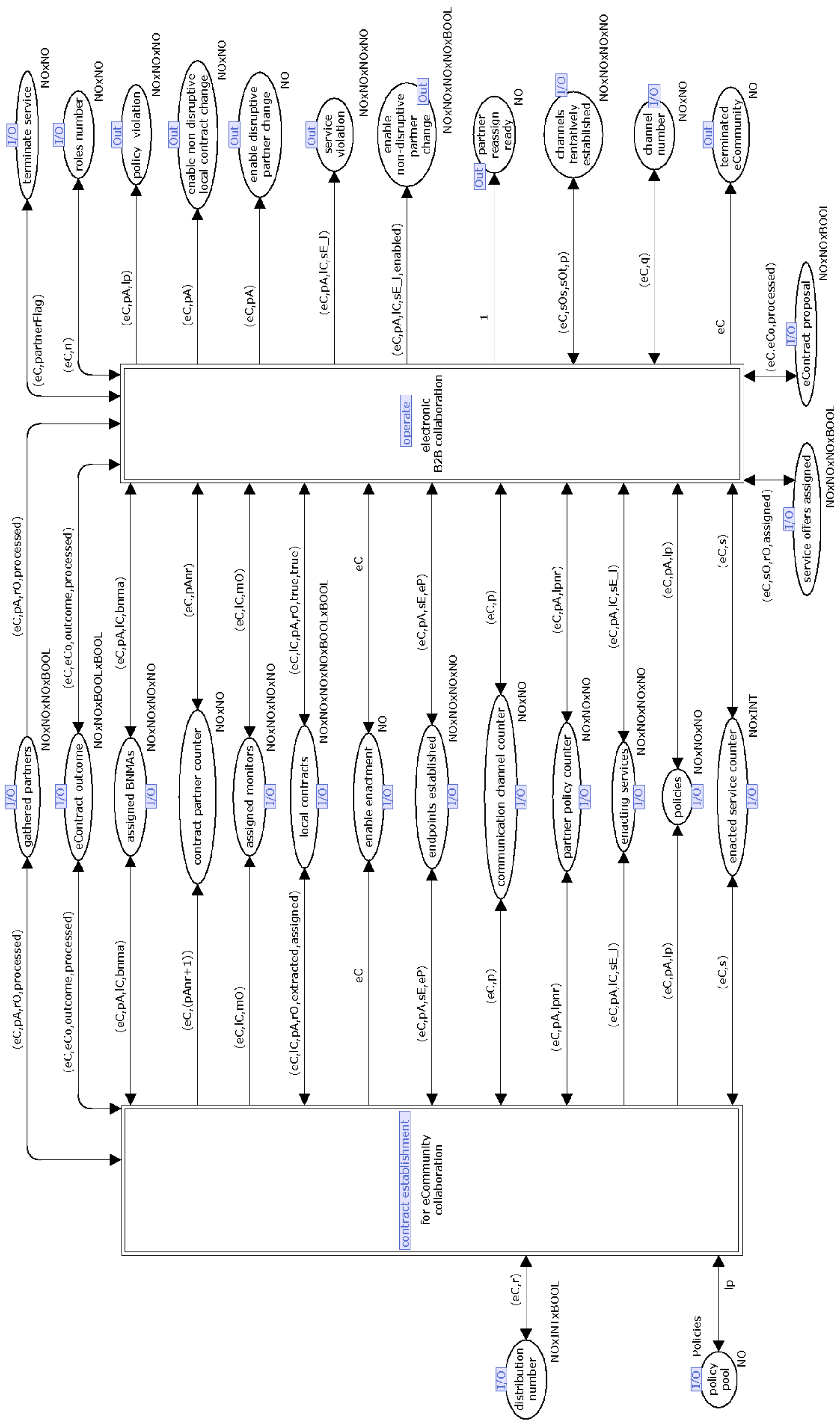


Figure 5.2: Establishing a DGI and operating it (*rollout*).

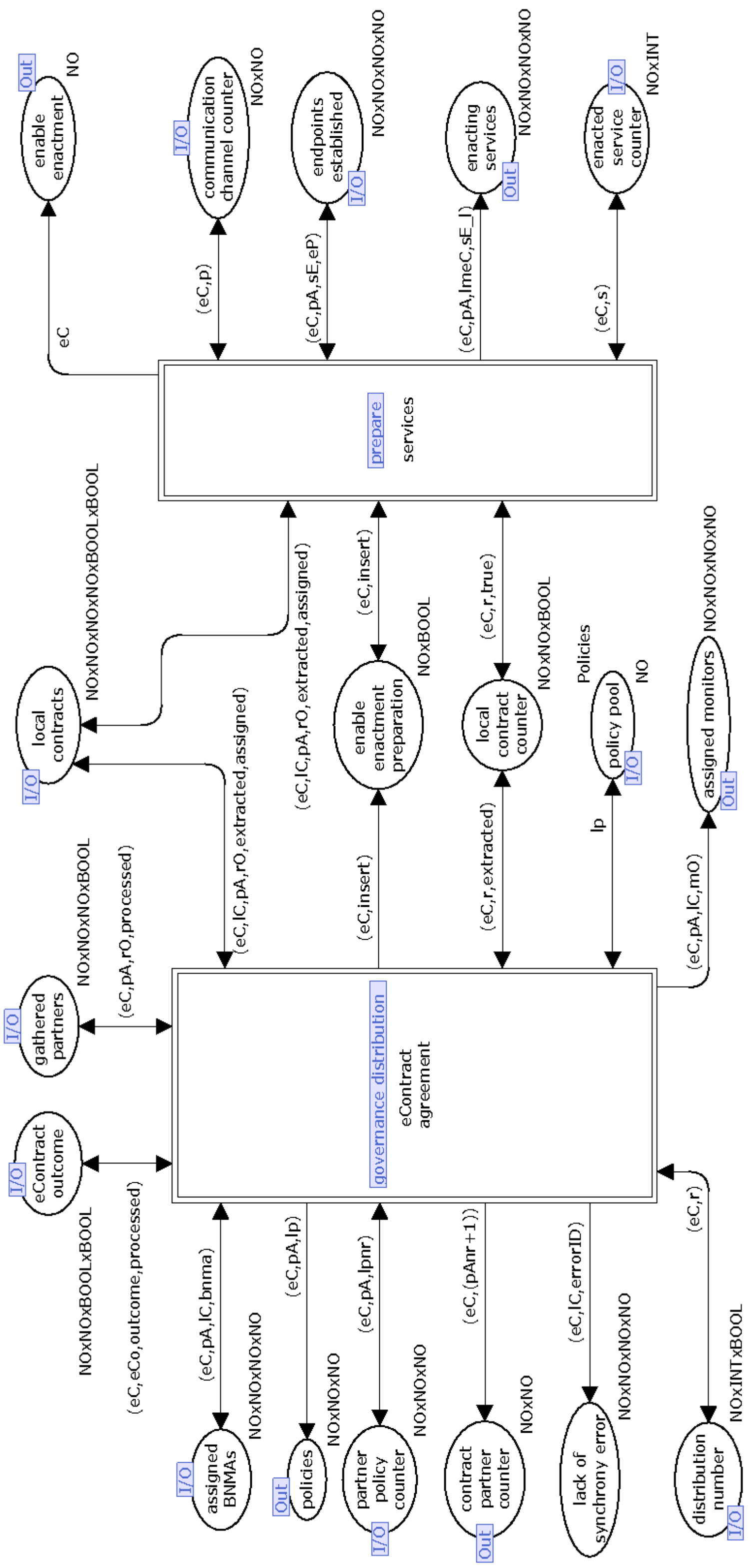


Figure 5.3: Distributing local contracts, policies and populating with electronic services (*contract establishment*).

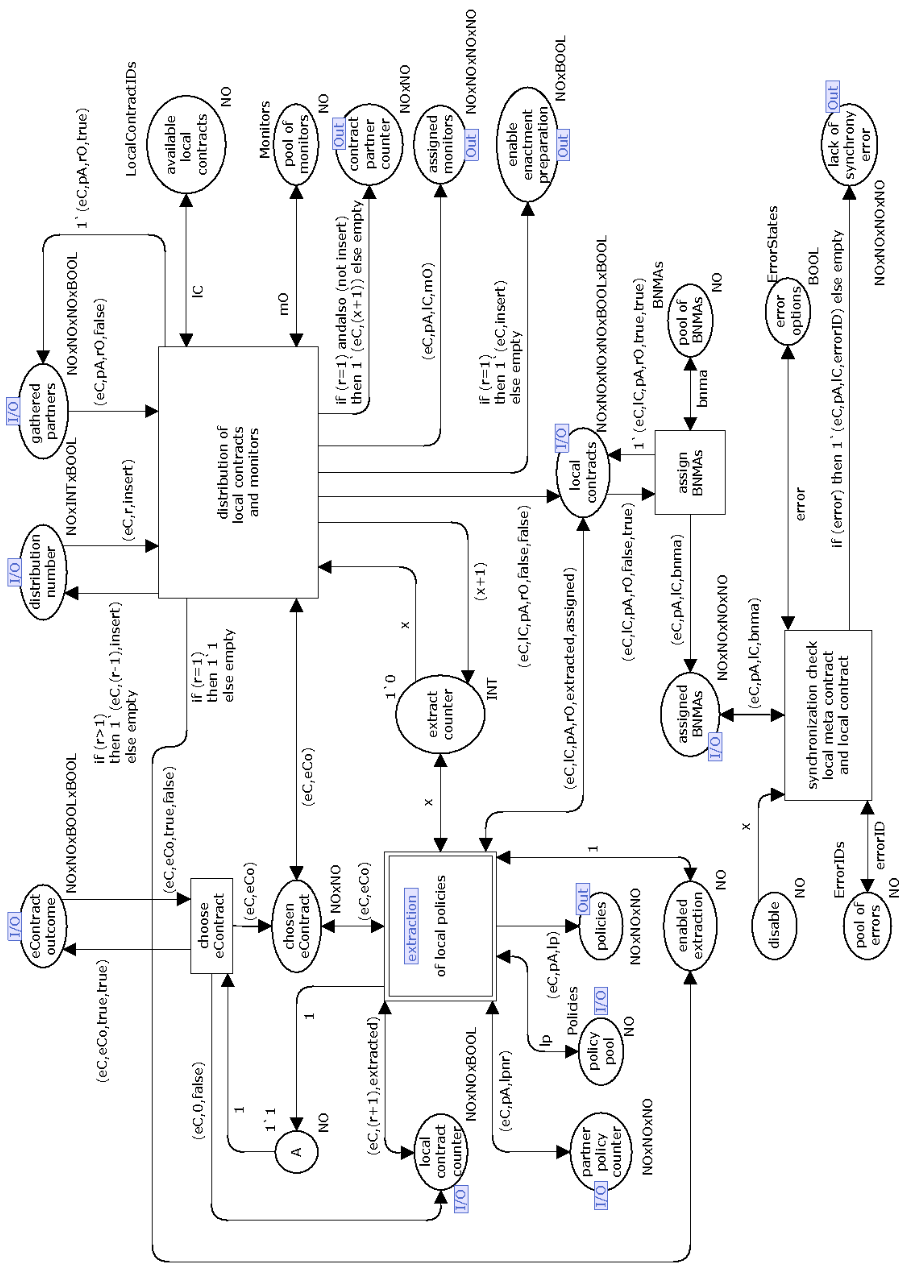


Figure 5.4: Policy extraction from local contract copies and assignment of monitors and BNMAs (*governance distribution*).

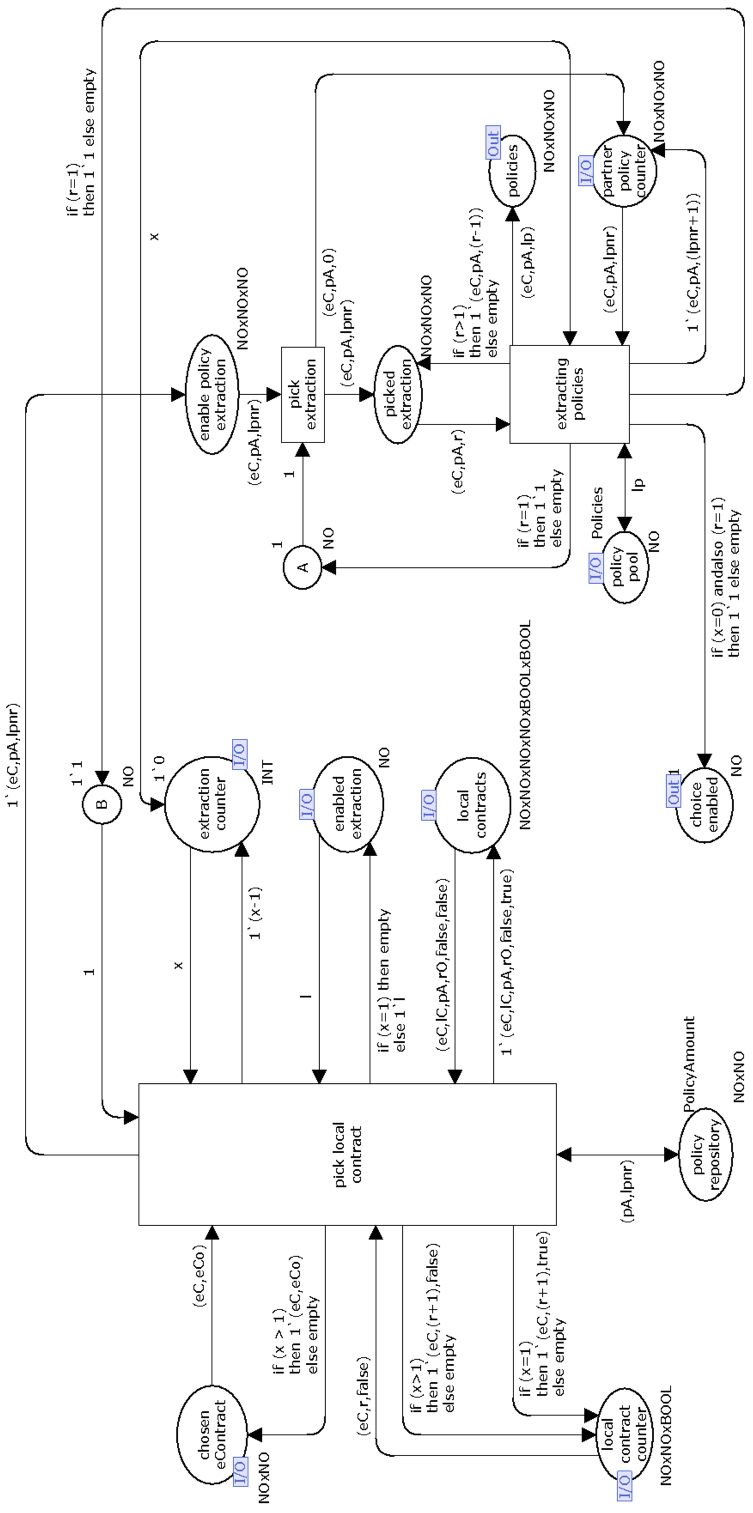


Figure 5.5: Extracting from local contract copies the amount of required policies (*extraction*).

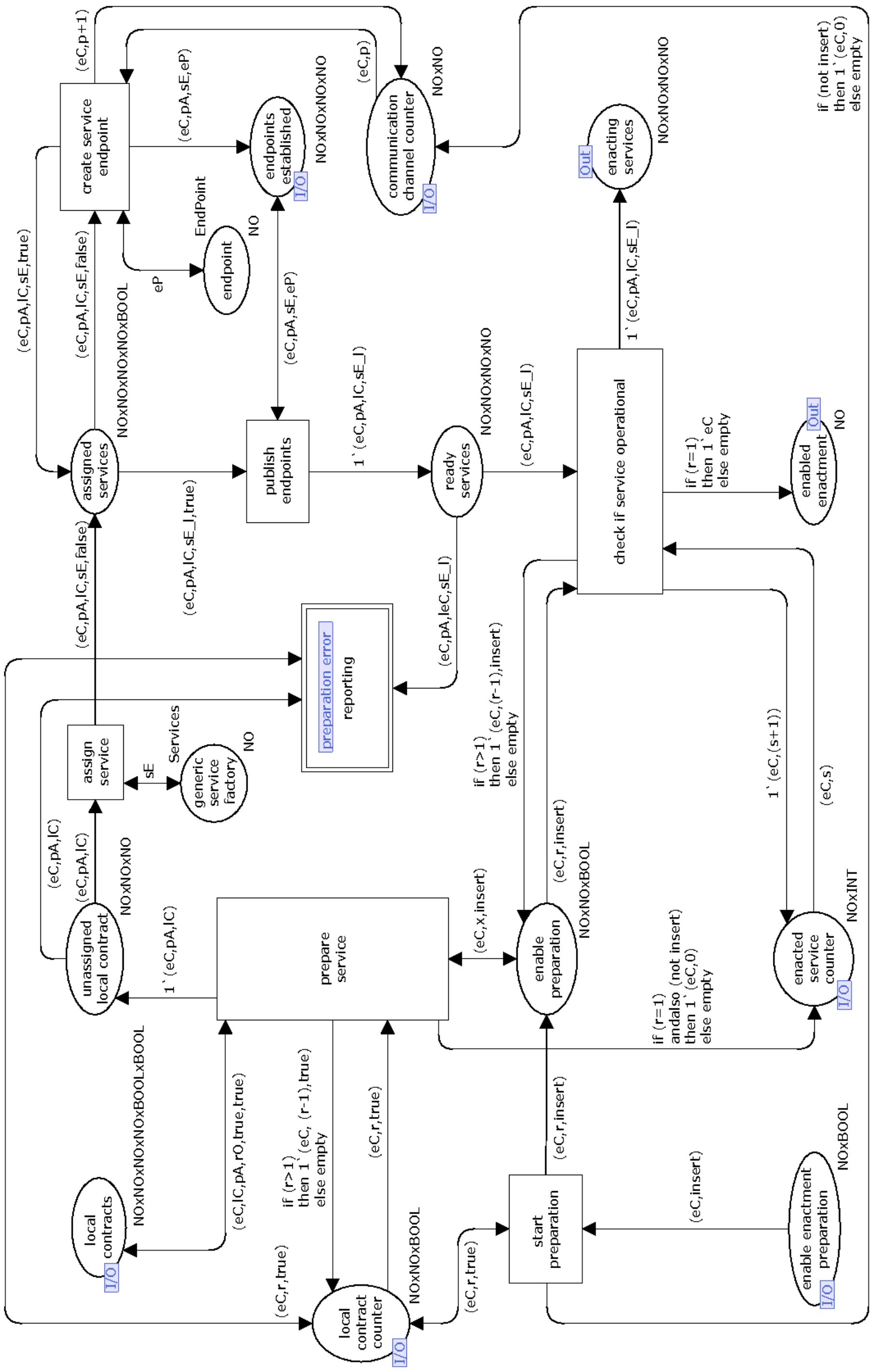


Figure 5.6: Electronic service choosing and communication endpoint creation (*prepare*).

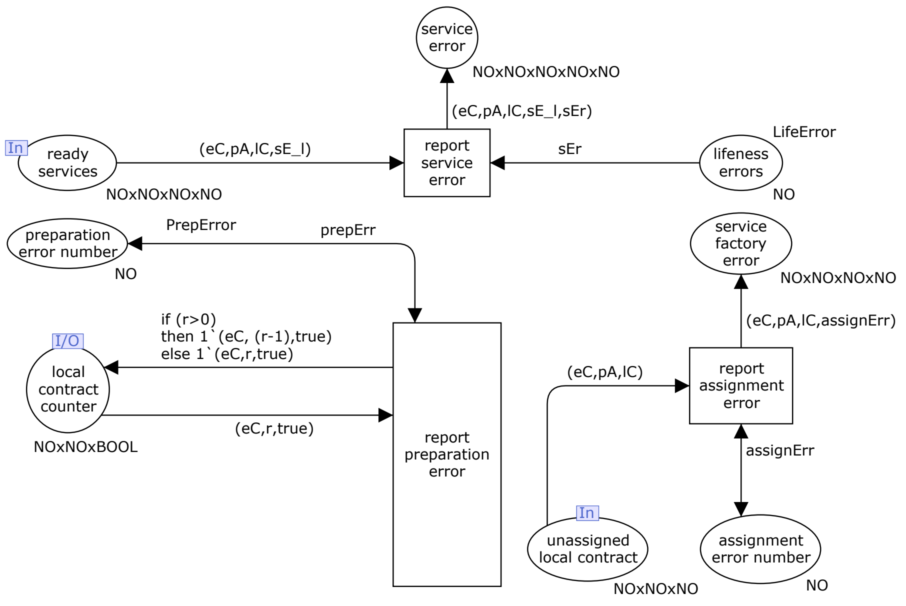


Figure 5.7: Exception handling for service preparation (*preparation error*).

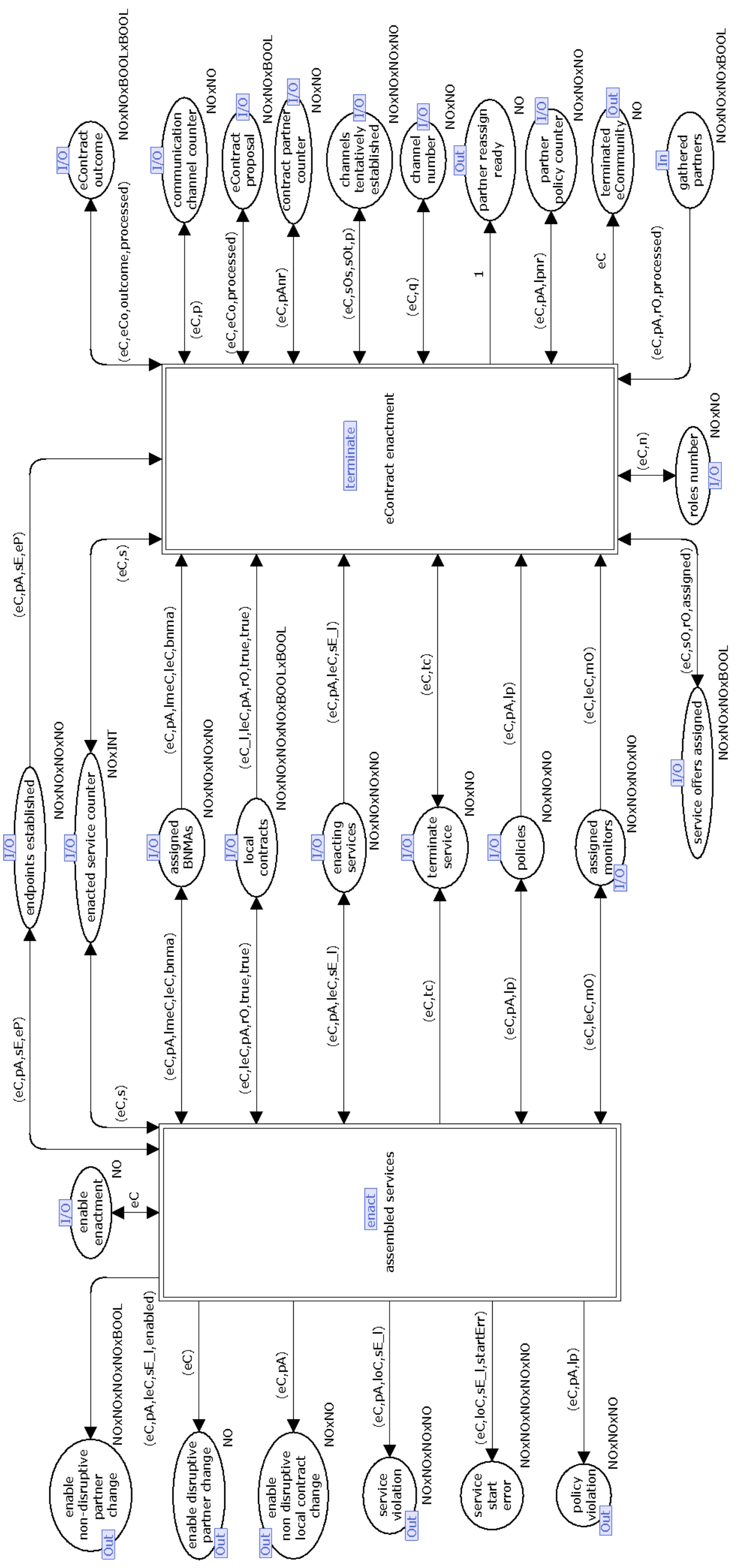


Figure 5.8: Protocol for enacting and terminating an eContract (*operate*).

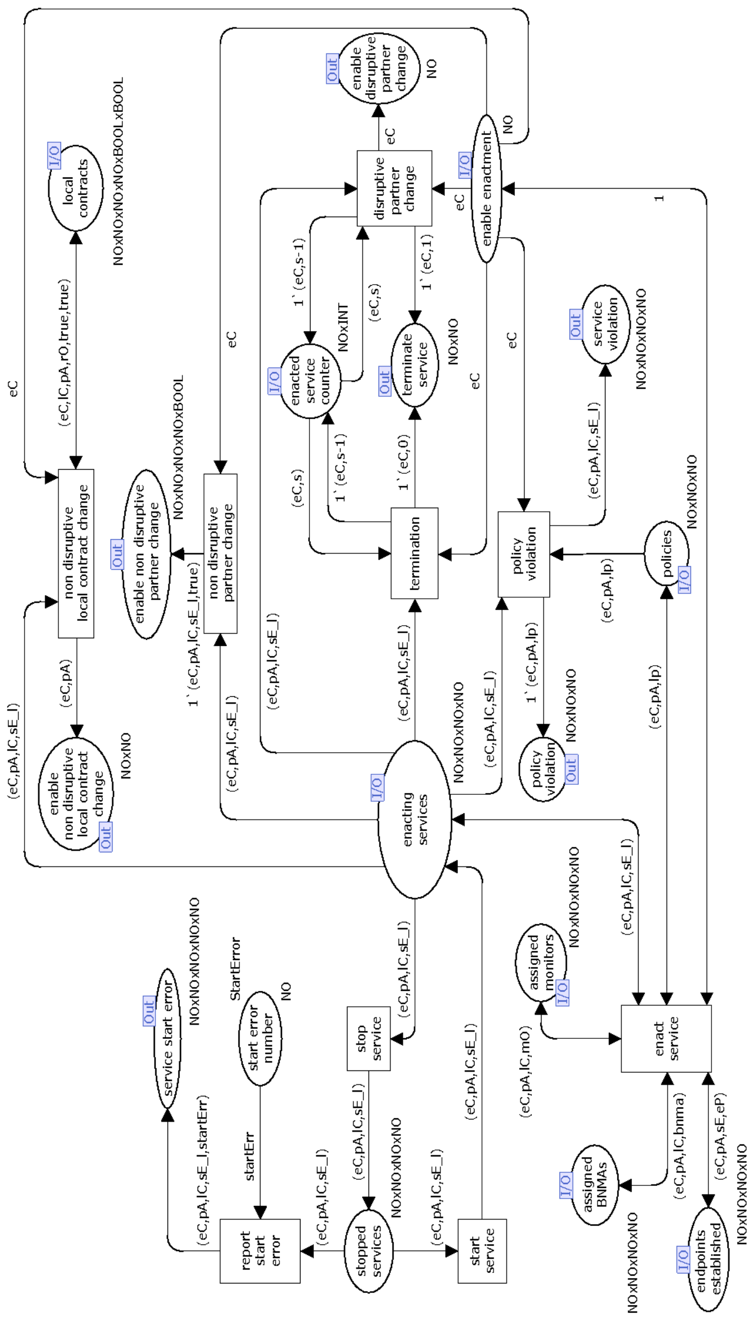


Figure 5.9: Enacting an eContract and triggering business-semantics rollbacks or a termination (*enact*).

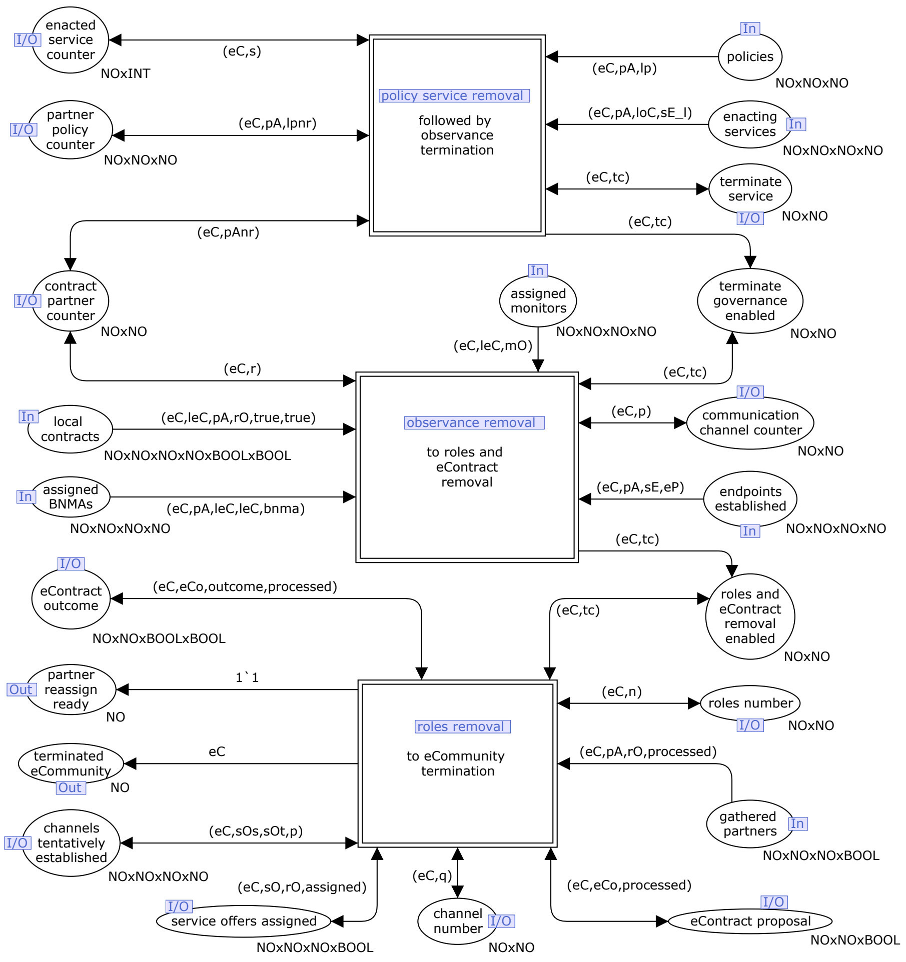


Figure 5.10: Terminating an eCommunity (*terminate*).

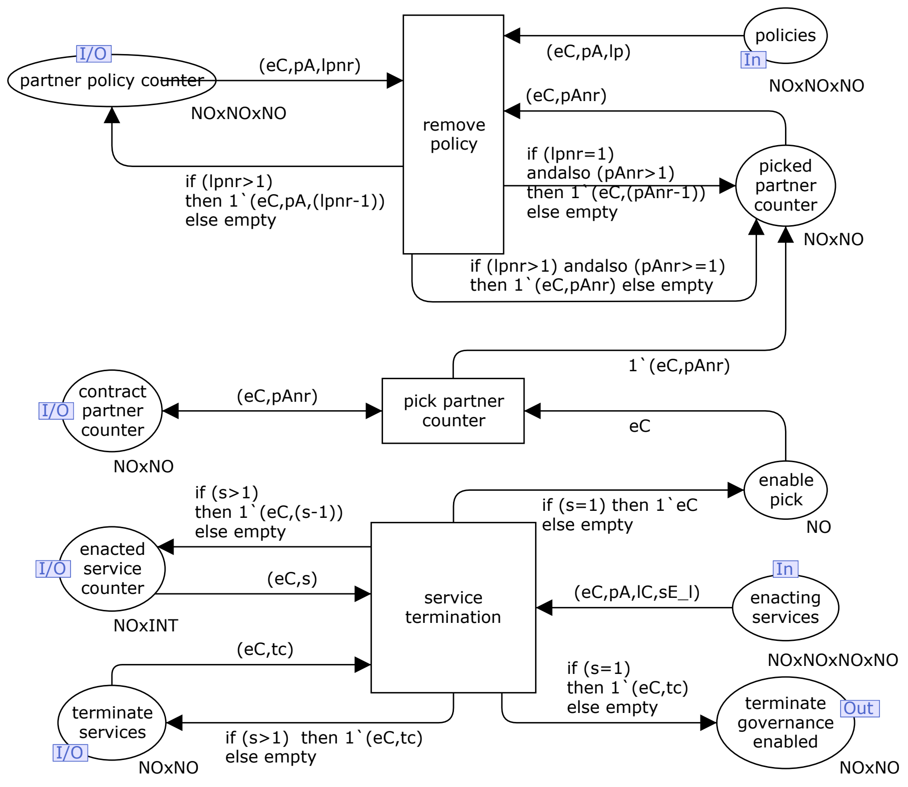


Figure 5.11: Removing policies and electronic services that are enacted (*policy service removal*).

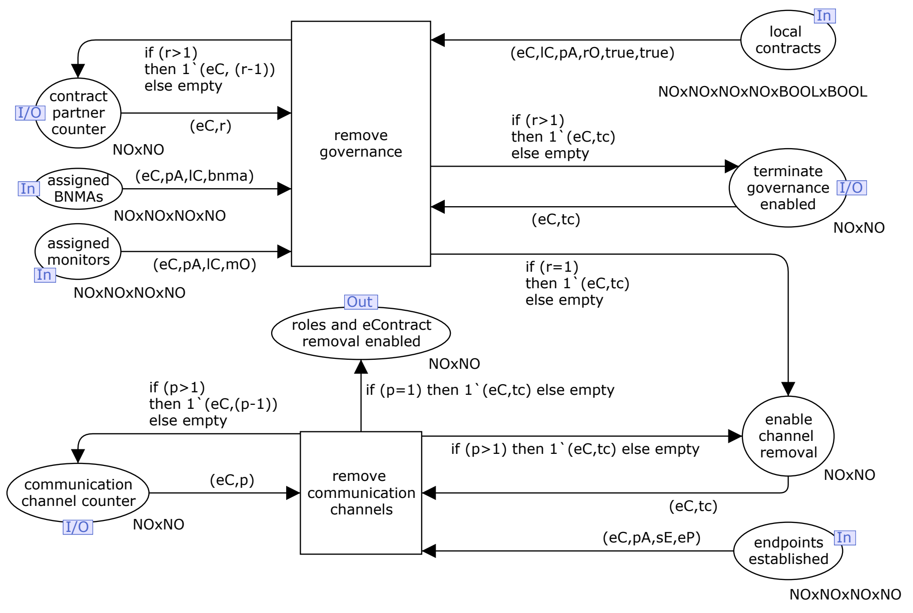

if (r>1) then 1`(eC, (r-1)) else empty
contract partner counter
I/O
NOxNO
(eC,r)
remove governance
(eC,lC,pA,rO,true,true)
local contracts
In
NOxNOxNOxNOxBOOLxBOOL
if (r>1) then 1`(eC,tc) else empty
terminate governance enabled
I/O
NOxNO
(eC,tc)
assigned BNMAs
In
(eC,pA,lC,bnma)
NOxNOxNOxNO
assigned monitors
In
(eC,pA,lC,mO)
NOxNOxNOxNO
if (r=1) then 1`(eC,tc) else empty
Out
roles and eContract removal enabled
NOxNO
if (p=1) then 1`(eC,tc) else empty
if (p>1) then 1`(eC,(p-1)) else empty
if (p>1) then 1`(eC,tc) else empty
enable channel removal
NOxNO
communication channel counter
I/O
NOxNO
(eC,p)
remove communication channels
(eC,tc)
(eC,pA,sE,eP)
endpoints established
In
NOxNOxNOxNO

Figure 5.12: Removing BNMAs, communication endpoints and local contracts (*observance removal*).

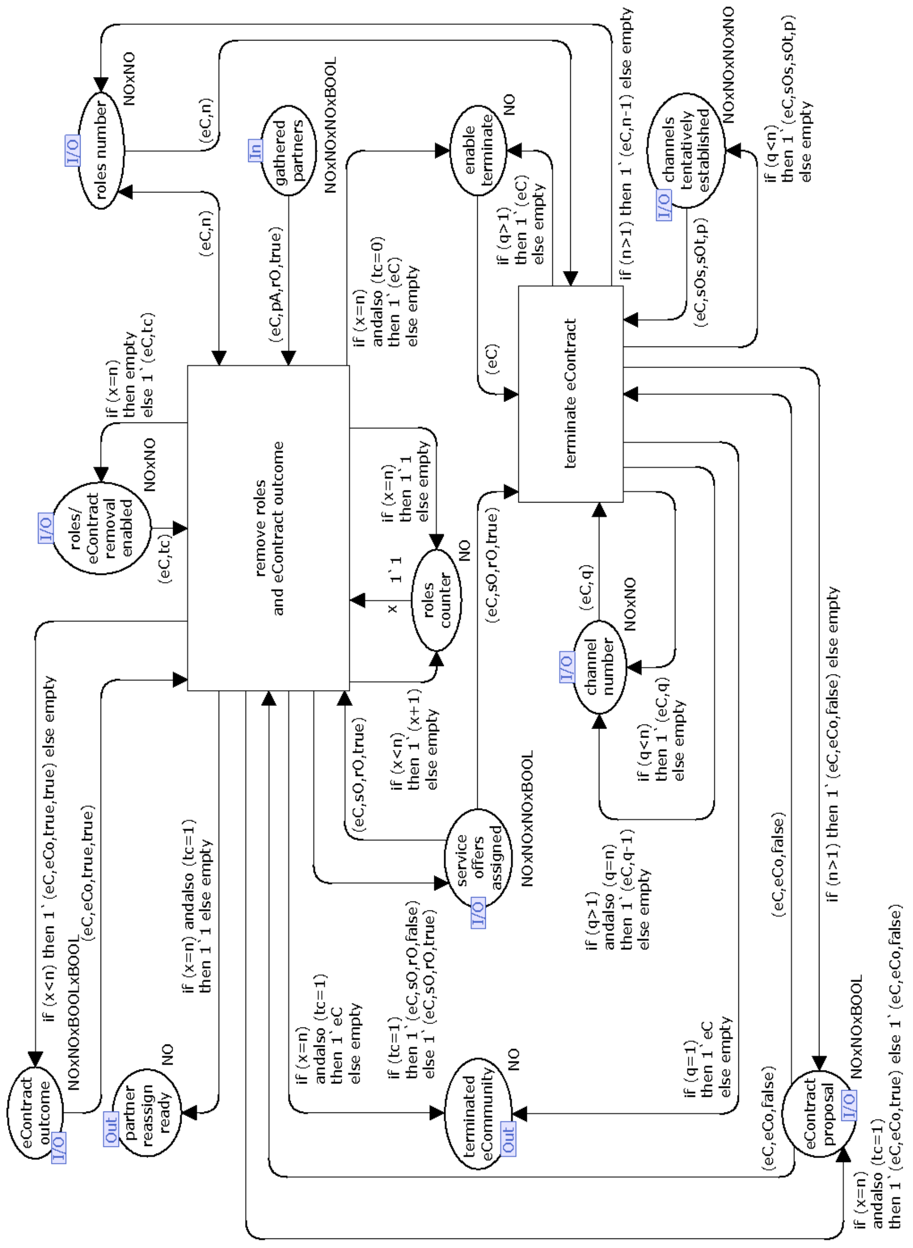

roles number
I/O
NOxNO
(eC,n)
(eC,n)
gathered partners
In
NOxNOxNOxBOOL
(eC,pA,rO,true)
enable terminate
NO
if (x=n) andalso (tc=0) then 1`(eC) else empty
if (q>1) then 1`(eC) else empty
(eC)
if (n>1) then 1`(eC,n-1) else empty
channels tentatively established
I/O
NOxNOxNOxNO
(eC,sOs,sOt,p)
if (q<n) then 1`(eC,sOs,sOt,p) else empty
terminate eContract
remove roles and eContract outcome
roles/ eContract removal enabled
I/O
NOxNO
(eC,tc)
if (x=n) then empty else 1`(eC,tc)
if (x=n) then 1`1 else empty
roles counter
NO
1`1
x
if (x<n) then 1`(x+1) else empty
(eC,sO,rO,true)
(eC,sO,rO,true)
service offers assigned
I/O
NOxNOxNOxBOOL
channel number
I/O
NOxNO
(eC,q)
if (q<n) then 1`(eC,q) else empty
if (q>1) andalso (q=n) then 1`(eC,q-1) else empty
eContract outcome
I/O
NOxNOxBOOLxBOOL
if (x<n) then 1`(eC,eCo,true,true) else empty
(eC,eCo,true,true)
partner reassign ready
Out
NO
if (x=n) andalso (tc=1) then 1`1 else empty
if (x=n) andalso (tc=1) then 1`eC else empty
if (tc=1) then 1`(eC,sO,rO,false) else 1`(eC,sO,rO,true)
terminated eCommunity
Out
NO
if (q=1) then 1`eC else empty
(eC,eCo,false)
(eC,eCo,false)
if (n>1) then 1`(eC,eCo,false) else empty
eContract proposal
I/O
NOxNOxBOOL
if (x=n) andalso (tc=1) then 1`(eC,eCo,true) else 1`(eC,eCo,false)

Figure 5.13: Finalizing an eContract disagreement (*roles removal*).

Chapter 6

perform - rollback

Contents

6.1 Introduction **65**
6.2 disruptively reset **67**
6.3 nondisruptively choose **67**
6.3.1 remove 68
6.3.2 reinsert 68
6.4 nondisruptively change **70**
6.4.1 removal 72
6.4.2 insertion 73
6.5 nondisruptively manage **73**
6.6 Conclusion **73**

Based on the previously specified collaboration, the research question is tackled how to safeguard the CEC-lifecycle with eBT that commences with the breeding ecosystem and leads to enactment and encompasses various business-semantics rollback processes that we present in this chapter. This Saga-type eBT comprises the orderly termination of a transacting eCommunity. Thus, the disruptive and non-disruptive business-semantics rollbacks and compensation ensure there is no transaction breakdown when violations occur. Section 6.2 specifies a disruptive business-semantics rollback and compensation that removes an existing DGI. Sections 6.3 till 6.5 explain non-disruptive business-semantics rollbacks and compensations that leave the DGI intact. Finally, Section 6.6 concludes this chapter.

6.1 Introduction

As an eBT comprises several layers (see Figure 2.3) of which one is for cross-enterprise collaboration and several are nested enterprise internal, we specify the semantics of the top-level eBT with its features in this chapter. As the actual eBT commences with the negotiation (see Figure 5.1) of a proto-contract, the business-semantics rollback option has its furthest business-semantics rollback point from the enactment stage into this eBT-lifecycle stage. If a business-semantics rollback returns an eContract enactment back to the $negotiate$ phase, it is disruptive as it involves the dismantling of the DGI.

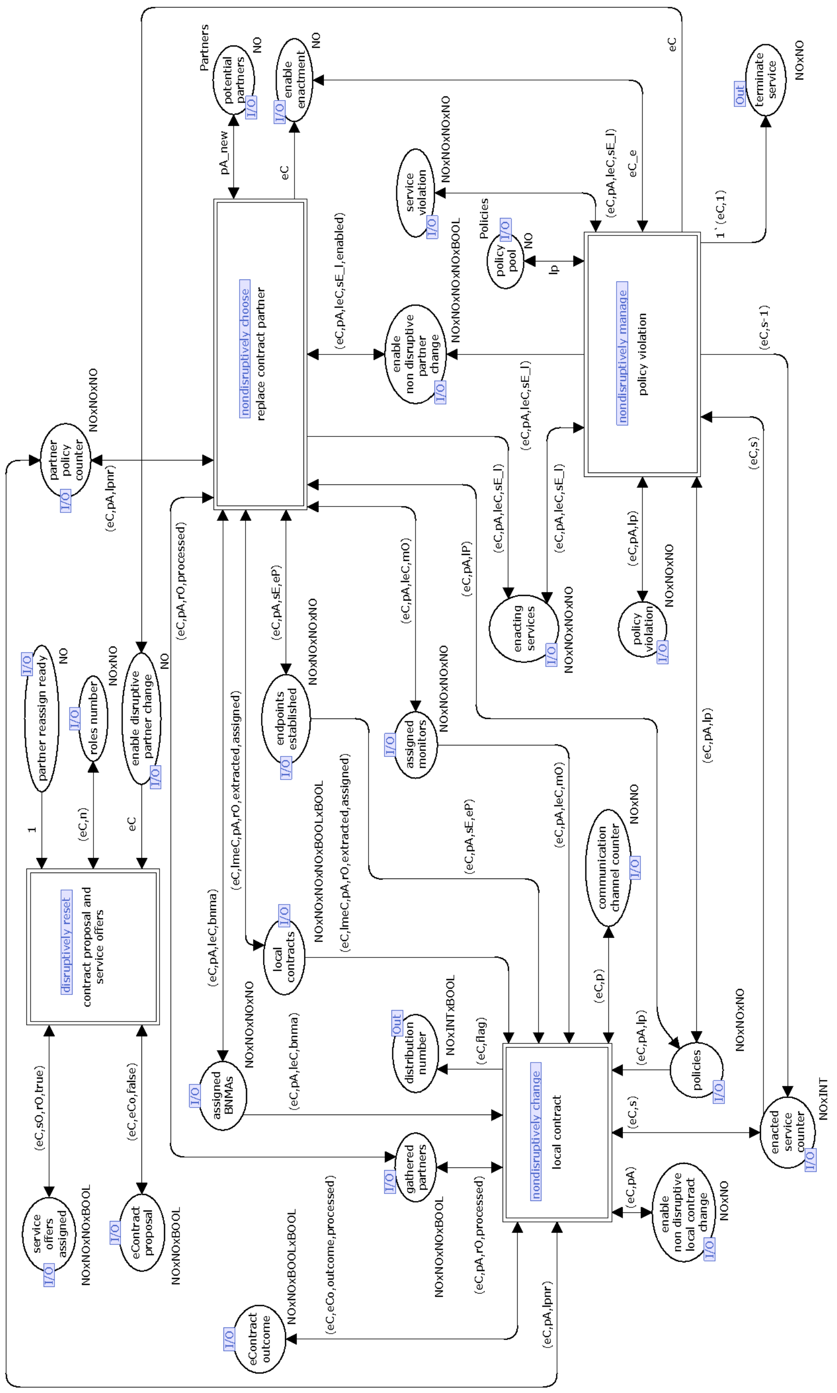


Figure 6.1: Options for rolling back violations in an eContract enactment (*business-semantics rollback*).

The other business-semantics rollback processes leave the DGI intact and are therefore non-disruptive.

The top-level eBT resembles a Saga [23] transaction that is an idea adopted from chained transactions of including a compensation mechanism to roll back. Sagas divide a long lasting transaction into sequentially executed atomic sub-transactions with ACID properties and each sub-transaction, except the last one, has its own compensating sub-transaction. When any failure arises, the committed sub-transactions are undone by compensating sub-transactions. Unlike chained transactions, Sagas can return the whole transaction back to the very beginning with compensations. Note that failures refer to traditional database settings. However, in the eBT-lifecycle in this manuscript, it is behavior-controlling policy violations that result in the undoing by sub-transactions.

The service in Figure 6.2 shows one disruptive and three non-disruptive business-semantics rollbacks. The first type of disruptive business-semantics rollback (see Section 6.2) commences after the decision to replace a current eCommunity partner with the objective to set up a new DGI. Thus, the disruptive business-semantics rollback dismantles the existing DGI and rolls back to the $negotiation$-service. Such a business-semantics rollback implies it is possible to have multiple eCommunity-partners choose to discontinue their involvement in a newly emerging eContract. Note that the policy-violating partner may again be part of the new eContract. We infer, the reason for a disruptive partner change is caused by a policy violation of a severity that does not permit an eCommunity to continue collaboration.

Of the remaining three non-disruptive business-semantics rollbacks, one (see Section 6.3) also replaces an eCommunity-partner, however, in a way where it does not dismantle and recreate the DGI. Likewise, the other non-disruptive business-semantics rollbacks equally leave the existing DGI intact. Secondly, a policy violation causes the second non-disruptive business-semantics rollback (see Section 6.5) that an eCommunity does not consider severe enough to justify a dismantling and re-creation of a DGI. If the eCommunity-partners vote to ignore a respective policy violation, the related eContract enactment resumes without any modification. The third type of non-disruptive business-semantics rollback (see Section 6.4) allows the complete replacement of a local contract with a new one as part of the existing DGI that remains otherwise unchanged. Thus, the new local contract and related policies, BNMA and monitor must adhere to the DGI-coordinating eContract agent.

6.2 disruptively reset

The delivery of the identification number of the eCommunity from the $enact$-service in Figure 5.9 by a corresponding transition labeled *disruptive partner change*, triggers this service depicted in Figure 6.2. The same transition also triggers a partial termination and removal of the DGI. When the partial termination completes, an enabling token enters the input node labeled *partner reassign ready*, which triggers the reaction of eContract negotiation by the correspondingly named service in Figure 6.1.

Before the negotiation starts, it is necessary to reset the service offers for extracting contract proposals. Since every eCommunity-partner has a service offer to fill, the amount of resets equally results from the partial DGI-termination.

6.3 nondisruptively choose

This service of Figure 6.3 consists of two parts. On the one hand, a part for removing temporarily all elements related to a local contract, i.e., policies, BNMA, endpoint,

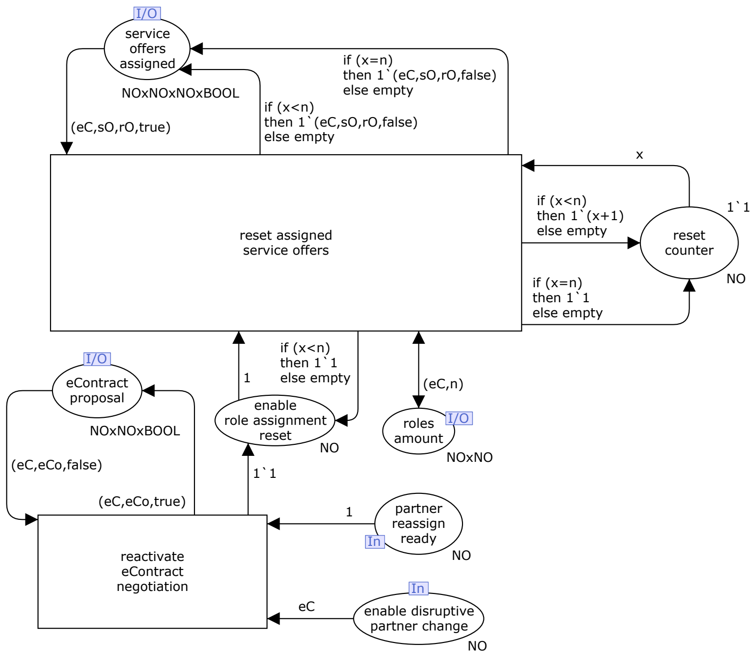

I/O
service offers assigned
NOxNOxNOxBOOL
(eC,sO,rO,true)
if (x<n) then 1`(eC,sO,rO,false) else empty
if (x=n) then 1`(eC,sO,rO,false) else empty
x
reset assigned service offers
if (x<n) then 1`(x+1) else empty
1`1
reset counter
NO
if (x=n) then 1`1 else empty
I/O
eContract proposal
NOxNOxBOOL
(eC,eCo,false)
(eC,eCo,true)
1
if (x<n) then 1`1 else empty
enable role assignment reset
NO
(eC,n)
roles amount
I/O
NOxNO
1`1
1
partner reassign ready
In
NO
reactivate eContract negotiation
eC
In
enable disruptive partner change
NO

Figure 6.2: Establishing a distributed governance infrastructure and operating it (*disruptively reset*).

monitor. The reason for this temporary local removal is the identification key requiring change for all these related elements that comprise a composite of the unique number of the eCommunity and contained partner. Thus, besides inserting a new eCommunity-partner itself, a replication happens in the composite identification keys of the related local contracting elements. The second nested service in Figure 6.3 performs this reinsertion with the replacing new eCommunity-partner into the existing DGI and enables the overall eContract enactment again after the replacement process completes.

6.3.1 remove

For locally replacing an eCommunity-partner in Figure 6.4, a token enters in the state labeled *enable local partner change* comprising the identifiers of the eCommunity, old partner to be replaced, local contract and local service. The arrival of that token commences a stepwise, temporary removal of a uniquely identified local contract and related elements starting with the respective BNMA.

The old partner is taken out of the eCommunity under concern, followed by the local contract taken aside. The same happens to related monitors and endpoints, the policies and their related number counter. The latter temporarily removes the complete set of policies of the old eCommunity-partner from the $enact$-service. Note, the properties of the color tuple in the enabling token contains all required information for this stepwise, temporary removal.

6.3.2 reinsert

When the $remove$-service completes and adds a token into the state labeled *enable reinsertion*, the service in Figure 6.5 commences with reinserting into the existing DGI

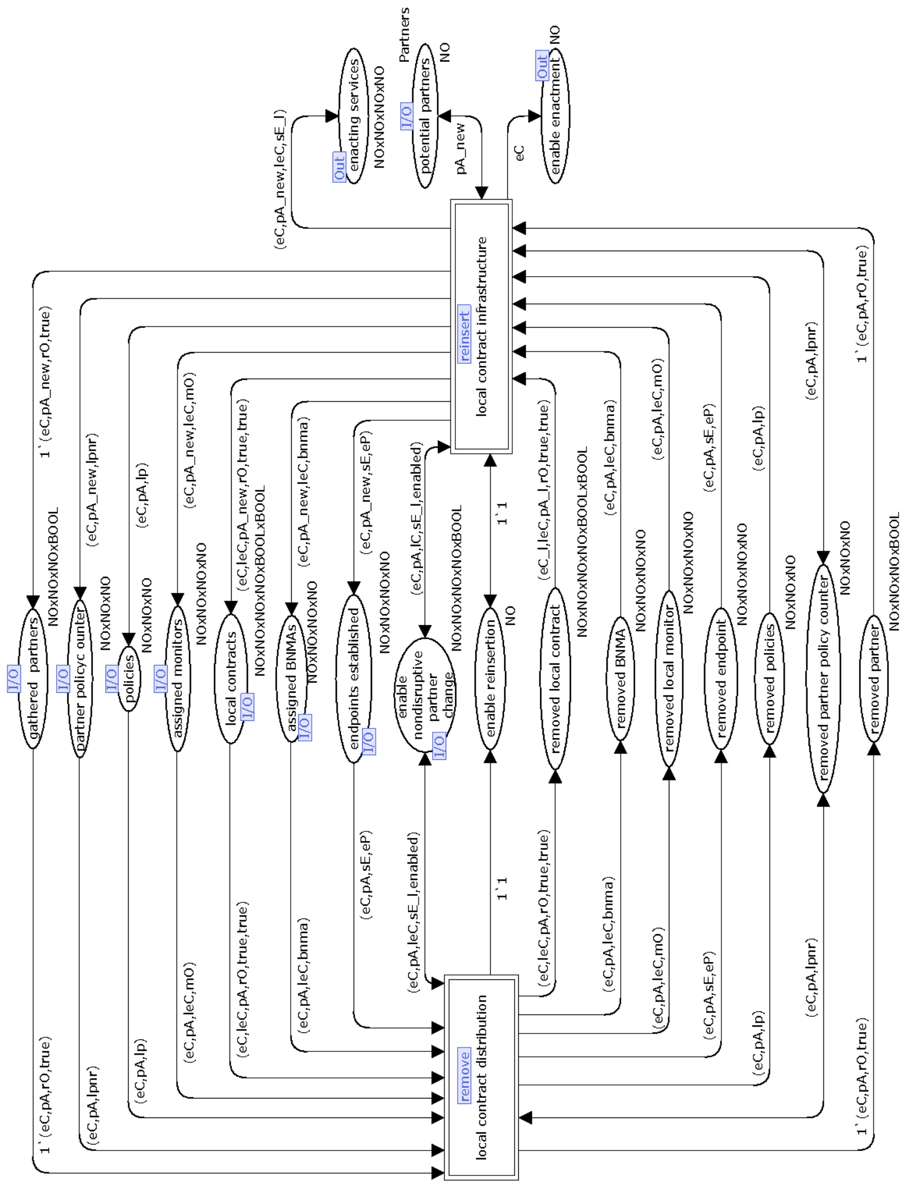


Figure 6.3: Distributing local contracts, policies and populating with local electronic services (*nondisruptively choose*).

the local contract and related elements. To enable reinserting a local contract and related other DGI-elements, a new partner is chosen based on the acceptance of the remaining eCommunity to be a new replacing partner, which is future refinement work and out of scope for now.

The chosen new eCommunity-partner is the prerequisite for reconstructing the complete DGI that is missing one new local contract with related elements after a $remove$-performance. Thus, the new partner identification becomes part of the otherwise unchanged, composed identification key for the reinserted local contract and associated endpoint, BNMA, monitor, policies and related amount counter.

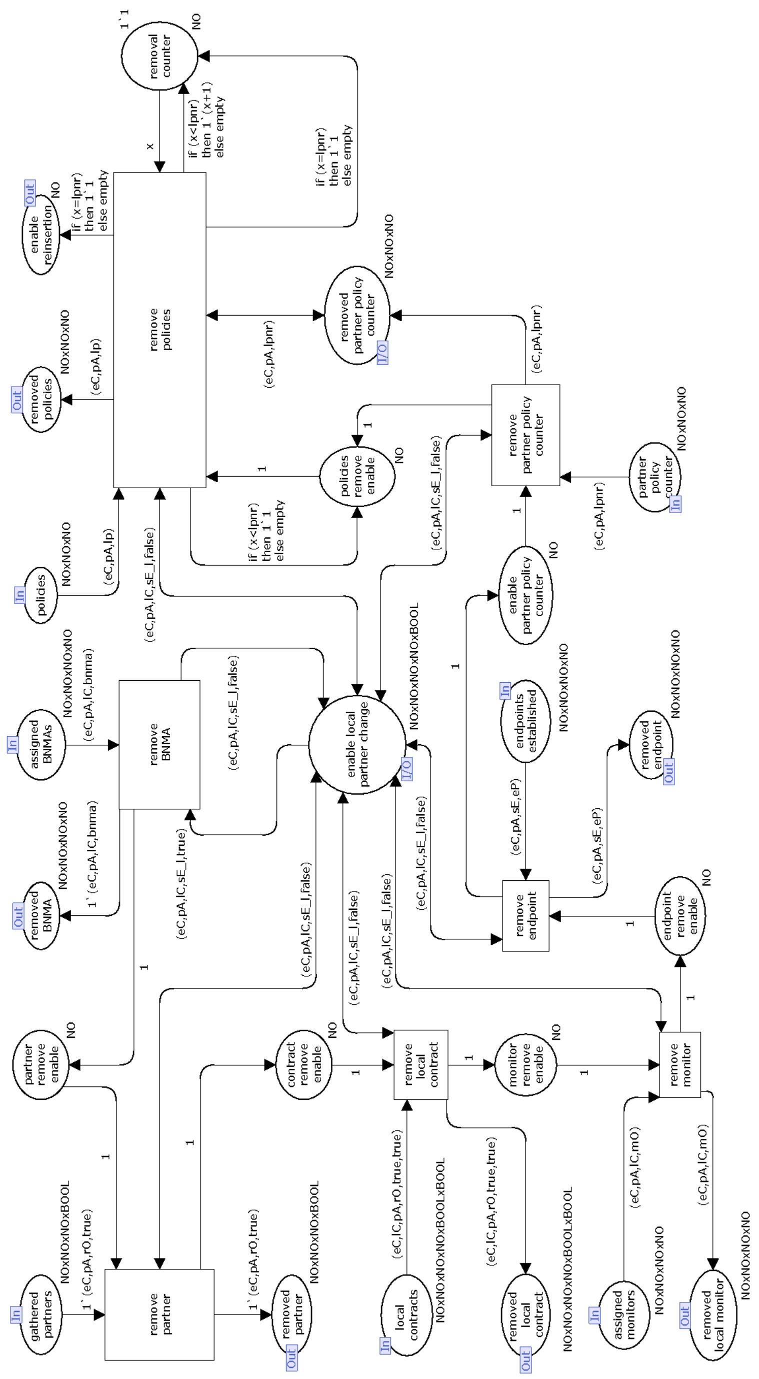


Figure 6.4: Policy extraction from local contract copies and assignment of monitors and BNMAs (*remove*).

6.4 nondisruptively change

A business-semantics rollback-option in Figure 6.6 that leaves the existing DGI intact, replaces a local contract with a modified version. Thus, a composed identification key

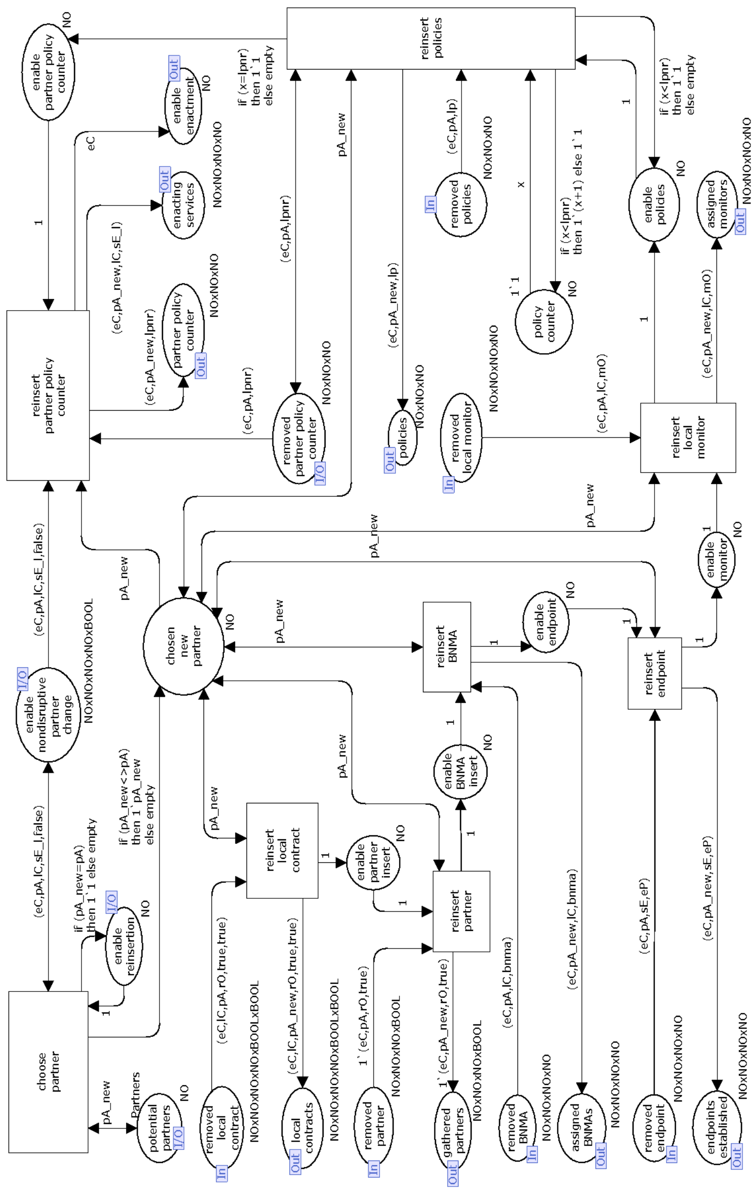


Figure 6.5: Policy extraction from local contract copies and assignment of monitors and BNMAs (*reinsert*).

comprises the unique key of the eCommunity and partner who's local contract requires modification.

As a modification trigger, we consider, e.g., a minor change in the business environment of the party that is not significant enough to justify an entirely newly created DGI. Figure 6.6 shows that a nested subservice first stepwise temporarily removes a local contract and related elements from an existing DGI that requires modification. When local removal completes, the enabling of the $insertion$-service repopulates the DGI with a modified local contract by rolling back into the Figure 5.4 depicted *governance distribution* service.

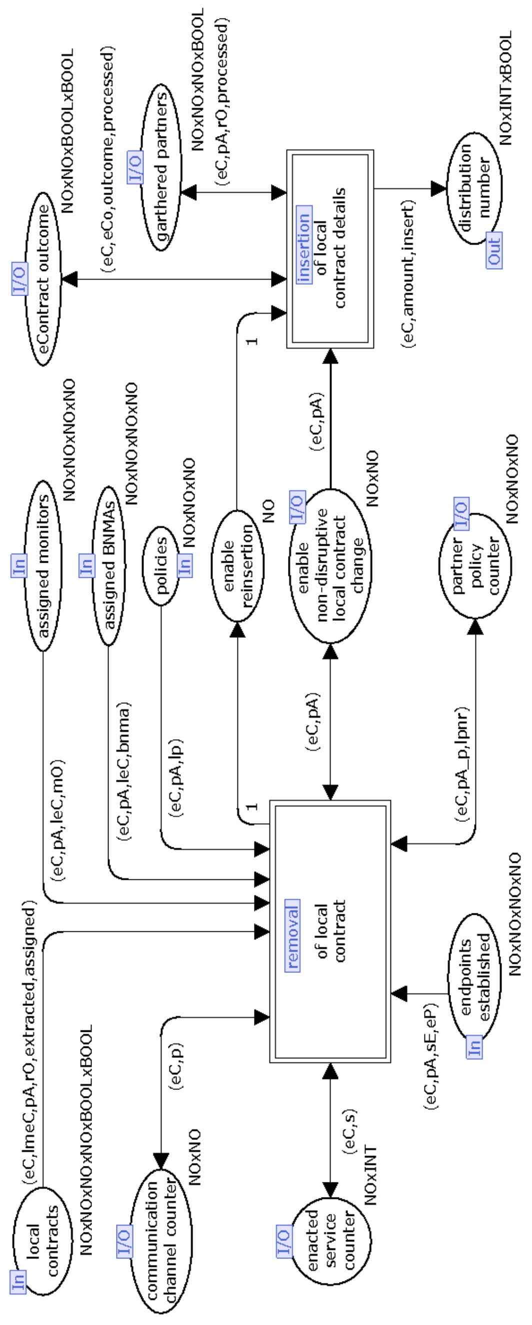


Figure 6.6: Protocol for enacting and terminating an eContract (*nondisruptively change*).

6.4.1 removal

This service in Figure 6.7 is organized in a cycle around the state labeled *enable non disruptive local contract change*. The enabling token for local contract removal comprises a tuple with colors representing a composed identification key, i.e., the unique number of the eCommunity and party who's local contract requires modification, all elements related to the local contract get stepwise removed while leaving the rest of the DGI intact.

6.4.2 insertion

In Figure 6.8, the DGI repopulation initiation follows a contract modification. The $insertion$-service does not refine the actual contract modification process that we consider future work. Contract modification takes place in the transition labeled *insertion of local contracts*, which is currently a black-box.

The re-completion of the DGI results from the $insertion$-service by delivering the changed local contract via the *contract outcome* state to the service termed *governance distribution* that is part of the contract establishment. The consequence is a resumption of overall eContract enactment after completion.

6.5 nondisruptively manage

If a policy violation starts the service in Figure 6.9, the first procedure is a vote of the eCommunity about the severity of that respective violation. Since the refinement with elaborate voting mechanisms is out of scope and part of future work, we assume a randomly chosen token from the state labeled *vote options* determines which connected path is taken.

In Figure 6.9, one path option is that the eCommunity ignores a policy violation, e.g., because of its insignificance. The response is to reinject the violated policy and related electronic service back into a resumed eContract enactment. The $seond$ possible voting outcome is the eCommunity decides to either disruptively or non-disruptively replace a partner. Note, the first case dismantles the entire DGI and allows current eCommunity partners to not be part of a new eContract. The latter case keeps the DGI intact and inserts a new eCommunity-partner. The $third$ voting outcome assumes the policy violation is desirable, e.g., because of an unpredicted change of the eCommunity environment that results in a ”pragmatic” violation. Thus, this fact is acknowledged by the eCommunity and the consensus response is to replace the unsuitable local policy. Finally, the eCommunity agrees to reconcile the committed policy violation, e.g., a warning issuance for the concerned party to not repeat a violation. Since Figure 6.9 does nor refine the reconciliation option, we consider it too future work.

6.6 Conclusion

This chapter specifies the Saga-type business-semantics rollback- and compensation steps in a semantically deterministic way for the main phase of an eBT-lifecycle. For ensuring correctness of these business-semantics rollback- and compensations while also knowing where performance peaks occur during runtime, we employ CPN-notation.

The business-semantics rollbacks and compensations are in one case disruptive, namely when a policy violation is so severe that an agreed upon eContract needs renegotiation after a DGI-dismantling. In the remaining three cases, the business-semantics rollbacks and compensations leave the DGI intact and result in minor modifications, namely, the replacement of an eCommunity partner or particular policy; or not performing any adjustments at all.

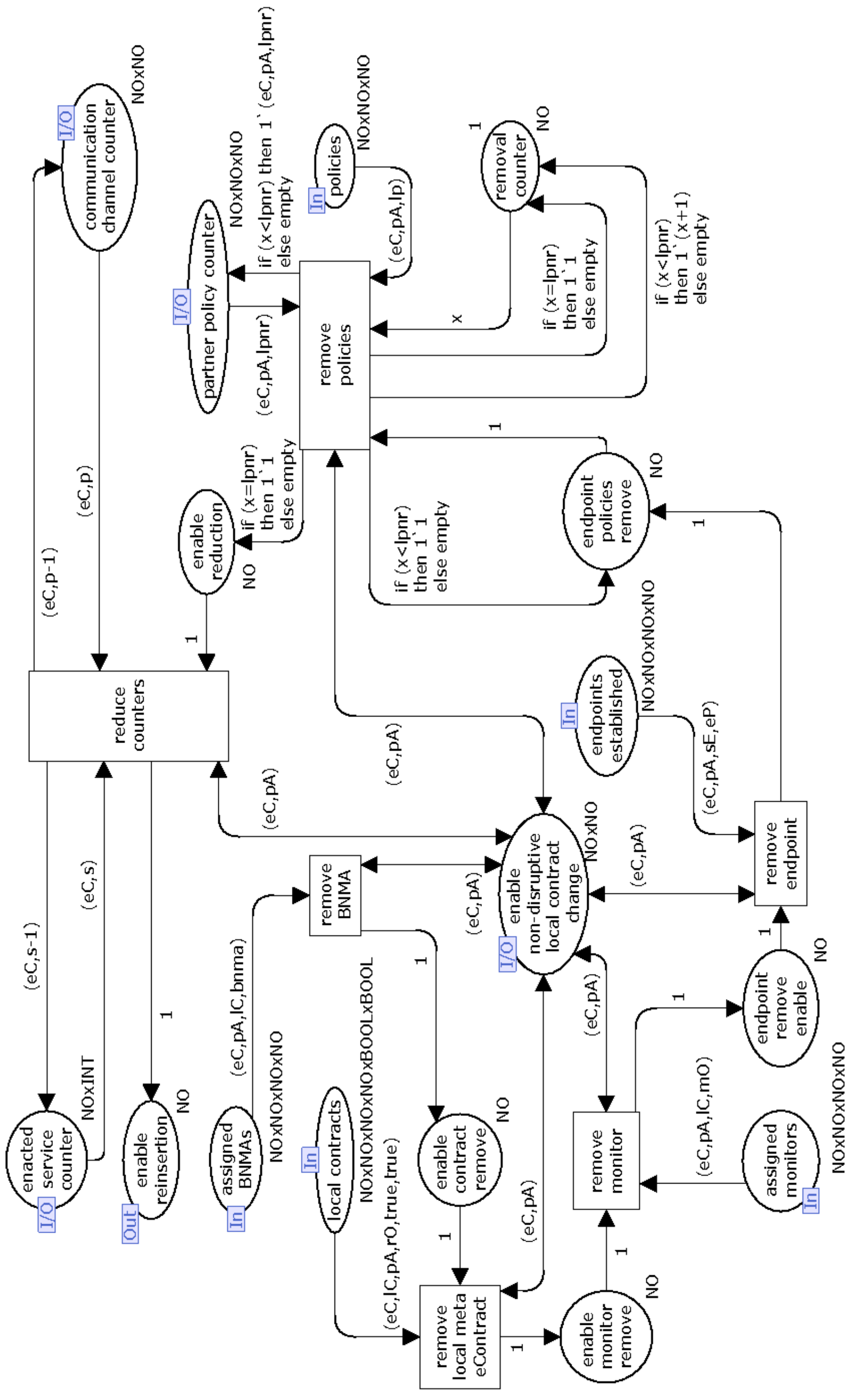


Figure 6.7: Terminating an eCommunity (*removal*).

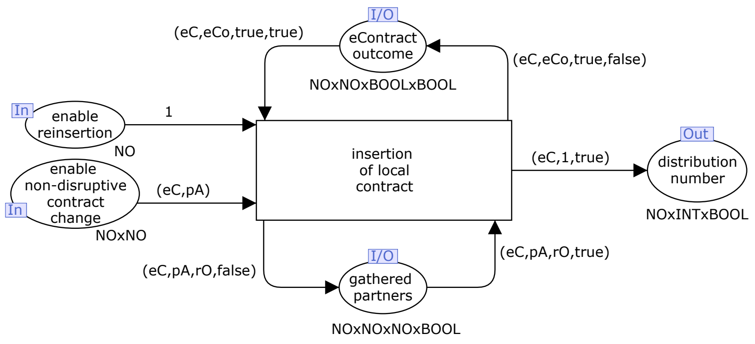


Figure 6.8: Enacting an eContract and triggering business-semantics rollbacks or a termination (*insertion*).

To support these Saga-type business-semantics rollbacks and compensations, a subset of the post-transaction, namely, the $termination$-service; removes either parts of, or the entire DGI. For the final step of the specified Saga-type eBT, the full $termination$ brings the eBT-lifecycle to an end.

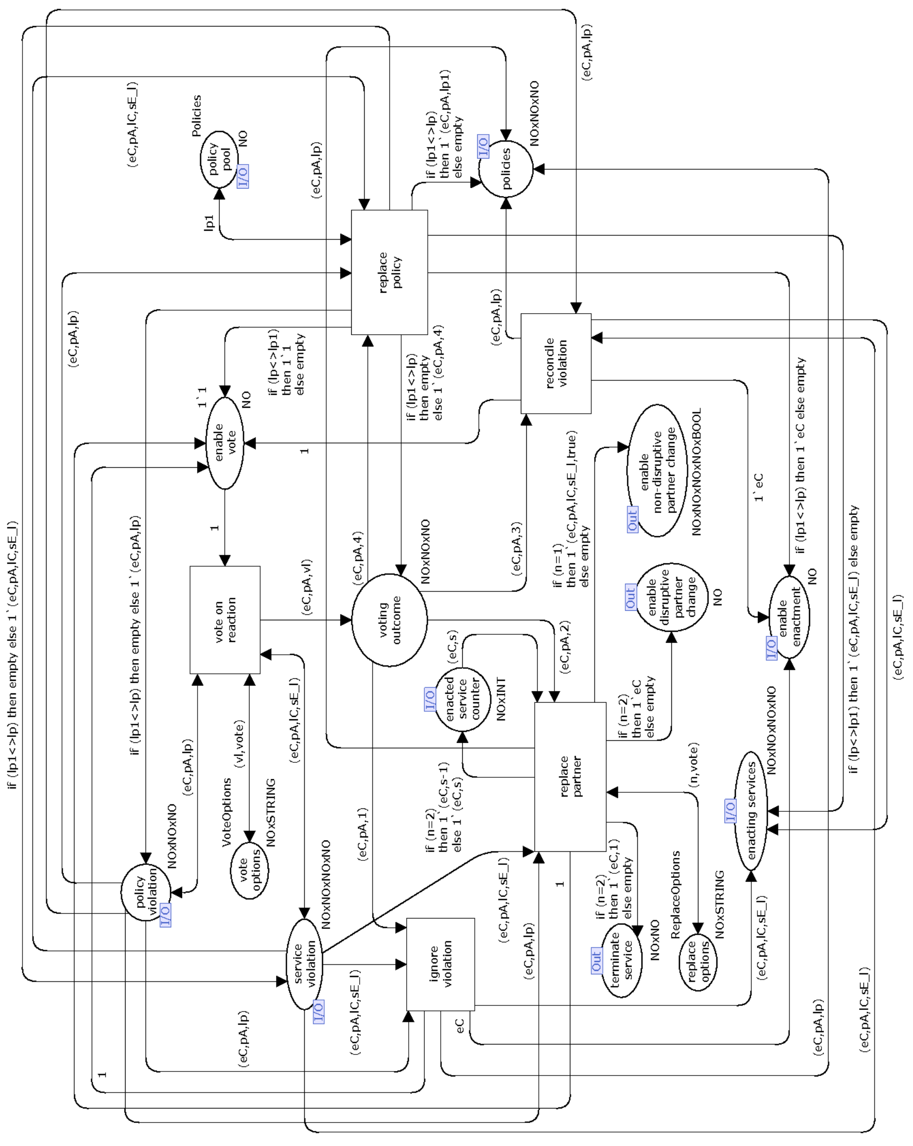


Figure 6.9: Electronic service choosing and communication endpoint creation (*nondisruptively manage*).

Chapter 7

Outlook: aligning transaction quality with business goals

Contents

7.1 Introduction 78
7.1.1 Dependability in eBTs 78
7.2 Specifying transactional quality of service 79
7.2.1 Fluency 80
7.2.2 Alternation 80
7.2.3 Transparency 80
7.2.4 Interferability 81
7.3 Nesting Abstract Transaction Constructs 81
7.3.1 Organizing Abstract Transaction Constructs 82
7.4 Industry initiatives for e-business transactions 84
7.4.1 Business Transaction Protocol 84
7.4.2 Web Services Transactions 84
7.4.3 Web Services Composite Application Framework 85
7.4.4 Comparison of industry initiatives 86
7.5 Conclusion 86

For the Saga-type eBT-lifecycle that we specify in previous chapters with CPN-notation, we explore conceptually the realization of dependability in e-business transactions, through aligning transaction quality measurement with business-level goals. The structure of this chapter, which partially gives an outlook for future eBT-lifecycle refinements, is as follows. Section 7.1 gives an overview of a business-transaction framework to adopt a notion of dependability establishment based on which extensions for the eBT-lifecycle of this manuscript are deducible. Aligned to the adopted notion of dependability, Section 7.2 defines transactional properties particularly suited for service-oriented cloud computing and aligned to the eBT-lifecycle. For the presented business-transaction framework, Section 7.3 nests in a conceptual model abstract representations of existing transaction constructs from Section 2.2 so that they form an inheritance hierarchy. Section 7.4 shows industry standards for instantiating the abstract transaction constructs in a machine-readable way and Section 7.5 gives a conclusion.

7.1 Introduction

Some transaction frameworks integrate several transaction models that is a prerequisite for external-level business-transaction harmonization. We first briefly introduce two relevant older frameworks, namely ACTA and BTF, and then discuss a recent transaction framework called XTC.

The comprehensive framework called ACTA [17] unifies existing models to capture the semantics and reason about the concurrency and recovery properties of complex transactions. More elaborate extensions to this ACTA model comprise in [19, 18, 20]. In the ACTA framework, interactions among transactions are expressed in terms of effects, i.e., effects of transactions on other transactions and effects of transactions on objects they access. ACTA captures the effects of transactions on objects by two object sets and the concept of delegation. Every transaction is associated with several other objects in a view set or access set. The view set contains all the objects potentially accessible to the transaction while the access set contains the objects that have already been accessed by the transaction. Based on ACTA, the ASSET transaction model (Biliris et al. 1994) uses primitives at a programming-language level based on ACTA building blocks such as history, delegation, dependency, conflict set, etc.

A more applicable approach than ACTA for the eBT-framework of Figure 1, is explored in the XTC (eXecution of Transactional Contracted electronic services) project [64, 65] that aims at laying a generic foundation to the transactional support for business processes in a service-oriented environment. Hence, a Business Transaction Framework (BTF) supports contract-driven, cross-enterprise business processes. A BTF is a transaction hierarchy composed of so-called Abstract Transaction Constructs (ATC) that comprise existing transaction models stored in a library. The architecture of the BTF is multi-level and multi-phase [54]. Three phases exist along the BTF lifecycle, namely, definition phase, composition phase and execution phase. During the definition phase, the ATCs adopt the transaction models based on a taxonomy, which covers and classifies the existing work in the transaction-management domain. After the design of an ATC-library contained constructs are available to build a transaction plan for a complex process within the composition phase. The instantiation of the abstract plans resulting from the composition phase [54, 66] form real business transactions for the execution phase.

Next, we conceptually specify in further detail how the BTF from the XTC-approach injects trust in the eBT-lifecycle of previous chapters.

7.1.1 Dependability in eBTs

Based on the research work in [64], the depiction in Figure 7.1 shows what elements ensure dependability in the eBT-lifecycle. Note that the eBT-lifecycle uses eContracts for establishing a DGI of Saga-transactional cross-enterprise collaboration. Thus, we infer that transaction quality can be aligned with business-level goals by checking service-level agreements (SLA) against templates with transactional quality of services (TxQoS). This TxQoS-assignment to the service types of a BNM happens in the breeding ecosystem and the SLA of the service-offers must match. On the other hand, this ensures also a transitive TxQoS-adherence of electronic services chosen for populating the technical enactment level of a DGI. Thus, we assume in the eBT-lifecycle a second SLA-matching step between the service offers embedded in the local copies of an eContract and the electronic services from the technical level.

In Figure 7.1, when an eContract is established it consists of a set of clauses and

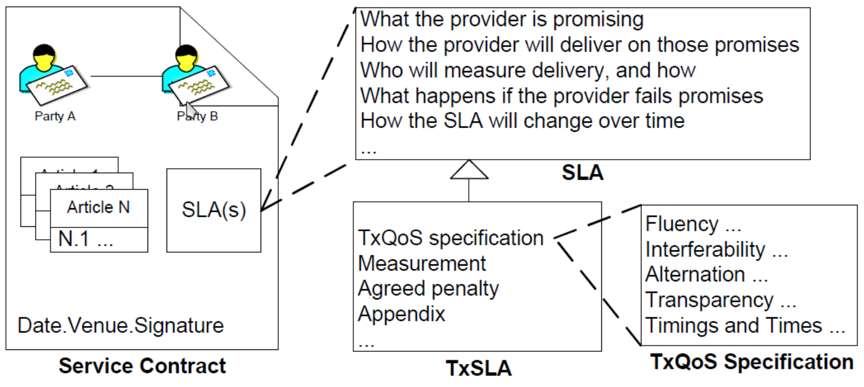


Figure 7.1: Trustworthiness injection in the eBT-lifecycle [64].

enclosed SLAs that emerge based on a consensus of the eCommunity partners. These clauses are obligations and conditional rights and are agreed for the purpose of exchanging value between the two parties.

The enclosed SLAs specify the expected service qualities that the eCommunity partners must adhere to. These SLA formalize an agreement between the parties regarding service availability, performance, measurement, and other aspects that enforce the operational qualities during service execution. When an SLA contain TxQoS, e.g., specification, measurement, penalty, etc., it is termed a TxSLA. The latter are only part of an eContract when the TxQoS specifications by both the provider and consumer agree.

Figure 7.1 depicts conceptually the eContract-SLA-TxSLA-TxQoS structure. The SLA contains the content from the SLA concept defined in [69]. The contract structure bridges the gap between the business and technical worlds by enclosing TxSLAs. The TxSLA contain a TxQoS specification with agreed transactional qualities. This eContract structure enforces execution reliability of an electronic service and guides the technical transaction management mechanisms.

7.2 Specifying transactional quality of service

A TxQoS understanding as depicted in Figure 7.1 for technical and business experts supports a shared understanding of transactional reliability. Thus, a need exists for criteria to specify transactional reliability in service contracts. According to [25], the technical mechanisms listed below ensure safeguarding support.

Recovery, or forward recovery, handles exceptions and errors that occur during service enactment to continue until completion. The recovery may lead to a restart or an alternative execution path. *Concurrency Control* guarantees a consistent execution of applications when they operate simultaneously on the same data. In a cloud-computing setting, currency control keeps common objects safely updated when accessed by multiple service instances. *Compensation* guarantees forward execution by invoking a compensating activity [23] when the original one fails.

Inspired by the traditional ACID-properties [32] from flat DB-transactions, the so-called FIAT-qualities for services [64] of Fluency, Interferablity, Alternation and Transparency are suitable for the TxQoS-specification to establish a trusted eBT-lifecycle. The original definitions for FIAT-properties we adapt to the features of the eBT-lifecycle.

7.2.1 Fluency

The probability of $breakdown$ of a service defines Fluency, which is a numerical value computable into statistical models of choice. A breakdown happens when a service ceases to execute so that there is no delivery of intended results. To calculate the statistical number for breakdown, there are two scenarios to cater for. For unpublished services where no experience data exists, the service publisher gives the initial breakdown number. Otherwise for runtime, monitors collect occurrences of errors and failures together with their timestamps.

There are two levels of fluency for the dependable eBT. On the level of eBT-lifecycle, policies govern the behavior of eCommunity partners. These policies come into existence after extraction from the eContract as Figure 4.8 depicts. Together with assigned monitors, Figure 5.9 depicts the $enact$-service where the recording of policy violations contribute to breakdown calculations. The second level of breakdown recording results on a local electronic service level. The complexity of the business process [45] is a reason for breakdown as high complexity leads to increased design errors that are challenging to detect.

7.2.2 Alternation

This FIAT-property describes the allowed execution paths available as an alternative when a breakdown occurs in an ongoing path. These alternations have two levels in this case. First, on a level of a dependable eBT-lifecycle, pre-defined alternative execution paths result from the specification of business-semantics rollbacks as Chapter 6 specifies. Thus, these hard-specified alternatives for an eContract-enactment exception govern in a disruptive case the total dismantling of a DGI by the termination as in Figure 5.10 and re-negotiation of an eContract in Figure 4.7. That leads to the creation of a new DGI or a final eBT-lifecycle termination. In a non-disruptive case, alternation reasons are policy violations that either lead to an eCommunity partner change into an existing DGI, a policy change, or an enactment resumption after the eCommunity partner decides to ignore a policy violation.

Alternation-specifications may also be on a respective eContract level in a cascading way. First, such alternation-specifications are already part of a service type. Figure 4.2 depicts how service types determine which service offers may pass a conformance validation. If the service type specifies an alternation is optional, it is secondly, possible on the level of the service offer to state whether the alternation either must be present on a level of a local electronic service for enactment, or if the alternation is optional or not not possible.

7.2.3 Transparency

This property determines how visible part of a service execution is for a service consumer to varying degrees during the enactment phase. Note that visibility is specified during the setup phase of a DGI. Thus, just as in the case of alternations, the transparency definition is cascading either on a level of service types that determines the transparency definition on the level of service offers, which in turn determines transparency on the level of a local electronic service. Again, just as for alternation, this cascading definition allows flexibility on a lower level when the higher level states transparency is optional.

Transparency has two levels. The first case is in accordance with the peer-to-peer collaboration nature of the eCommunity-concept where several eCommunity partners connect their respective services to create a bigger, value-added service. Here, transparency definitions for the perspective of control-flow, data-flow, or resource definitions affect visibility for the overall eCommunity. Thus, for control-flow it means that the enactment of a specific service-activity is visible to either all or a subset of an eCommunity. For data-flow, transparency means again that a specific amount of partners has visibility with respect to value changes. Finally, for the resource perspective, transparency can mean that certain eCommunity partners see what concrete persons or organizations fill roles.

In the second case of a consumer and provider for services, transparency levels affect the visibility the first has in service provision. For the context of an eContract that establishes a DGI for enacting a composed, value-added service, we infer the existence of a higher-level service consumer who relates in a pyramidal way to the lower-level eCommunity as Figure 1.1 depicts. For example, the visibility during service enactment allows a consumer to observe when a concrete activity-enactment takes place in the provision.

7.2.4 Interferability

The final FIAT-property, namely, interferability specifies control allowed from service-external parties upon a service invocation during execution time. Again, interferability-specifications happen at design time and cascade from the service type via the service offer to the local electronic service of an eContract equally as in the case of transparency and alternation.

Beforehand agreed upon commands such as $cancel$, $revise$, $rollback$ of business semantics control parts of the service execution [2], which makes this FIAT-property relevant for outsourcing scenarios where a different level of control is necessary [28] in a consumer and provider way as Figure 1.1 depicts. For a peer-to-peer way, interferability means that it is possible for other eCommunity partners to control through agreed commands the local service enactment of another partner. Again, the definition of interferability may be cascading from service type via service offer to local electronic services.

7.3 Nesting Abstract Transaction Constructs

For the emergency eContract in an eBT-lifecycle that several services realize for each respective eCommunity partner, the enactment environment is heterogeneous. Thus, the TxQoS must be flexibly realizable for a heterogeneous system enactment. An example for the approach Figure 6.8 depicts where eight identified ATCs carry the labels A to H. Note, the unnamed activities that belong to activities G and H are not relevant for this conceptual example. ATCs are rectangles and the dashed lines represent encapsulation.

Assigning certain ATCs to different parts of the overall business process of Figure 7.2, results in a specific behavior for exception occurrence that involves a trans-

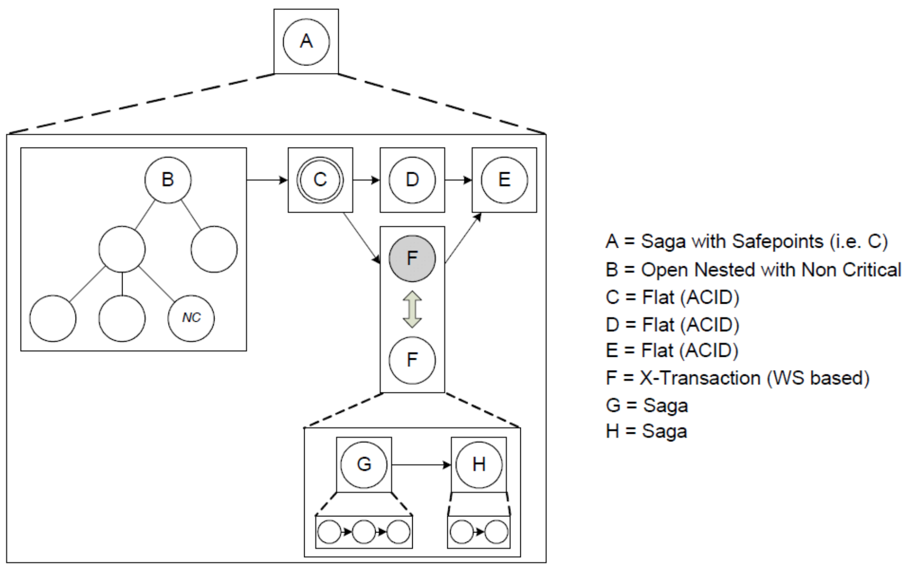

A
B
C
D
E
F
F
NC
G
H
A = Saga with Safepoints (i.e. C)
B = Open Nested with Non Critical
C = Flat (ACID)
D = Flat (ACID)
E = Flat (ACID)
F = X-Transaction (WS based)
G = Saga
H = Saga

Figure 7.2: An example for a nested ATC [64].

action management system. Assigning other ATCs, or by further dividing the process over ATCs, the transactional behavior is different in case exceptions occur. For example, as the complete business process is long-running, a Saga transaction model as specified in Chapter 6 supports the entire process.

A variation of the open nested transaction model supports the activities in Figure 7.2, as these tasks form parallel branches. The execution of the overall electronic service requires safeguarding by a transaction model, while the nested levels may also employ a Saga-transaction again. Chapter 2 describes all other transaction types that Figure 7.2 lists.

Next, we show based on research work from [64], what the nesting taxonomy is of transactions for safeguarding a heterogeneous process.

7.3.1 Organizing Abstract Transaction Constructs

An abstract transaction construct (ATC) is a general, reusable template in an ATC-repository for allowing a run-time transaction-construct instantiation with concrete parameters. Composing the instantiations into a transaction schema creates a BTF that safeguards cross-enterprise business-process collaboration. The BTF is a multi-level transaction scheme that guarantees reliability along the process execution in a best-effort manner. Hence, Figure 7.3 depicts a classification and abstraction of various existing transaction models into three classes with the objective to generalize and abstract commonalities. The hierarchy depicted in Figure 7.3, shows classification [66] according to their structure: The Flat Transaction Model (FTxM) with a flat structure, the Choreographed Transaction Model (CTxM) with a sequence and a complex structure, and the Nested Transaction Model (NTxM).

According to [64], the sequence and complex types stem from the Choreographed group in the classification. For example, the *X-Transaction* model proposed in [62] is a workflow transaction model that leverages the idea of compensation originated from $Sagas$, a sequence type of ATC. In the classification according to Figure 7.3, the X-Transaction model is under *ESTxM* (extended Saga transaction model) within the *CTxM* group. However, if it is defined in the template, the *X-Transaction* model has a complex type of structure. For the transaction models in Figure 7.3, white boxes are abstract, non-executable transaction models and gray boxes are specific enough to be executed, i.e., the gray boxes indicate the real existing transaction models and the white boxes are the generalization of transaction models.

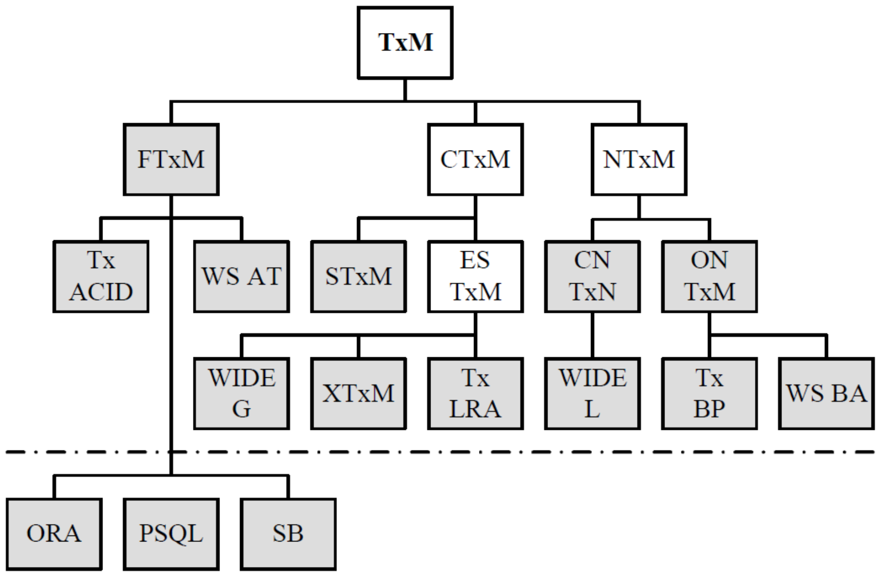


Figure 7.3: Hierarchy of ATCs [64].

Following an inheritance approach comparable to an object-oriented class diagram, the empty transaction model called *TxM* (i.e., transaction model) in Figure 7.3 defines common properties of a transaction model such as name, structural couplings, and transaction management events. Furthermore, the hierarchy in Figure 7.3 identifies and classifies three main transaction groups under *TxM*, namely, the flat transaction model (FTxM), the choreographed transaction model (CTxM), and the nested transaction model (NTxM).

The FTxM adheres to the ACID properties and has extensions, for instance, the two phase commit protocol (2PC) in case of distributed flat transactions. In a service-oriented environment, the flat transaction model varies from, for example, WS Atomic Transactions (WS AT) to the WS-CAF ACID Transactions (Tx ACID).

The CTxM-models are used in environments that require long-lived transactions. Here, a choreographed transaction decomposes a long running transaction into small, (sequentially-executing) sub-transactions. This approach is further specialized, for example into the $Saga$ or extended Saga transaction models as in earlier chapters of this manuscript. The extended Saga transaction model is decomposable into workflow transaction models, like the WIDE Global Transaction Support (WGTxM) (see Chapter 2) and X-Transaction (XTxM) [62]. The Long Running Action (Tx LRA) model from WS-CAF, which we explain in the sequel of this chapter, also fits into this category.

The NTxM-model decomposes top-down a complex transaction into child transactions according to application semantics. NTxMs permit parts of a transaction to fail, without aborting the entire transaction. Further classifications specialize into the open nested and the closed nested transaction model, which is further specialized into, the WIDE Local Transactions model (WLTxM) [30]. Examples of open-nested transaction models in a service-oriented environment are the WS Business Activity (WS BA) and the Business Process Transaction Model (Tx BA) from WS-CAF that are all explained in the sequel.

The next section explains the lower-level transaction models of Figure 7.3 that stem from industry initiatives.

7.4 Industry initiatives for e-business transactions

A survey [66] that covers the development of business transactions up to the latest developments in the domain of service-oriented cloud computing, gives an overview of existing transaction concepts suitable for populating nested levels of the eBT-framework depicted in Figure 2.3.

For service-oriented eBT-applications, there is a need for supporting technologies and standards to guarantee consistency and reliability in eBTs. While no transaction mechanism is widely accepted as a standard, there are three possible candidates, which realize the eBT-framework of Section 2.2 to different degrees.

7.4.1 Business Transaction Protocol

The XML-based business transaction protocol BTP [15] is not exclusively designed for electronic services. BTP is instrumental for representing and seamlessly managing complex, multi-step eBT over the Internet to ensure consistent outcomes of parties that use applications disparate in time, location and administration, and that participate in long running business transactions [41].

In a BTP compliant SOCC-environment, a transaction manager confirms or cancels the backend system a electronic service encapsulates. Hence, a direct communication exists between the transaction manager and the backend system, which contradicts the SOCC philosophy. Opening up backend systems to play the role of participant within the transaction for external parties introduces security issues and bypasses the purpose of SOCC.

Every phase of a transaction within BTP [21] stands on its own and is implementable in any way by a BTP compliant service. To reflect the differences with the traditional 2PC protocol, the commands used in BTP are different and extended. Additionally, using business logic in BTP, the application also determines which participants to commit as a consensus group and which to cancel.

7.4.2 Web Services Transactions

The combined Web Services Transactions specifications that of WS-Tx, consist of WS-Coordination (WS-C) [14], WS-AtomicTransaction (WS-AT) [12], and WS-BusinessActivity (WS-BA) [13]. The specifications aim at the reliable and consistent execution of eBT with different interconnected services.

While in BTP the coordination of an eBT is interwoven with transaction management, WS-Coordination (WS-C) defines a framework that solely focuses on outcome determination and processing. This way WS-C provides a generic coordination infrastructure for services, making it possible to plug in specific coordination protocols [22, 43]. Currently, the WS-Transaction specifications (WS-AT and WS-BA) are the first and only protocol specifications based on WS-Coordination.

The WS-AtomicTransaction (WS-AT) specification focuses on the existing transaction systems and protocols with strict ACID-requirements. These systems are heterogeneous and coupling them together within one organization is the first step towards interoperability. WS-AT specifies the following protocols: Completion, TwoPhase Commit (2PC) with two variants, Volatile 2PC, and Durable 2PC. Details [12] of these protocols the WS-AT specification comprises.

While the WS-AT specification resembles traditional 2PC ACID transactions with its problems, WS-BA supports long running business transactions and uses atomic transactions to preserve the autonomy of participating organizations while it also provides mechanisms to reach overall agreement. The WS-BA specification defines two types for a coordinator, namely the atomic outcome type and mixed outcome type. The first type requires the coordinator to drive all participants to the same final state. The latter type allows a coordinator to choose which participants need to commit or compensate. The behavior of the coordinator is determined by the application driving the activity. Besides the coordination types mentioned above, the following two coordination protocols [14] part of WS-C: $BusinessAgreementWithParticipantCompletion$ and $BusinessAgreementWithCoordinatorCompletion$. We refer the reader to the specification details on these protocols.

7.4.3 Web Services Composite Application Framework

The purpose of WS-CAF [10] is to develop an interoperable, easy to use framework for composite services applications. WS-CAF is composed of a series of specifications consisting of WS Context [8], WS Coordination Framework [9] and WS Transaction Management [11]. Each specification covers a certain level of the overall architecture to build reliable business applications that span multiple systems and use electronic-service technology.

In contrast to BTP and WS-Tx, the WS-CTX specification [8] defines a generic context-management mechanism for sharing common system data (i.e., context) across multiple electronic services. Compared to WS-CTX, WS-Coordination combines both context and coordination, while BTP combines context, coordination as well as transaction management. WS-CTX [8] enables connecting multiple electronic services into one activity or scope for correlating them with specific context information that is not managed by a coordinator.

The second layer of WS-CAF is the Web Service Coordination Framework WS-CF which provides a coordination service plugged into WS-CTX. It manages and coordinates multiple electronic services as one or more activities to perform a task together. The WS-CF architecture has three main services. The $Coordinator$ at which $Participants$ can register so that they receive the context and outcome of an activity, and the *Coordination Service,* which defines the behavior for a specific coordination model.

On top of the coordination framework, Web Service Transaction Management WS-TXM [11] specifies three different transaction protocols. These protocols are an agreement of outcome among the participants of a transaction in a consistent way. To do this, the transaction protocols can use context information and coordination protocols.

WS-TXM defines three protocols that are part of WS-CF and useful with a coordinator for negotiating a set of actions for all participants that need to be executed, based on the outcome of a series of related electronic services executions [10]. The electronic service executions link together in scopes by the overall context, nest and execute concurrently [43]. WS-TXM binds the scope of an activity to the lifetime of a transaction.

Three specific transaction models are part of WS-TXM and useful for different situations: ACID Transaction resembles the traditional ACID transactions applied to electronic services, enabling tightly-coupled network-based transactions to achieve interoperability between existing transaction systems within one organization just like in WS-AT. Long Running Action (LRA) covers transactions that have a long duration where an activity is seen as a set of business interactions for which compensation is possible comparable to Sagas. Business Process Transaction Model integrates different heterogeneous transaction systems, e.g., using ACID transactions and messaging, from different business domains into one overall eBT [43]. This lifecycle is applicable for the external level of the eBT-framework in Figure 2.3.

7.4.4 Comparison of industry initiatives

In a comparison of BTP and WS-Tx [43], both specifications address the problems of running transactions with electronic services while the differences in critical areas are present, e.g., transaction interoperability. The main problem of BTP is it needs to leverage the ACID transactions that underlie the strongly coupled internal information infrastructures instead of replacing them with new models to design transactions for loosely coupled electronic services.

Comparing the specifications shows that a need exists for one open standard to realize the interoperability both in electronic services and business areas, possibly by integrating the existing ones within the WS-CAF framework. According to [36], BTP is the most promising standard candidate for transaction management in combination with agent technology.

With the exception of BTP, the WS-Tx and the WS-CAF initiative show the development towards an eBT with ACID properties on a lower level and advanced business transactions on a higher level. Still, the latter two industry initiatives lack the ability to create elaborate transaction frameworks for inter-organizationally harmonizing heterogeneous transactions of collaborating business domains.

7.5 Conclusion

The outlook of the manuscript explores how to align transaction quality with business goals to achieve dependability in the eBT-lifecycle that earlier chapters specify with CPN-notation. The dependability approach defines transactional qualities for services as part of an eContract to match service-level agreements of service offers and electronic services intended for technical enactment. The safeguarding of an eContract that composes smaller electronic services enact, requires an assembly of nested transactions into an eBT-framework. The specification of the BTF uses abstract transaction concepts that are pluggable abstract templates and transform into instances by populating a template with transactional parameters.

For ensuring dependability in the form of quality-of-service specification, we require additional properties that support transactional safeguarding. Inspired by ACID properties for traditional database transactions, this chapter adapts conceptually defined FIAT-properties to safeguard eBT in cloud-computing. Fluency measures the robustness of service execution. Interferability indicates the extent of control from the outside during execution. Alternation represents the possible choices when encountering problems. Transparency reflects the degree of visibility a service consumer has into an electronic service in the domain of a service provider.

The outlook of this chapter shows open issues for future work. First, the specifications for transactional qualities of services lead to extension proposals of existing industry standards. Also other industry standards for service-level agreements require a comparison to the proposed eBT-framework to explore the suitability of existing industry standards with respect to the concepts and properties they comprise. Second, the abstract transaction frameworks require formalization and based on that a semantically deterministic exploration about their compositionality. It is also open how to achieve the ATF-integration into specifications for transactional qualities of an electronic service. Thirdly, the FIAT-properties relate to each other and have a specific semantic meaningfulness in the context of an eBT-lifecycle. A deeper semantic exploration of the FIAT interdependencies is a prerequisite for achieving an automation realization in a cloud-computing environment.

Chapter 8

Conclusion

Contents

8.1 Research summary 89
8.2 Contributions 90
8.2.1 eBT-lifecycle definition 90
8.2.2 Orderly termination 91
8.2.3 Elastic resource assignment 91
8.2.4 System-architecture specification 92
8.2.5 Dependability extension 92
8.3 Limitations and future work 93

This chapter concludes the manuscript for scientifically investigating dependable eBT-lifecycles. A summary of the conducted research follows a discussion explaining the contributions to our research field. At the end, we discuss the limitations of the thesis, based on which there exists scope for future research activities.

8.1 Research summary

The research work carried out in this manuscript investigates the features of dependable electronic business transactions for cross-enterprise collaboration. The objective is to create a flexible and comprehensive approach for such transactionality safeguarding. The research starts with literature published about the so-called peer-to-peer eCommunity concept, an automated set of business partners who form a temporary group for the purpose of carrying out agreed upon eContracts. An eCommunity experiences changes in case of behavior-guiding policy violations. Consequently, if the eCommunity does not dissolve entirely, it either requires an eContract renegotiation, or partner replacement, or a policy change. With this foundation, the manuscript addresses the following research question: *How can the automation of cross-enterprise collaboration be safeguarded in a heterogeneous system environment so that it pays attention to the semantic complexities of a dependable business transaction? Based on this main research question, we deduce three sub-questions.* Answering this research question first requires a conceptualization and subsequent formalization of the collaborative eCommunity-lifecycle for contracting. That way a semantically deterministic top-level business transaction emerges with business-semantics rollbacks and compensations in the control-flow and data-flow perspective. Secondly, the answer proposes a flexible and comprehensive approach for injecting dependability into a business transaction. We validate the lifecycle of the electronic business transaction for correctness and performance bottlenecks.

8.2 Contributions

The contributions of this research work we give by answering the deduced subquestions. In Section 8.2.1, we describe the formalized lifecycle of the electronic business transaction from the perspectives of control-flow and data-flow. Section 8.2.2 explains a successful partial and total termination of an eCommunity that leaves no lifecycle-collapsing waste behind. Section 8.2.3 shows the results for model-checking of leave services for correctness and performance. Section 8.2.4 summarizes the features of an application-system architecture that support the eBT-lifecycle. Finally, Section 8.2.5 comprises a conceptual investigation for injecting dependability into the electronic business transaction and lists the industry standards available for supporting differing transaction concepts. Each of those subsections first gives the sub-research question and then an the answer.

8.2.1 eBT-lifecycle definition

How is the eBT-lifecycle defined in terms of its collaboration pipeline for transactionality including its contained data flow?

Based on literature, we first deduce an informal and conceptually specified lifecycle with nested services and data-elements that facilitate eBT-management. This informal lifecycle comprises business-semantics rollbacks in the model. For semantically deterministic reasoning and laying an automation foundation, the informal model requires a formalization with CPN-notation that permits simulation, tool-supported correctness verification and performance testing. The formalized model covers the control-flow and data-flow perspective of the eBT-lifecycle commencing from a unique start state and specifies the lifecycle of an eCommunity that transacts until a dissolution results in the eCommunity reaching a unique end state. The formalization reveals a Saga-transactionality with deterministic business-semantics rollbacks and compensations.

The detected elements of the eBT-lifecycle are as follows. A breeding ecosystem generates business-network models equipped with service offers that are conformance-validated against service types, and service-offer-assigned roles that subsequently eCommunity partners fill. A business-network model enters a population stage next for assigning concrete eCommunity partners and tentatively established communication channels. The result is a proto-contract for a consensus-establishment negotiation by all eCommunity partners. A respective partner may propose a counter offer for a re-negotiation. A partner may disagree, which terminates the eBT-lifecycle instantaneously by delivering the eCommunity identification to the unique output state.

Assuming an overall consensus by all partners results in the establishment of an eContract. The rollout of the eContract establishes a distributed governance infrastructure with the eContract functioning as a controlling root agent. Local contract copies for every respective eCommunity partner are under the controlling eContract agent. Local electronic services for enactment that match the service offers in the local contracts relate to the extracted policies, a monitor and business-network model agent for every eCommunity partner. Every local electronic service has communication endpoints that adhere to the earlier established communication channels. At the end of the rollout-stage, the enactment of the local services takes place until exceptions occur that require compensations and business-semantics rollbacks. The business-semantics rollbacks are on the one hand, caused by policy violations that either result in a non-disruptive case with an eCommunity partner exchanged into an existing distributed governance infrastructure, or a policy modification. On the other hand, in a disruptive case, after dismantling the distributed governance infrastructure, a business-semantics rollback into the negotiation stage of the eBT-lifecycle takes place. Finally, the termination-stage totally dismantles the distributed governance infrastructure and delivers the eCommunity identification to the end.

The eBT-lifecycle has a data-flow with data-visibility. Table 3.2 lists the complete set of data-elements with their explanation and refinement-level presence. The data-flow elements are token colors in the CPN-formalization and the inscription definitions of the respective CPN-state show at what lifecycle-stages the respective data-flow elements have visibility.

8.2.2 Orderly termination

How is the orderly partial or complete termination of a lifecycle for an electronic business transaction ensured without leaving behind abandoned processes and database entries that lead to a cloud-collapse?

When an established distributed governance infrastructure for an eContract is in the $enact$-stage, a disruptive partner change leads to a partial termination. Thus, the violation is so severe that it leads to a dismantling of the distributed governance infrastructure as shown in Figure 3.2. However, the eCommunity identification returns to the $negotiate$-stage where the earlier established proto-contract enters the negotiation again. Thus, the eBT-lifecycle remains partially populated in the case of a partial termination with the results of the $populate$-stage.

A total termination may occur because no reason exists any more for the eCommunity to continue contracting. Thus, a dismantling of the distributed governance infrastructure takes place as in the case of partial termination, followed by an emptying of the input states for the $negotiate$-service and the identification of the eCommunity identification enters the unique eBT-lifecycle output state. Left over in the eBT-lifecycle are the elements in the input states of the $populate$-service related to a populated business network model.

8.2.3 Elastic resource assignment

Assuming a cloud-computing environment that caters for elastic resource assignment, where are performance spikes in the eBT-lifecycle?

Given the problem of state-space explosion, the leave-services of the eBT-lifecycle we check for performance bottlenecks as the higher-level services are relevant for understanding the data-flow protocol. The listed services in Table 3.3 are pragmatically chosen with respect to their testability. For the overall eBT-lifecycle, we employ an automated token-game simulation for evaluating the overall termination correctness. For the $negotiate$-service, there are three separate outcomes, namely, either all eCommunity members agree on a contract proposal, or they newly negotiate a counteroffer, or one eCommunity disagrees entirely and terminates the proposal.

We infer, high numbers in tokens in the upper best integer bound of input- and output states in combination with enhanced transition fairness show performance peaks in respective services. With the exception of the services *BNM selection* and *terminate*, the remaining services in Table 3.3 have peaks that predictably require elastic resource assignment in a cloud-computing environment. For the $populate$-service, peaks occur not only for populating roles but also for checking if channel requirements and data-semantics match.

For all three negotiation cases, contract extraction represents a peak and for the cases with forced agreement and a counteroffer, performance peaks occur in agreement finalizing during preparing the $negotiate$-service for the next eContract negotiation. For negotiation with counteroffer, predictable performance peaks occur in the distribution of new contracts to eCommunity partners. For *governance distribution* and $extraction$, the extraction of policies (see Figure 4.8) is most performance demanding. Finally, in the $preparation$-service, assigning electronic services is most performance intensive, followed by creating and publishing their endpoints and checking for operationality. In Section 3.5, we explain the remaining model-checking results of Table 3.3.

8.2.4 System-architecture specification

How is a system architecture specified for privacy protection so that it supports the automation of the eBT-lifecycle?

The architecture design follows the principles of functional decomposition and whole-part to achieve a separation of concerns in a top-down way. As the eBT-framework depicted in Figure 2.3 is structured according to three layers, the same structure is found in the eBT system architecture. The latter's specification is abstract and semi-formal to allow a population with concrete technology in instantiations. Furthermore, the system architecture supports modifiability and loose coupling, which makes it suitable for a service-oriented cloud-computing environment.

The transactional lifecycle of an eCommunity populates the services of a privacy-preserving collaboration architecture. Hence, privacy in the eBT-lifecycle is ensured by using the layer pattern to protect business secrets that constitute competitive advantages and also to secure the legacy information-system infrastructure from corruption.

The collaboration architecture aligns with the specified framework of electronic business transactions as they both have a compatible layer architecture in common. That way, system implementers have a conceptual indication of service functionality and runtime behavior. The eBT-lifecycle that populates the application system-architecture, indicates business semantics and resembles an open, nested Saga transaction.

8.2.5 Dependability extension

How must the architecture be extended with features for ensuring the dependability of an eBT for cross-enterprise collaboration?

To answer this research question, we assume the establishment of dependability results from matching requested and offered service qualities. Thus, the first step is an exploration of the features for such a dependability extension of an eBT that we conduct as a conceptual outlook in this manuscript. Based on these conceptual findings, conclusions result for extensions of the specified eBT-lifecycle.

For the first extension option, transactional quality specifications exposed together with service-requests in a $rendevous$ place that resembles an intelligent broker, require matching with SLA specifications of service-offers that express comparable quality levels. Thus, the architecture extension in the breeding ecosystem of Section 4.2 comprises on the one hand, service matching abilities in a varied way from informal and human readable to formal and machine readable. Since the collaboration type between eCommunity partners is peer-to-peer, in the eBT-lifecycle the service-request is the service type to which a service offer must conformance validate. Thus, in that part of *BNM selection*, the matching of transactional qualities of services of the service type takes place against the service-level agreement of the service offer. A similar matching must occur between the latter is matched with a concrete local electronic service. Section 5.2.2 specifies the latter matching and in Figure 5.6 the *assign service* transition, the service-level agreement matching occurs if that transition is refined in future work.

The second extension option addresses the service-level agreements that comprise so-called FIAT-properties for fluency, alternation, transparency, and interferability of services. $Fluency$ represents the concern for customers on the smoothness of a service. Satisfaction increases when the fluency is high. For supporting the calculation of the fluency statistic, the $enact$-service in Figure 5.9 must log statistics that capture fluency failures, e.g., exceptions, policy violations. $Transparency$ specifies how much of the service execution details are exposed to the outside, i.e., the service consumer or other eCommunity-partners. Extensions of transparency in the $enact$-service on the one hand, allow other eCommunity-partners to monitor enactment progress of a respective local electronic service but also signal on the other hand, monitoring data to third parties from outside of the eCommunity that are consumers of the overall composed service. The $Interferability$-property expresses the limitation of service providers in service execution enforced by other eCommunity-partners. Also in this case, extensions for interferability in the $enact$-service must on the one hand, allow cancellation, pausing, and so on, for permitted eCommunity partners and also for third-party eCommunity-partners. Finally, the property $Alternation$ specifies alternative enactment paths that are different from the execution path at the locations where exceptions are probable to occur. This property is currently covered on an eBT-lifecycle level by offering various disruptive and non-disruptive business-semantics rollbacks as Chapter 6 explains. On the level of services, enactment paths comprise alternations for exception occurrence that achieve business-semantics rollbacks and compensations.

The third extension relates to the management and instantiation of abstract transaction constructs for safeguarding an overall composed service of an eCommunity. Figure 7.3 shows various transaction concepts organized in a feature-inheritance hierarchy. As depicted in Figure 7.2, abstract transaction concepts safeguard a composed service after an instantiation process, e.g., with industry standards as Section 7.4 explains. Thus, a second class of intelligent broker needs to offer semi-automated support for managing the hierarchy of abstract transaction concepts. Furthermore, such a broker must offer instantiation options of a concrete ATC with an industry standard of choice. The instantiations require on-the-fly assembly into a business-transaction framework and assignment to the composed services of eCommunities the business-transaction framework safeguards.

8.3 Limitations and future work

While the conceptual specifications succeed in organizing eBT-safeguarded collaboration, electronic automation requires further formalization. Thus, open issues are the semantically precise collaboration control-, and data flow. That way we aim to clarify precise features of eBT.

The verification of the eBT-lifecycle currently focuses on the leave services as listed in Table 3.3 where the state-space explosion is manageable. The overall eBT-lifecycle we simulate in a manual and automated way in CPNTools. The tested services are either atomic refinement leaves without CPN-modules or comprise a module that itself only contain atomic transitions. In the future, an overall verification is achievable with more computing power available. Thus, we plan to use a computing grid for verification that is capable of managing the state-space explosion problem.

The outlook of this chapter shows open issues for future work. First, the specifications for transactional qualities of services should lead to extension proposals of existing industry standards. Also other industry standards for service-level agreements require a comparison to the proposed business-transaction framework to explore what the existing standards don't cover. Second, the abstract transaction frameworks require formalization and based on a semantically deterministic exploration about their composability. It is also open how to precisely achieve the ATF-integration into specifications for transactional qualities of a service. Secondly, the FIAT-properties relate to each other and have a specific formal meaningfulness in the context of the CPN-defined eBT-lifecycle. A deeper semantic exploration of the FIAT interdependencies is still open but a prerequisite for achieving an automation realization in a cloud-computing environment.

Finally, we plan to select respective CPN-models and design services in different notations, e.g., from UML. The resulting set of models require implementation in a set of prototypes that realize respective parts of the eBT-lifecycle. The objective is o eventually have a complete implementation to permit conducting radically service-oriented cross-enterprise collaboration in a cloud-computing environment.

Appendix A

BNM selection

Contents

A.1 Statistics 95
A.2 Boundedness Properties 95
A.3 Home Properties 100
A.4 Liveness Properties 100
A.5 Fairness Properties 100

A.1 Statistics

```
 State Space
     Nodes:  46630
     Arcs:   114684
     Secs:   300
     Status: Partial

  Scc Graph
     Nodes:  46630
     Arcs:   114684
     Secs:   6
```

A.2 Boundedness Properties

Best Integer Bounds

```
                               Upper       Lower
     BNM_selection'BNM 1       2           0
     BNM_selection'conformance_validated_service_offers 1
                               4           0
     BNM_selection'repository_service_offers 1
                               6           6
     BNM_selection'repository_service_types 1
                               6           6
     BNM_selection'role_counter 1
```

```
                                    3               0
     BNM_selection'selected_BNM_draft 1
                                    3               0
     repository_accessing'BNM_drafts 1
                                    3               3
     repository_accessing'roles 1
                                    6               6
```

Best Upper Multi-set Bounds

```
     BNM_selection'BNM 1  2`(1,1,1,false)++
2`(1,1,2,false)++
2`(1,1,3,false)++
2`(1,1,4,false)++
2`(1,1,5,false)++
2`(1,1,6,false)++
2`(1,2,1,false)++
2`(1,2,2,false)++
2`(1,2,3,false)++
2`(1,2,4,false)++
2`(1,2,5,false)++
2`(1,2,6,false)++
2`(1,3,1,false)++
2`(1,3,2,false)++
2`(1,3,3,false)++
2`(1,3,4,false)++
2`(1,3,5,false)++
2`(1,3,6,false)++
2`(1,4,1,false)++
2`(1,4,2,false)++
2`(1,4,3,false)++
2`(1,4,4,false)++
2`(1,4,5,false)++
2`(1,4,6,false)++
2`(1,5,1,false)++
2`(1,5,2,false)++
2`(1,5,3,false)++
2`(1,5,4,false)++
2`(1,5,5,false)++
2`(1,5,6,false)++
2`(1,6,1,false)++
2`(1,6,2,false)++
2`(1,6,3,false)++
2`(1,6,4,false)++
2`(1,6,5,false)++
2`(1,6,6,false)++
2`(2,1,1,false)++
2`(2,1,2,false)++
2`(2,1,3,false)++
2`(2,1,4,false)++
2`(2,1,5,false)++
2`(2,1,6,false)++
2`(2,2,1,false)++
2`(2,2,2,false)++
2`(2,2,3,false)++
2`(2,2,4,false)++
2`(2,2,5,false)++
2`(2,2,6,false)++
2`(2,3,1,false)++
2`(2,3,2,false)++
2`(2,3,3,false)++
2`(2,3,4,false)++
2`(2,3,5,false)++
2`(2,3,6,false)++
2`(2,4,1,false)++
2`(2,4,2,false)++
2`(2,4,3,false)++
2`(2,4,4,false)++
2`(2,4,5,false)++
2`(2,4,6,false)++
2`(2,5,1,false)++
2`(2,5,2,false)++
2`(2,5,3,false)++
2`(2,5,4,false)++
2`(2,5,5,false)++
2`(2,5,6,false)++
2`(2,6,1,false)++
2`(2,6,2,false)++
2`(2,6,3,false)++
2`(2,6,4,false)++
2`(2,6,5,false)++
2`(2,6,6,false)++
2`(3,1,1,false)++
2`(3,1,2,false)++
2`(3,1,3,false)++
2`(3,1,4,false)++
2`(3,1,5,false)++
2`(3,1,6,false)++
2`(3,2,1,false)++
2`(3,2,2,false)++
2`(3,2,3,false)++
2`(3,2,4,false)++
2`(3,2,5,false)++
2`(3,2,6,false)++
2`(3,3,1,false)++
2`(3,3,2,false)++
2`(3,3,3,false)++
2`(3,3,4,false)++
2`(3,3,5,false)++
2`(3,3,6,false)++
2`(3,4,1,false)++
2`(3,4,2,false)++
2`(3,4,3,false)++
2`(3,4,4,false)++
2`(3,4,5,false)++
2`(3,4,6,false)++
2`(3,5,1,false)++
2`(3,5,2,false)++
2‘(3,5,3,false)++
2‘(3,5,4,false)++
2‘(3,5,5,false)++
2‘(3,5,6,false)++
2‘(3,6,1,false)++
2‘(3,6,2,false)++
2‘(3,6,3,false)++
2‘(3,6,4,false)++
2‘(3,6,5,false)++
2‘(3,6,6,false)
    BNM_selection'conformance_validated_service_offers 1
                    3‘(1,1)++
3‘(1,2)++
3‘(1,3)++
3‘(1,4)++
3‘(1,5)++
3‘(1,6)++
3‘(2,1)++
3‘(2,2)++
3‘(2,3)++
4‘(2,4)++
3‘(2,5)++
4‘(2,6)++
3‘(3,1)++
3‘(3,2)++
3‘(3,3)++
3‘(3,4)++
3‘(3,5)++
3‘(3,6)++
3‘(4,1)++
3‘(4,2)++
3‘(4,3)++
3‘(4,4)++
3‘(4,5)++
3‘(4,6)++
3‘(5,1)++
3‘(5,2)++
3‘(5,3)++
3‘(5,4)++
3‘(5,5)++
3‘(5,6)++
3‘(6,1)++
3‘(6,2)++
3‘(6,3)++
3‘(6,4)++
3‘(6,5)++
3‘(6,6)
    BNM_selection'repository_service_offers 1
                    1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
```

```
     BNM_selection ’repository_service_types 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     BNM_selection ’role_counter 1
                         3‘(1,0)++
1‘(1,1)++
1‘(1,2)++
3‘(2,0)++
1‘(2,1)++
1‘(2,2)++
3‘(3,0)++
1‘(3,1)++
1‘(3,2)
     BNM_selection ’selected_BNM_draft 1
                         3‘(1,2)++
3‘(2,4)++
3‘(3,3)
     repository_accessing ’BNM_drafts 1
                         1‘(1,2)++
1‘(2,4)++
1‘(3,3)
     repository_accessing ’roles 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
```

Best Lower Multi-set Bounds

```
     BNM_selection ’BNM 1 empty
     BNM_selection ’conformance_validated_service_offers 1
                         empty
     BNM_selection ’repository_service_offers 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     BNM_selection ’repository_service_types 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     BNM_selection ’role_counter 1
                         empty
     BNM_selection ’selected_BNM_draft 1
```

```
                                   empty
       repository_accessing 'BNM_drafts  1
                                   1'(1,2)++
1'(2,4)++
1'(3,3)
       repository_accessing 'roles  1
                                   1'1++
1'2++
1'3++
1'4++
1'5++
1'6
```

A.3 Home Properties

Home Markings None

A.4 Liveness Properties

Dead Markings 44174 [9999,9998,9997,9996,9995,...]
    Dead Transition Instances None
    Live Transition Instances None

A.5 Fairness Properties

No infinite occurrence sequences.

Appendix B

populate

Contents

B.1 Statistics 101
B.2 Boundedness Properties 101
B.3 Home Properties 105
B.4 Liveness Properties 105
B.5 Fairness Properties 106

B.1 Statistics

```
   State Space
      Nodes:   940
      Arcs:    1598
      Secs:    0
      Status:  Full

  Scc Graph
      Nodes:   940
      Arcs:    1598
      Secs:    0
```

B.2 Boundedness Properties

Best Integer Bounds

```
                                 Upper        Lower
      finalize'D 1               1            0
      finalize'E 1               1            0
      finalize'F 1               1            0
      finalize'eCommunity 1      1            0
      finalize'finalizing_count 1
                                 1            1
      interoperability_checking'A 1
                                 1            0
```

```
interoperability_checking'B 1
                          1         0
interoperability_checking'C 1
                          1         0
interoperability_checking'picked_source 1
                          1         0
interoperability_checking'picked_target 1
                          1         0
interoperability_checking'stored_count 1
                          1         0
populate'A 1              1         0
populate'BNM 1            3         3
populate'channel_counter 1
                          1         1
populate'channel_number 1
                          1         0
populate'channels_tentatively_established 1
                          2         0
populate'conformance_validated_service_offers 1
                          3         3
populate'eContract_proposal 1
                          1         0
populate'enable_completion 1
                          1         0
populate'enable_populating 1
                          1         0
populate'enabled_checking 1
                          1         0
populate'in_population 1
                          1         0
populate'populated 1      3         0
populate'populating_count 1
                          1         0
populate'population_counter 1
                          1         1
populate'role_counter 1 1           1
populate'roles_number 1 1           0
populate'selected_BNM_draft 1
                          1         1
populate'service_offers_assigned 1
                          3         0
populate'start_lifecycle 1
                          1         0
```

Best Upper Multi-set Bounds

```
    finalize'D 1          1`1
    finalize'E 1          1`1
    finalize'F 1          1`1
    finalize'eCommunity 1
                          1`1
    finalize'finalizing_count 1
                          1`1++
1`2++
1`3++
1`4
     interoperability_checking'A 1
                         1`1
     interoperability_checking'B 1
                         1`1
     interoperability_checking'C 1
                         1`1
     interoperability_checking'picked_source 1
                         1`(1,1,1,1)++
1`(1,2,1,2)++
1`(1,3,1,3)
     interoperability_checking'picked_target 1
                         1`(1,1,1,1)++
1`(1,2,1,2)++
1`(1,3,1,3)
     interoperability_checking'stored_count 1
                         1`(3,1)++
1`(3,2)
     populate'A 1        1`1
     populate'BNM 1      1`(1,1,1,false)++
1`(1,1,1,true)++
1`(1,2,2,false)++
1`(1,2,2,true)++
1`(1,3,3,false)++
1`(1,3,3,true)
     populate'channel_counter 1
                         1`0++
1`1++
1`2
     populate'channel_number 1
                         1`(1,2)
     populate'channels_tentatively_established 1
                         1`(1,1,2,0)++
1`(1,1,2,1)++
1`(1,1,3,0)++
1`(1,1,3,1)++
1`(1,2,1,0)++
1`(1,2,1,1)++
1`(1,2,3,0)++
1`(1,2,3,1)++
1`(1,3,1,0)++
1`(1,3,1,1)++
1`(1,3,2,0)++
1`(1,3,2,1)
     populate'conformance_validated_service_offers 1
                         1`(1,1)++
1`(2,2)++
1`(3,3)
     populate'eContract_proposal 1
                         1`(1,1,true)
     populate'enable_completion 1
                         1`1
     populate'enable_populating 1
```

```
                              1‘1
     populate’enabled_checking 1
                              1‘3
     populate’in_population 1
                              1‘(1,1,2)
     populate’populated 1
                              1‘(1,1,1,1)++
1‘(1,2,1,2)++
1‘(1,3,1,3)
     populate’populating_count 1
                              1‘(0,2)++
1‘(1,2)++
1‘(2,2)++
1‘(3,0)++
1‘(3,1)++
1‘(3,2)
     populate’population_counter 1
                              1‘1++
1‘2++
1‘3
     populate’role_counter 1
                              1‘(1,3)
     populate’roles_number 1
                              1‘(1,3)
     populate’selected_BNM_draft 1
                              1‘(1,2)
     populate’service_offers_assigned 1
                              1‘(1,1,1,false)++
1‘(1,2,2,false)++
1‘(1,3,3,false)
     populate’start_lifecycle 1
                              1‘1
```

Best Lower Multi-set Bounds

```
     finalize’D 1             empty
     finalize’E 1             empty
     finalize’F 1             empty
     finalize’eCommunity 1
                              empty
     finalize’finalizing_count 1
                              empty
     interoperability_checking’A 1
                              empty
     interoperability_checking’B 1
                              empty
     interoperability_checking’C 1
                              empty
     interoperability_checking’picked_source 1
                              empty
     interoperability_checking’picked_target 1
                              empty
     interoperability_checking’stored_count 1
                              empty
     populate'A 1           empty
     populate'BNM 1         empty
     populate'channel_counter 1
                            empty
     populate'channel_number 1
                            empty
     populate'channels_tentatively_established 1
                            empty
     populate'conformance_validated_service_offers 1
                            1`(1,1)++
1`(2,2)++
1`(3,3)
     populate'eContract_proposal 1
                            empty
     populate'enable_completion 1
                            empty
     populate'enable_populating 1
                            empty
     populate'enabled_checking 1
                            empty
     populate'in_population 1
                            empty
     populate'populated 1
                            empty
     populate'populating_count 1
                            empty
     populate'population_counter 1
                            empty
     populate'role_counter 1
                            1`(1,3)
     populate'roles_number 1
                            empty
     populate'selected_BNM_draft 1
                            1`(1,2)
     populate'service_offers_assigned 1
                            empty
     populate'start_lifecycle 1
                            empty
```

B.3 Home Properties

Home Markings None

B.4 Liveness Properties

Dead Markings 72 [940,939,938,937,936,...]

Dead Transition Instances None

Live Transition Instances None

B.5 Fairness Properties

No infinite occurrence sequences.

Appendix C

negotiate with forced agreement

Contents

C.1 Statistics 107
C.2 Boundedness Properties 107
C.3 Home Properties 114
C.4 Liveness Properties 114
C.5 Fairness Properties 114

C.1 Statistics

```
State Space
    Nodes:   6010
    Arcs:    14759
    Secs:    7
    Status:  Full

Scc Graph
    Nodes:   6010
    Arcs:    11253
    Secs:    0
```

C.2 Boundedness Properties

Best Integer Bounds

```
                              Upper       Lower
agreement_finalizing'agreed 1
                              1           0
agreement_finalizing'all_extracted 1
                              1           0
agreement_finalizing'enable_offers_reset 1
                              0           0
agreement_finalizing'reset_counter 1
```

```
                           1           1
contract_extraction'all_partners_extracted 1
                           1           0
contract_extraction'channel_counter 1
                           1           0
contract_extraction'channels_merged 1
                           1           0
contract_extraction'chosen 1
                           1           0
contract_extraction'merged_channel_spec 1
                           1           1
disagreeing'enable_partner_preparation 1
                           0           0
disagreeing'enable_re 1 0           0
disagreeing'enable_termination 1
                           0           0
disagreeing'need_to_merge 1
                           0           0
disagreeing'partner_preparation_counter 1
                           1           1
disagreeing'remaining_distributed_eContracts 1
                           0           0
disagreement_finalizing'enable_channel_removal 1
                           0           0
disagreement_finalizing'enable_service_offer_removal 1
                           0           0
disagreement_finalizing'partner_removal_counter 1
                           0           0
negotiate'agreed_count 1
                           1           1
negotiate'canceled_agreement 1
                           0           0
negotiate'channel_number 1
                           1           1
negotiate'channels_tentatively_established 1
                           2           2
negotiate'disable 1        0           0
negotiate'distributed_eContract_proposals 1
                           3           0
negotiate'distribution_count 1
                           1           1
negotiate'distribution_number 1
                           1           0
negotiate'eContract_outcome 1
                           1           0
negotiate'eContract_proposal 1
                           1           1
negotiate'enable_closure 1
                           0           0
negotiate'enable_eCommunity_pick 1
                           1           0
negotiate'enable_set_false_outcome 1
                           0           0
negotiate'enable_withdrawal 1
```

```
                                1          0
     negotiate'extracted_eContract_proposal 1
                                1          0
     negotiate'gathered_partners 1
                                3          0
     negotiate'partner_count 1
                                1          0
     negotiate'partners_gathered 1
                                3          0
     negotiate'permit_agree 1
                                1          0
     negotiate'permit_counteroffer 1
                                1          0
     negotiate'potential_partners 1
                                6          6
     negotiate'roles_number 1
                                1          1
     negotiate'service_offers_assigned 1
                                3          3
```

Best Upper Multi-set Bounds

```
     agreement_finalizing'agreed 1
                           1`(1,1,true)
     agreement_finalizing'all_extracted 1
                           1`1
     agreement_finalizing'enable_offers_reset 1
                           empty
     agreement_finalizing'reset_counter 1
                           1`1
     contract_extraction'all_partners_extracted 1
                           1`1
     contract_extraction'channel_counter 1
                           1`0++
1`1
     contract_extraction'channels_merged 1
                           1`1
     contract_extraction'chosen 1
                           1`1
     contract_extraction'merged_channel_spec 1
                           1`""++
1`"SPEC"
     disagreeing'enable_partner_preparation 1
                           empty
     disagreeing'enable_re 1
                           empty
     disagreeing'enable_termination 1
                           empty
     disagreeing'need_to_merge 1
                           empty
     disagreeing'partner_preparation_counter 1
                           1`0
     disagreeing'remaining_distributed_eContracts 1
                           empty
```

```
          disagreement_finalizing'enable_channel_removal 1
                                  empty
          disagreement_finalizing'enable_service_offer_removal 1
                                  empty
          disagreement_finalizing'partner_removal_counter 1
                                  empty
          negotiate'agreed_count 1
                                  1`0++
1`1++
1`2++
1`3
          negotiate'canceled_agreement 1
                                  empty
          negotiate'channel_number 1
                                  1`(1,2)
          negotiate'channels_tentatively_established 1
                                  1`(1,1,2,0)++
1`(1,1,3,1)
          negotiate'disable 1 empty
          negotiate'distributed_eContract_proposals 1
                                  3`(1,1,1)++
3`(1,1,2)++
3`(1,1,3)++
3`(1,1,4)++
3`(1,1,5)++
3`(1,1,6)
          negotiate'distribution_count 1
                                  1`0++
1`1++
1`2++
1`3
          negotiate'distribution_number 1
                                  1`(1,3,false)
          negotiate'eContract_outcome 1
                                  1`(1,1,true,false)
          negotiate'eContract_proposal 1
                                  1`(1,1,false)++
1`(1,1,true)
          negotiate'enable_closure 1
                                  empty
          negotiate'enable_eCommunity_pick 1
                                  1`1
          negotiate'enable_set_false_outcome 1
                                  empty
          negotiate'enable_withdrawal 1
                                  1`(1,"agreed")++
1`(2,"agreed")++
1`(3,"agreed")
          negotiate'extracted_eContract_proposal 1
                                  1`(1,1)
          negotiate'gathered_partners 1
                                  1`(1,1,1,false)++
1`(1,1,2,false)++
1`(1,1,3,false)++
1`(1,2,1,false)++
1`(1,2,2,false)++
1`(1,2,3,false)++
1`(1,3,1,false)++
1`(1,3,2,false)++
1`(1,3,3,false)++
1`(1,4,1,false)++
1`(1,4,2,false)++
1`(1,4,3,false)++
1`(1,5,1,false)++
1`(1,5,2,false)++
1`(1,5,3,false)++
1`(1,6,1,false)++
1`(1,6,2,false)++
1`(1,6,3,false)
     negotiate'partner_count 1
                         1`0++
1`1++
1`2++
1`3
     negotiate'partners_gathered 1
                         1`(1,1,1,1,false)++
1`(1,1,1,1,true)++
1`(1,1,2,1,false)++
1`(1,1,2,1,true)++
1`(1,1,3,1,false)++
1`(1,1,3,1,true)++
1`(1,1,4,1,false)++
1`(1,1,4,1,true)++
1`(1,1,5,1,false)++
1`(1,1,5,1,true)++
1`(1,1,6,1,false)++
1`(1,1,6,1,true)++
1`(1,2,1,2,false)++
1`(1,2,1,2,true)++
1`(1,2,2,2,false)++
1`(1,2,2,2,true)++
1`(1,2,3,2,false)++
1`(1,2,3,2,true)++
1`(1,2,4,2,false)++
1`(1,2,4,2,true)++
1`(1,2,5,2,false)++
1`(1,2,5,2,true)++
1`(1,2,6,2,false)++
1`(1,2,6,2,true)++
1`(1,3,1,3,false)++
1`(1,3,1,3,true)++
1`(1,3,2,3,false)++
1`(1,3,2,3,true)++
1`(1,3,3,3,false)++
1`(1,3,3,3,true)++
1`(1,3,4,3,false)++
1`(1,3,4,3,true)++
1`(1,3,5,3,false)++
1`(1,3,5,3,true)++
1`(1,3,6,3,false)++
1`(1,3,6,3,true)
     negotiate'permit_agree 1
                    1`1
     negotiate'permit_counteroffer 1
                    1`1
     negotiate'potential_partners 1
                    1`1++
1`2++
1`3++
1`4++
1`5++
1`6
     negotiate'roles_number 1
                    1`(1,3)
     negotiate'service_offers_assigned 1
                    1`(1,1,1,false)++
1`(1,1,1,true)++
1`(1,2,2,false)++
1`(1,2,2,true)++
1`(1,3,3,false)++
1`(1,3,3,true)
```

Best Lower Multi-set Bounds

```
     agreement_finalizing'agreed 1
                    empty
     agreement_finalizing'all_extracted 1
                    empty
     agreement_finalizing'enable_offers_reset 1
                    empty
     agreement_finalizing'reset_counter 1
                    1`1
     contract_extraction'all_partners_extracted 1
                    empty
     contract_extraction'channel_counter 1
                    empty
     contract_extraction'channels_merged 1
                    empty
     contract_extraction'chosen 1
                    empty
     contract_extraction'merged_channel_spec 1
                    empty
     disagreeing'enable_partner_preparation 1
                    empty
     disagreeing'enable_re 1
                    empty
     disagreeing'enable_termination 1
                    empty
     disagreeing'need_to_merge 1
                    empty
     disagreeing'partner_preparation_counter 1
                    1`0
```

```
      disagreeing'remaining_distributed_eContracts 1
                              empty
      disagreement_finalizing'enable_channel_removal 1
                              empty
      disagreement_finalizing'enable_service_offer_removal 1
                              empty
      disagreement_finalizing'partner_removal_counter 1
                              empty
      negotiate'agreed_count 1
                              empty
      negotiate'canceled_agreement 1
                              empty
      negotiate'channel_number 1
                              1`(1,2)
      negotiate'channels_tentatively_established 1
                              1`(1,1,2,0)++
1`(1,1,3,1)
      negotiate'disable 1 empty
      negotiate'distributed_eContract_proposals 1
                              empty
      negotiate'distribution_count 1
                              empty
      negotiate'distribution_number 1
                              empty
      negotiate'eContract_outcome 1
                              empty
      negotiate'eContract_proposal 1
                              empty
      negotiate'enable_closure 1
                              empty
      negotiate'enable_eCommunity_pick 1
                              empty
      negotiate'enable_set_false_outcome 1
                              empty
      negotiate'enable_withdrawal 1
                              empty
      negotiate'extracted_eContract_proposal 1
                              empty
      negotiate'gathered_partners 1
                              empty
      negotiate'partner_count 1
                              empty
      negotiate'partners_gathered 1
                              empty
      negotiate'permit_agree 1
                              empty
      negotiate'permit_counteroffer 1
                              empty
      negotiate'potential_partners 1
                              1`1++
1`2++
1`3++
1`4++
1`5++
1‘6
     negotiate’roles_number 1
                       1‘(1,3)
     negotiate’service_offers_assigned 1
                       empty
```

C.3 Home Properties

Home Markings None

C.4 Liveness Properties

Dead Markings 216 [6010,6009,6008,6007,6006,...]
Dead Transition Instances

```
agreement_finalizing’reset_service_offers 1
disagreeing’calculate_emptying 1
disagreeing’empty_distribution 1
disagreeing’prepare_partners_for_re 1
disagreeing’re 1
disagreement_finalizing’remove_channels 1
disagreement_finalizing’remove_gathered_partners 1
disagreement_finalizing’remove_reset_service_offers 1
disagreement_finalizing’remove_unreset_service_offers 1
disagreement_finalizing’set_false_outcome 1
negotiate’disagree 1
negotiate’make_counteroffer 1
```

Live Transition Instances None

C.5 Fairness Properties

Impartial Transition Instances None
Fair Transition Instances

```
agreement_finalizing’check_agreed 1
agreement_finalizing’reset_service_offers 1
agreement_finalizing’set_true_outcome 1
agreement_finalizing’withdraw_extracted_partners 1
contract_extraction’extract_eContract_proposal 1
contract_extraction’merge_specs 1
contract_extraction’pick_eCommunity 1
disagreeing’calculate_emptying 1
disagreeing’empty_distribution 1
disagreeing’prepare_partners_for_re 1
disagreeing’re 1
disagreement_finalizing’remove_channels 1
disagreement_finalizing’remove_gathered_partners 1
disagreement_finalizing’remove_reset_service_offers 1
disagreement_finalizing’remove_unreset_service_offers 1
disagreement_finalizing’set_false_outcome 1
negotiate’disagree 1
negotiate’make_counteroffer 1
```

Just Transition Instances None
Transition Instances with No Fairness

```
contract_extraction'extract_partners 1
negotiate'agree 1
negotiate'distribute_eContract_to_partners 1
```

Appendix D

negotiate with forced counteroffer

Contents

D.1 Statistics 117
D.2 Boundedness Properties 117
D.3 Home Properties 123
D.4 Liveness Properties 123
D.5 Fairness Properties 124

D.1 Statistics

```
   State Space
     Nodes:   6010
     Arcs:    15953
     Secs:    6
     Status:  Full

  Scc Graph
     Nodes:   1246
     Arcs:    3549
     Secs:    0
```

D.2 Boundedness Properties

Best Integer Bounds

```
                              Upper       Lower
     agreement_finalizing'agreed 1
                              0           0
     agreement_finalizing'all_extracted 1
                              0           0
     agreement_finalizing'enable_offers_reset 1
```

```
                             0            0
agreement_finalizing'reset_counter 1
                             1            1
contract_extraction'all_partners_extracted 1
                             1            0
contract_extraction'channel_counter 1
                             1            0
contract_extraction'channels_merged 1
                             1            0
contract_extraction'chosen 1
                             1            0
contract_extraction'merged_channel_spec 1
                             1            1
disagreeing'enable_partner_preparation 1
                             1            0
disagreeing'enable_re 1 1             0
disagreeing'enable_termination 1
                             0            0
disagreeing'need_to_merge 1
                             1            0
disagreeing'partner_preparation_counter 1
                             1            1
disagreeing'remaining_distributed_eContracts 1
                             1            0
disagreement_finalizing'enable_channel_removal 1
                             0            0
disagreement_finalizing'enable_service_offer_removal 1
                             0            0
disagreement_finalizing'partner_removal_counter 1
                             0            0
negotiate'agreed_count 1
                             1            1
negotiate'canceled_agreement 1
                             1            0
negotiate'channel_number 1
                             1            1
negotiate'channels_tentatively_established 1
                             2            2
negotiate'disable 1          0            0
negotiate'distributed_eContract_proposals 1
                             3            0
negotiate'distribution_count 1
                             1            1
negotiate'distribution_number 1
                             0            0
negotiate'eContract_outcome 1
                             0            0
negotiate'eContract_proposal 1
                             1            1
negotiate'enable_closure 1
                             0            0
negotiate'enable_eCommunity_pick 1
                             1            0
negotiate'enable_set_false_outcome 1
```

```
                                0            0
     negotiate'enable_withdrawal 1
                                0            0
     negotiate'extracted_eContract_proposal 1
                                1            0
     negotiate'gathered_partners 1
                                0            0
     negotiate'partner_count 1
                                1            0
     negotiate'partners_gathered 1
                                3            0
     negotiate'permit_agree 1
                                1            0
     negotiate'permit_counteroffer 1
                                1            0
     negotiate'potential_partners 1
                                6            6
     negotiate'roles_number 1
                                1            1
     negotiate'service_offers_assigned 1
                                3            3
```

Best Upper Multi-set Bounds

```
     agreement_finalizing'agreed 1
                             empty
     agreement_finalizing'all_extracted 1
                             empty
     agreement_finalizing'enable_offers_reset 1
                             empty
     agreement_finalizing'reset_counter 1
                             1`1
     contract_extraction'all_partners_extracted 1
                             1`1
     contract_extraction'channel_counter 1
                             1`0++
1`1
     contract_extraction'channels_merged 1
                             1`1
     contract_extraction'chosen 1
                             1`1
     contract_extraction'merged_channel_spec 1
                             1`""++
1`"SPEC"
     disagreeing'enable_partner_preparation 1
                             1`(1,1)
     disagreeing'enable_re 1
                             1`1
     disagreeing'enable_termination 1
                             empty
     disagreeing'need_to_merge 1
                             1`(1,1)
     disagreeing'partner_preparation_counter 1
                             1`0++
1‘1++
1‘2
          disagreeing'remaining_distributed_eContracts 1
                                   1‘(1,1,1,"counter")++
1‘(1,1,2,"counter")++
1‘(1,1,3,"counter")
          disagreement_finalizing'enable_channel_removal 1
                                   empty
          disagreement_finalizing'enable_service_offer_removal 1
                                   empty
          disagreement_finalizing'partner_removal_counter 1
                                   empty
          negotiate'agreed_count 1
                                   1‘0
          negotiate'canceled_agreement 1
                                   1‘(1,1,"counter")
          negotiate'channel_number 1
                                   1‘(1,2)
          negotiate'channels_tentatively_established 1
                                   1‘(1,1,2,0)++
1‘(1,1,3,1)
          negotiate'disable 1 empty
          negotiate'distributed_eContract_proposals 1
                                   3‘(1,1,1)++
3‘(1,1,2)++
3‘(1,1,3)++
3‘(1,1,4)++
3‘(1,1,5)++
3‘(1,1,6)
          negotiate'distribution_count 1
                                   1‘0++
1‘1++
1‘2++
1‘3
          negotiate'distribution_number 1
                                   empty
          negotiate'eContract_outcome 1
                                   empty
          negotiate'eContract_proposal 1
                                   1‘(1,1,false)++
1‘(1,1,true)
          negotiate'enable_closure 1
                                   empty
          negotiate'enable_eCommunity_pick 1
                                   1‘1
          negotiate'enable_set_false_outcome 1
                                   empty
          negotiate'enable_withdrawal 1
                                   empty
          negotiate'extracted_eContract_proposal 1
                                   1‘(1,1)
          negotiate'gathered_partners 1
                                   empty
          negotiate'partner_count 1
```

```
                                        1‘0++
1‘1++
1‘2++
1‘3
     negotiate’partners_gathered 1
                                        1‘(1,1,1,1,false)++
1‘(1,1,1,1,true)++
1‘(1,1,2,1,false)++
1‘(1,1,2,1,true)++
1‘(1,1,3,1,false)++
1‘(1,1,3,1,true)++
1‘(1,1,4,1,false)++
1‘(1,1,4,1,true)++
1‘(1,1,5,1,false)++
1‘(1,1,5,1,true)++
1‘(1,1,6,1,false)++
1‘(1,1,6,1,true)++
1‘(1,2,1,2,false)++
1‘(1,2,1,2,true)++
1‘(1,2,2,2,false)++
1‘(1,2,2,2,true)++
1‘(1,2,3,2,false)++
1‘(1,2,3,2,true)++
1‘(1,2,4,2,false)++
1‘(1,2,4,2,true)++
1‘(1,2,5,2,false)++
1‘(1,2,5,2,true)++
1‘(1,2,6,2,false)++
1‘(1,2,6,2,true)++
1‘(1,3,1,3,false)++
1‘(1,3,1,3,true)++
1‘(1,3,2,3,false)++
1‘(1,3,2,3,true)++
1‘(1,3,3,3,false)++
1‘(1,3,3,3,true)++
1‘(1,3,4,3,false)++
1‘(1,3,4,3,true)++
1‘(1,3,5,3,false)++
1‘(1,3,5,3,true)++
1‘(1,3,6,3,false)++
1‘(1,3,6,3,true)
     negotiate’permit_agree 1
                                        1‘1
     negotiate’permit_counteroffer 1
                                        1‘1
     negotiate’potential_partners 1
                                        1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     negotiate’roles_number 1
                                        1‘(1,3)
     negotiate'service_offers_assigned 1
                           1`(1,1,1,false)++
1`(1,1,1,true)++
1`(1,2,2,false)++
1`(1,2,2,true)++
1`(1,3,3,false)++
1`(1,3,3,true)
```

Best Lower Multi-set Bounds

```
     agreement_finalizing'agreed 1
                           empty
     agreement_finalizing'all_extracted 1
                           empty
     agreement_finalizing'enable_offers_reset 1
                           empty
     agreement_finalizing'reset_counter 1
                           1`1
     contract_extraction'all_partners_extracted 1
                           empty
     contract_extraction'channel_counter 1
                           empty
     contract_extraction'channels_merged 1
                           empty
     contract_extraction'chosen 1
                           empty
     contract_extraction'merged_channel_spec 1
                           empty
     disagreeing'enable_partner_preparation 1
                           empty
     disagreeing'enable_re 1
                           empty
     disagreeing'enable_termination 1
                           empty
     disagreeing'need_to_merge 1
                           empty
     disagreeing'partner_preparation_counter 1
                           empty
     disagreeing'remaining_distributed_eContracts 1
                           empty
     disagreement_finalizing'enable_channel_removal 1
                           empty
     disagreement_finalizing'enable_service_offer_removal 1
                           empty
     disagreement_finalizing'partner_removal_counter 1
                           empty
     negotiate'agreed_count 1
                           1`0
     negotiate'canceled_agreement 1
                           empty
     negotiate'channel_number 1
                           1`(1,2)
     negotiate'channels_tentatively_established 1
                           1`(1,1,2,0)++
1 ‘(1 ,1 ,3 ,1)
      negotiate ’disable 1 empty
      negotiate ’distributed_eContract_proposals 1
                              empty
      negotiate ’distribution_count 1
                              empty
      negotiate ’distribution_number 1
                              empty
      negotiate ’eContract_outcome 1
                              empty
      negotiate ’eContract_proposal 1
                              empty
      negotiate ’enable_closure 1
                              empty
      negotiate ’enable_eCommunity_pick 1
                              empty
      negotiate ’enable_set_false_outcome 1
                              empty
      negotiate ’enable_withdrawal 1
                              empty
      negotiate ’extracted_eContract_proposal 1
                              empty
      negotiate ’gathered_partners 1
                              empty
      negotiate ’partner_count 1
                              empty
      negotiate ’partners_gathered 1
                              empty
      negotiate ’permit_agree 1
                              empty
      negotiate ’permit_counteroffer 1
                              empty
      negotiate ’potential_partners 1
                              1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1 ‘6
      negotiate ’roles_number 1
                              1 ‘(1 ,3)
      negotiate ’service_offers_assigned 1
                              empty
```

D.3 Home Properties

Home Markings None

D.4 Liveness Properties

Dead Markings None

Dead Transition Instances

```
agreement_finalizing'reset_service_offers 1
agreement_finalizing'set_true_outcome 1
agreement_finalizing'withdraw_extracted_partners 1
disagreement_finalizing'remove_channels 1
disagreement_finalizing'remove_gathered_partners 1
disagreement_finalizing'remove_reset_service_offers 1
disagreement_finalizing'remove_unreset_service_offers 1
disagreement_finalizing'set_false_outcome 1
negotiate'agree 1
negotiate'disagree 1
```

Live Transition Instances

```
agreement_finalizing'check_agreed 1
disagreeing'calculate_emptying 1
disagreeing'empty_distribution 1
disagreeing'prepare_partners_for_re 1
disagreeing're 1
negotiate'distribute_eContract_to_partners 1
negotiate'make_counteroffer 1
```

D.5 Fairness Properties

Impartial Transition Instances None
Fair Transition Instances

```
agreement_finalizing'reset_service_offers 1
agreement_finalizing'set_true_outcome 1
agreement_finalizing'withdraw_extracted_partners 1
contract_extraction'extract_eContract_proposal 1
contract_extraction'merge_specs 1
contract_extraction'pick_eCommunity 1
disagreeing'prepare_partners_for_re 1
disagreement_finalizing'remove_channels 1
disagreement_finalizing'remove_gathered_partners 1
disagreement_finalizing'remove_reset_service_offers 1
disagreement_finalizing'remove_unreset_service_offers 1
disagreement_finalizing'set_false_outcome 1
negotiate'agree 1
negotiate'disagree 1
```

Just Transition Instances

```
agreement_finalizing'check_agreed 1
```

Transition Instances with No Fairness

```
contract_extraction'extract_partners 1
disagreeing'calculate_emptying 1
disagreeing'empty_distribution 1
disagreeing're 1
negotiate'distribute_eContract_to_partners 1
negotiate'make_counteroffer 1
```

Appendix E

negotiate with forced disagree

Contents

E.1 Statistics **125**
E.2 Boundedness Properties **125**
E.3 Home Properties **132**
E.4 Liveness Properties **132**
E.5 Fairness Properties **132**

E.1 Statistics

```
State Space
    Nodes:   28822
    Arcs:    110789
    Secs:    247
    Status:  Full

Scc Graph
    Nodes:   28822
    Arcs:    88539
    Secs:    2
```

E.2 Boundedness Properties

Best Integer Bounds

```
                              Upper        Lower
    agreement_finalizing'agreed 1
                              0            0
    agreement_finalizing'all_extracted 1
                              0            0
    agreement_finalizing'enable_offers_reset 1
                              1            0
    agreement_finalizing'reset_counter 1
```

```
                       1          1
contract_extraction'all_partners_extracted 1
                       1          0
contract_extraction'channel_counter 1
                       1          0
contract_extraction'channels_merged 1
                       1          0
contract_extraction'chosen 1
                       1          0
contract_extraction'merged_channel_spec 1
                       1          1
disagreeing'enable_partner_preparation 1
                       0          0
disagreeing'enable_re 1 0          0
disagreeing'enable_termination 1
                       1          0
disagreeing'need_to_merge 1
                       0          0
disagreeing'partner_preparation_counter 1
                       1          1
disagreeing'remaining_distributed_eContracts 1
                       1          0
disagreement_finalizing'enable_channel_removal 1
                       1          0
disagreement_finalizing'enable_service_offer_removal 1
                       1          0
disagreement_finalizing'partner_removal_counter 1
                       1          0
negotiate'agreed_count 1
                       1          1
negotiate'canceled_agreement 1
                       1          0
negotiate'channel_number 1
                       1          0
negotiate'channels_tentatively_established 1
                       2          0
negotiate'disable 1    0          0
negotiate'distributed_eContract_proposals 1
                       3          0
negotiate'distribution_count 1
                       1          1
negotiate'distribution_number 1
                       0          0
negotiate'eContract_outcome 1
                       1          0
negotiate'eContract_proposal 1
                       1          0
negotiate'enable_closure 1
                       1          0
negotiate'enable_eCommunity_pick 1
                       1          0
negotiate'enable_set_false_outcome 1
                       1          0
negotiate'enable_withdrawal 1
```

```
                                 1            0
      negotiate'extracted_eContract_proposal 1
                                 1            0
      negotiate'gathered_partners 1
                                 3            0
      negotiate'partner_count 1
                                 1            0
      negotiate'partners_gathered 1
                                 3            0
      negotiate'permit_agree 1
                                 1            0
      negotiate'permit_counteroffer 1
                                 1            0
      negotiate'potential_partners 1
                                 6            6
      negotiate'roles_number 1
                                 1            0
      negotiate'service_offers_assigned 1
                                 3            0
```

Best Upper Multi-set Bounds

```
      agreement_finalizing'agreed 1
                           empty
      agreement_finalizing'all_extracted 1
                           empty
      agreement_finalizing'enable_offers_reset 1
                           1`1
      agreement_finalizing'reset_counter 1
                           1`1++
1`2++
1`3
      contract_extraction'all_partners_extracted 1
                           1`1
      contract_extraction'channel_counter 1
                           1`0++
1`1
      contract_extraction'channels_merged 1
                           1`1
      contract_extraction'chosen 1
                           1`1
      contract_extraction'merged_channel_spec 1
                           1`""++
1`"SPEC"
      disagreeing'enable_partner_preparation 1
                           empty
      disagreeing'enable_re 1
                           empty
      disagreeing'enable_termination 1
                           1`(1,1)
      disagreeing'need_to_merge 1
                           empty
      disagreeing'partner_preparation_counter 1
                           1`0
```

```
     disagreeing'remaining_distributed_eContracts 1
                         1`(1,1,1,"disagree")++
1`(1,1,2,"disagree")++
1`(1,1,3,"disagree")
     disagreement_finalizing'enable_channel_removal 1
                         1`1
     disagreement_finalizing'enable_service_offer_removal 1
                         1`1
     disagreement_finalizing'partner_removal_counter 1
                         1`1++
1`2++
1`3
     negotiate'agreed_count 1
                         1`0
     negotiate'canceled_agreement 1
                         1`(1,1,"disagree")
     negotiate'channel_number 1
                         1`(1,1)++
1`(1,2)
     negotiate'channels_tentatively_established 1
                         1`(1,1,2,0)++
1`(1,1,3,1)
     negotiate'disable 1 empty
     negotiate'distributed_eContract_proposals 1
                         3`(1,1,1)++
3`(1,1,2)++
3`(1,1,3)++
3`(1,1,4)++
3`(1,1,5)++
3`(1,1,6)
     negotiate'distribution_count 1
                         1`0++
1`1++
1`2++
1`3
     negotiate'distribution_number 1
                         empty
     negotiate'eContract_outcome 1
                         1`(1,1,false,false)
     negotiate'eContract_proposal 1
                         1`(1,1,false)++
1`(1,1,true)
     negotiate'enable_closure 1
                         1`1
     negotiate'enable_eCommunity_pick 1
                         1`1
     negotiate'enable_set_false_outcome 1
                         1`1
     negotiate'enable_withdrawal 1
                         1`(1,"disagree")++
1`(2,"disagree")++
1`(3,"disagree")
     negotiate'extracted_eContract_proposal 1
                         1`(1,1)
     negotiate'gathered_partners 1
                         1`(1,1,1,false)++
1`(1,1,2,false)++
1`(1,1,3,false)++
1`(1,2,1,false)++
1`(1,2,2,false)++
1`(1,2,3,false)++
1`(1,3,1,false)++
1`(1,3,2,false)++
1`(1,3,3,false)++
1`(1,4,1,false)++
1`(1,4,2,false)++
1`(1,4,3,false)++
1`(1,5,1,false)++
1`(1,5,2,false)++
1`(1,5,3,false)++
1`(1,6,1,false)++
1`(1,6,2,false)++
1`(1,6,3,false)
     negotiate'partner_count 1
                         1`0++
1`1++
1`2++
1`3
     negotiate'partners_gathered 1
                         1`(1,1,1,1,false)++
1`(1,1,1,1,true)++
1`(1,1,2,1,false)++
1`(1,1,2,1,true)++
1`(1,1,3,1,false)++
1`(1,1,3,1,true)++
1`(1,1,4,1,false)++
1`(1,1,4,1,true)++
1`(1,1,5,1,false)++
1`(1,1,5,1,true)++
1`(1,1,6,1,false)++
1`(1,1,6,1,true)++
1`(1,2,1,2,false)++
1`(1,2,1,2,true)++
1`(1,2,2,2,false)++
1`(1,2,2,2,true)++
1`(1,2,3,2,false)++
1`(1,2,3,2,true)++
1`(1,2,4,2,false)++
1`(1,2,4,2,true)++
1`(1,2,5,2,false)++
1`(1,2,5,2,true)++
1`(1,2,6,2,false)++
1`(1,2,6,2,true)++
1`(1,3,1,3,false)++
1`(1,3,1,3,true)++
1`(1,3,2,3,false)++
1`(1,3,2,3,true)++
1`(1,3,3,3,false)++
1‘(1,3,3,3,true)++
1‘(1,3,4,3,false)++
1‘(1,3,4,3,true)++
1‘(1,3,5,3,false)++
1‘(1,3,5,3,true)++
1‘(1,3,6,3,false)++
1‘(1,3,6,3,true)
     negotiate'permit_agree 1
                         1‘1
     negotiate'permit_counteroffer 1
                         1‘1
     negotiate'potential_partners 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     negotiate'roles_number 1
                         1‘(1,1)++
1‘(1,2)++
1‘(1,3)
     negotiate'service_offers_assigned 1
                         1‘(1,1,1,false)++
1‘(1,1,1,true)++
1‘(1,2,2,false)++
1‘(1,2,2,true)++
1‘(1,3,3,false)++
1‘(1,3,3,true)
```

Best Lower Multi-set Bounds

```
     agreement_finalizing'agreed 1
                         empty
     agreement_finalizing'all_extracted 1
                         empty
     agreement_finalizing'enable_offers_reset 1
                         empty
     agreement_finalizing'reset_counter 1
                         empty
     contract_extraction'all_partners_extracted 1
                         empty
     contract_extraction'channel_counter 1
                         empty
     contract_extraction'channels_merged 1
                         empty
     contract_extraction'chosen 1
                         empty
     contract_extraction'merged_channel_spec 1
                         empty
     disagreeing'enable_partner_preparation 1
                         empty
     disagreeing'enable_re 1
                         empty
```

```
disagreeing'enable_termination 1
                        empty
disagreeing'need_to_merge 1
                        empty
disagreeing'partner_preparation_counter 1
                        1`0
disagreeing'remaining_distributed_eContracts 1
                        empty
disagreement_finalizing'enable_channel_removal 1
                        empty
disagreement_finalizing'enable_service_offer_removal 1
                        empty
disagreement_finalizing'partner_removal_counter 1
                        empty
negotiate'agreed_count 1
                        1`0
negotiate'canceled_agreement 1
                        empty
negotiate'channel_number 1
                        empty
negotiate'channels_tentatively_established 1
                        empty
negotiate'disable 1 empty
negotiate'distributed_eContract_proposals 1
                        empty
negotiate'distribution_count 1
                        empty
negotiate'distribution_number 1
                        empty
negotiate'eContract_outcome 1
                        empty
negotiate'eContract_proposal 1
                        empty
negotiate'enable_closure 1
                        empty
negotiate'enable_eCommunity_pick 1
                        empty
negotiate'enable_set_false_outcome 1
                        empty
negotiate'enable_withdrawal 1
                        empty
negotiate'extracted_eContract_proposal 1
                        empty
negotiate'gathered_partners 1
                        empty
negotiate'partner_count 1
                        empty
negotiate'partners_gathered 1
                        empty
negotiate'permit_agree 1
                        empty
negotiate'permit_counteroffer 1
                        empty
negotiate'potential_partners 1
```

```
                       1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     negotiate’roles_number 1
                       empty
     negotiate’service_offers_assigned 1
                       empty
```

E.3 Home Properties

Home Markings None

E.4 Liveness Properties

Dead Markings [28806,28816,28821,28822]
Dead Transition Instances

```
      agreement_finalizing’set_true_outcome 1
     disagreeing’prepare_partners_for_re 1
     disagreeing’re 1
     negotiate’agree 1
     negotiate’make_counteroffer 1
```

Live Transition Instances None

E.5 Fairness Properties

Impartial Transition Instances None
Fair Transition Instances

```
     agreement_finalizing’check_agreed 1
     agreement_finalizing’set_true_outcome 1
     contract_extraction’extract_eContract_proposal 1
     contract_extraction’merge_specs 1
     contract_extraction’pick_eCommunity 1
     disagreeing’prepare_partners_for_re 1
     disagreeing’re 1
     disagreement_finalizing’remove_channels 1
     disagreement_finalizing’remove_gathered_partners 1
     disagreement_finalizing’remove_reset_service_offers 1
     disagreement_finalizing’remove_unreset_service_offers 1
     disagreement_finalizing’set_false_outcome 1
     negotiate’agree 1
     negotiate’make_counteroffer 1
```

Just Transition Instances None
Transition Instances with No Fairness

```
     agreement_finalizing’reset_service_offers 1
     agreement_finalizing’withdraw_extracted_partners 1
     contract_extraction’extract_partners 1
```

```
disagreeing'calculate_emptying 1
disagreeing'empty_distribution 1
negotiate'disagree 1
negotiate'distribute_eContract_to_partners 1
```

Appendix F

governance distribution and extraction

Contents

F.1 Statistics 135
F.2 Boundedness Properties 135
F.3 Home Properties 140
F.4 Liveness Properties 140
F.5 Fairness Properties 140

F.1 Statistics

```
State Space
   Nodes:   3656
   Arcs:    11885
   Secs:    12
   Status:  Full

Scc Graph
   Nodes:   3656
   Arcs:    11885
   Secs:    0
```

F.2 Boundedness Properties

Best Integer Bounds

```
                          Upper       Lower
   extraction'A 1         1           0
   extraction'B 1         1           0
   extraction'enable_policy_extraction 1
                          1           0
   extraction'picked_extraction 1
                          1           0
```

```
extraction'policy_repository 1
                           6            6
governance_distribution'A 1
                           1            0
governance_distribution'assigned_BNMAs 1
                           3            0
governance_distribution'assigned_monitors 1
                           3            0
governance_distribution'available_local_contracts 1
                           1            1
governance_distribution'chosen_eContract 1
                           1            0
governance_distribution'contract_partner_counter 1
                           1            0
governance_distribution'disable 1
                           0            0
governance_distribution'distribution_number 1
                           1            0
governance_distribution'eContract_outcome 1
                           1            1
governance_distribution'enable_enactment_preparation 1
                           1            0
governance_distribution'enabled_extraction 1
                           1            0
governance_distribution'error_options 1
                           2            2
governance_distribution'extract_counter 1
                           1            1
governance_distribution'gathered_partners 1
                           3            3
governance_distribution'lack_of_synchrony_error 1
                           0            0
governance_distribution'local_contract_counter 1
                           1            0
governance_distribution'local_contracts 1
                           3            0
governance_distribution'partner_policy_counter 1
                           3            0
governance_distribution'policies 1
                           4            0
governance_distribution'policy_pool 1
                           1            1
governance_distribution'pool_of_BNMAs 1
                           6            6
governance_distribution'pool_of_errors 1
                           6            6
governance_distribution'pool_of_monitors 1
                           1            1
```

Best Upper Multi-set Bounds

```
extraction'A 1         1`1
extraction'B 1         1`1
extraction'enable_policy_extraction 1
```

```
                        1`(1,1,1)++
1`(1,2,2)++
1`(1,3,1)
     extraction'picked_extraction 1
                        1`(1,1,1)++
1`(1,2,1)++
1`(1,2,2)++
1`(1,3,1)
     extraction'policy_repository 1
                        1`(1,1)++
1`(2,2)++
1`(3,1)++
1`(4,1)++
1`(5,0)++
1`(6,4)
     governance_distribution'A 1
                        1`1
     governance_distribution'assigned_BNMAs 1
                        1`(1,1,1,1)++
1`(1,1,1,2)++
1`(1,1,1,3)++
1`(1,1,1,4)++
1`(1,1,1,5)++
1`(1,1,1,6)++
1`(1,2,1,1)++
1`(1,2,1,2)++
1`(1,2,1,3)++
1`(1,2,1,4)++
1`(1,2,1,5)++
1`(1,2,1,6)++
1`(1,3,1,1)++
1`(1,3,1,2)++
1`(1,3,1,3)++
1`(1,3,1,4)++
1`(1,3,1,5)++
1`(1,3,1,6)
     governance_distribution'assigned_monitors 1
                        1`(1,1,1,1)++
1`(1,2,1,1)++
1`(1,3,1,1)
     governance_distribution'available_local_contracts 1
                        1`1
     governance_distribution'chosen_eContract 1
                        1`(1,1)
     governance_distribution'contract_partner_counter 1
                        1`(1,3)
     governance_distribution'disable 1
                        empty
     governance_distribution'distribution_number 1
                        1`(1,1,false)++
1`(1,2,false)++
1`(1,3,false)
     governance_distribution'eContract_outcome 1
                        1`(1,1,true,false)++
1`(1,1,true,true)
     governance_distribution'enable_enactment_preparation 1
                         1`(1,false)
     governance_distribution'enabled_extraction 1
                         1`1
     governance_distribution'error_options 1
                         1`false++
1`true
     governance_distribution'extract_counter 1
                         1`0++
1`1++
1`2++
1`3
     governance_distribution'gathered_partners 1
                         1`(1,1,1,false)++
1`(1,1,1,true)++
1`(1,2,2,false)++
1`(1,2,2,true)++
1`(1,3,3,false)++
1`(1,3,3,true)
     governance_distribution'lack_of_synchrony_error 1
                         empty
     governance_distribution'local_contract_counter 1
                         1`(1,0,false)++
1`(1,1,false)++
1`(1,2,false)++
1`(1,3,true)
     governance_distribution'local_contracts 1
                         1`(1,1,1,1,false,false)++
1`(1,1,1,1,false,true)++
1`(1,1,1,1,true,true)++
1`(1,1,2,2,false,false)++
1`(1,1,2,2,false,true)++
1`(1,1,2,2,true,true)++
1`(1,1,3,3,false,false)++
1`(1,1,3,3,false,true)++
1`(1,1,3,3,true,true)
     governance_distribution'partner_policy_counter 1
                         1`(1,1,0)++
1`(1,1,1)++
1`(1,2,0)++
1`(1,2,1)++
1`(1,2,2)++
1`(1,3,0)++
1`(1,3,1)
     governance_distribution'policies 1
                         1`(1,1,1)++
2`(1,2,1)++
1`(1,3,1)
     governance_distribution'policy_pool 1
                         1`1
     governance_distribution'pool_of_BNMAs 1
                         1`1++
1`2++
1‘3++
1‘4++
1‘5++
1‘6
      governance_distribution’pool_of_errors 1
                              1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
      governance_distribution’pool_of_monitors 1
                              1‘1
```

Best Lower Multi-set Bounds

```
      extraction’A 1          empty
      extraction’B 1          empty
      extraction’enable_policy_extraction 1
                              empty
      extraction’picked_extraction 1
                              empty
      extraction’policy_repository 1
                              1‘(1,1)++
1‘(2,2)++
1‘(3,1)++
1‘(4,1)++
1‘(5,0)++
1‘(6,4)
      governance_distribution’A 1
                              empty
      governance_distribution’assigned_BNMAs 1
                              empty
      governance_distribution’assigned_monitors 1
                              empty
      governance_distribution’available_local_contracts 1
                              1‘1
      governance_distribution’chosen_eContract 1
                              empty
      governance_distribution’contract_partner_counter 1
                              empty
      governance_distribution’disable 1
                              empty
      governance_distribution’distribution_number 1
                              empty
      governance_distribution’eContract_outcome 1
                              empty
      governance_distribution’enable_enactment_preparation 1
                              empty
      governance_distribution’enabled_extraction 1
                              empty
      governance_distribution’error_options 1
                              1‘false++
1‘true
```

```
      governance_distribution'extract_counter 1
                              empty
      governance_distribution'gathered_partners 1
                              empty
      governance_distribution'lack_of_synchrony_error 1
                              empty
      governance_distribution'local_contract_counter 1
                              empty
      governance_distribution'local_contracts 1
                              empty
      governance_distribution'partner_policy_counter 1
                              empty
      governance_distribution'policies 1
                              empty
      governance_distribution'policy_pool 1
                              1`1
      governance_distribution'pool_of_BNMAs 1
                              1`1++
1`2++
1`3++
1`4++
1`5++
1`6
      governance_distribution'pool_of_errors 1
                              1`1++
1`2++
1`3++
1`4++
1`5++
1`6
      governance_distribution'pool_of_monitors 1
                              1`1
```

F.3 Home Properties

Home Markings None

F.4 Liveness Properties

Dead Markings 216 [3656,3655,3654,3653,3652,...]
Dead Transition Instances

```
      governance_distribution'synchronization_check_local_meta_contract_and_local_co
```

Live Transition Instances None

F.5 Fairness Properties

No infinite occurrence sequences.

Appendix G

prepare

Contents

G.1 Statistics 141
G.2 Boundedness Properties 141
G.3 Home Properties 145
G.4 Liveness Properties 146
G.5 Fairness Properties 146

G.1 Statistics

```
State Space
    Nodes:   11157
    Arcs:    32494
    Secs:    61
    Status:  Full

Scc Graph
    Nodes:   11157
    Arcs:    32494
    Secs:    1
```

G.2 Boundedness Properties

Best Integer Bounds

```
                            Upper        Lower
     preparation_error'assignment_error_number 1
                            6            6
     preparation_error'disabler 1
                            0            0
     preparation_error'lifeness_errors 1
                            6            6
     preparation_error'preparation_error_number 1
```

```
                             6           6
     preparation_error'service_error 1
                             0           0
     preparation_error'service_factory_error 1
                             0           0
     prepare'assigne_services 1
                             3           0
     prepare'communication_channel_counter 1
                             1           0
     prepare'enable_enactment_preparation 1
                             1           0
     prepare'enable_preparation 1
                             1           0
     prepare'enabled_enactment 1
                             1           0
     prepare'enacted_service_counter 1
                             1           0
     prepare'enacting_services 1
                             3           0
     prepare'endpoint 1      1           1
     prepare'endpoints_established 1
                             3           0
     prepare'generic_service_factory 1
                             3           3
     prepare'local_contract_counter 1
                             1           0
     prepare'local_contracts 1
                             3           3
     prepare'ready_services 1
                             3           0
     prepare'unassigned_local_contract 1
                             3           0
```

Best Upper Multi-set Bounds

```
     preparation_error'assignment_error_number 1
                         1`1++
1`2++
1`3++
1`4++
1`5++
1`6
     preparation_error'disabler 1
                         empty
     preparation_error'lifeness_errors 1
                         1`1++
1`2++
1`3++
1`4++
1`5++
1`6
     preparation_error'preparation_error_number 1
                         1`1++
1`2++
1‘3++
1‘4++
1‘5++
1‘6
      preparation_error’service_error 1
                        empty
      preparation_error’service_factory_error 1
                        empty
      prepare’assigne_services 1
                        3‘(1,1,1,1,false)++
3‘(1,1,1,1,true)++
3‘(1,1,1,2,false)++
3‘(1,1,1,2,true)++
3‘(1,1,1,3,false)++
3‘(1,1,1,3,true)++
3‘(1,2,1,1,false)++
3‘(1,2,1,1,true)++
3‘(1,2,1,2,false)++
3‘(1,2,1,2,true)++
3‘(1,2,1,3,false)++
3‘(1,2,1,3,true)++
3‘(1,3,1,1,false)++
3‘(1,3,1,1,true)++
3‘(1,3,1,2,false)++
3‘(1,3,1,2,true)++
3‘(1,3,1,3,false)++
3‘(1,3,1,3,true)
      prepare’communication_channel_counter 1
                        1‘(1,0)++
1‘(1,1)++
1‘(1,2)++
1‘(1,3)
      prepare’enable_enactment_preparation 1
                        1‘(1,false)
      prepare’enable_preparation 1
                        1‘(1,1,false)++
1‘(1,2,false)++
1‘(1,3,false)
      prepare’enabled_enactment 1
                        1‘1
      prepare’enacted_service_counter 1
                        1‘(1,0)++
1‘(1,1)++
1‘(1,2)++
1‘(1,3)
      prepare’enacting_services 1
                        3‘(1,1,1,1)++
3‘(1,1,1,2)++
3‘(1,1,1,3)++
3‘(1,2,1,1)++
3‘(1,2,1,2)++
3‘(1,2,1,3)++
3‘(1,3,1,1)++
3‘(1,3,1,2)++
3‘(1,3,1,3)
     prepare'endpoint 1   1‘1
     prepare'endpoints_established 1
                         3‘(1,1,1,1)++
3‘(1,1,2,1)++
3‘(1,1,3,1)++
3‘(1,2,1,1)++
3‘(1,2,2,1)++
3‘(1,2,3,1)++
3‘(1,3,1,1)++
3‘(1,3,2,1)++
3‘(1,3,3,1)
     prepare'generic_service_factory 1
                         1‘1++
1‘2++
1‘3
     prepare'local_contract_counter 1
                         1‘(1,1,true)++
1‘(1,2,true)++
1‘(1,3,true)
     prepare'local_contracts 1
                         1‘(1,1,1,1,true,true)++
1‘(1,1,2,2,true,true)++
1‘(1,1,3,3,true,true)
     prepare'ready_services 1
                         3‘(1,1,1,1)++
3‘(1,1,1,2)++
3‘(1,1,1,3)++
3‘(1,2,1,1)++
3‘(1,2,1,2)++
3‘(1,2,1,3)++
3‘(1,3,1,1)++
3‘(1,3,1,2)++
3‘(1,3,1,3)
     prepare'unassigned_local_contract 1
                         3‘(1,1,1)++
3‘(1,2,1)++
3‘(1,3,1)
```

Best Lower Multi-set Bounds

```
     preparation_error'assignment_error_number 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     preparation_error'disabler 1
                         empty
     preparation_error'lifeness_errors 1
                         1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     preparation_error’preparation_error_number 1
                    1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
     preparation_error’service_error 1
                    empty
     preparation_error’service_factory_error 1
                    empty
     prepare’assigne_services 1
                    empty
     prepare’communication_channel_counter 1
                    empty
     prepare’enable_enactment_preparation 1
                    empty
     prepare’enable_preparation 1
                    empty
     prepare’enabled_enactment 1
                    empty
     prepare’enacted_service_counter 1
                    empty
     prepare’enacting_services 1
                    empty
     prepare’endpoint 1   1‘1
     prepare’endpoints_established 1
                    empty
     prepare’generic_service_factory 1
                    1‘1++
1‘2++
1‘3
     prepare’local_contract_counter 1
                    empty
     prepare’local_contracts 1
                    1‘(1,1,1,1,true,true)++
1‘(1,1,2,2,true,true)++
1‘(1,1,3,3,true,true)
     prepare’ready_services 1
                    empty
     prepare’unassigned_local_contract 1
                    empty
```

G.3 Home Properties

Home Markings None

G.4 Liveness Properties

Dead Markings 165 [11157,11156,11155,11154,11153,...]
Dead Transition Instances

```
preparation_error'report_assignment_error 1
preparation_error'report_preparation_error 1
preparation_error'report_service_error 1
```

Live Transition Instances None

G.5 Fairness Properties

No infinite occurrence sequences.

Appendix H

enact

Contents

H.1 Statistics 147
H.2 Boundedness Properties 147
H.3 Home Properties 150
H.4 Liveness Properties 150
H.5 Fairness Properties 150

H.1 Statistics

```
State Space
    Nodes:   268
    Arcs:    570
    Secs:    0
    Status:  Full

Scc Graph
    Nodes:   268
    Arcs:    522
    Secs:    0
```

H.2 Boundedness Properties

Best Integer Bounds

```
                                Upper        Lower
    enact'assigned_BNMAs 1      3            3
    enact'assigned_monitors 1
                                3            3
    enact'disabler 1            0            0
    enact'enable_disruptive_partner_change 1
                                1            0
    enact'enable_enactment 1
                                1            0
```

```
     enact'enable_non_disruptive_local_contract_change 1
                         1          0
     enact'enable_non_disruptive_partner_change 1
                         1          0
     enact'enacted_service_counter 1
                         1          1
     enact'enacting_services 1
                         3          0
     enact'limiter 1     4          0
     enact'local_contracts 1 3      3
     enact'policies 1    4          3
     enact'policy_violation 1
                         1          0
     enact'service_start_error 1
                         0          0
     enact'service_violation 1
                         1          0
     enact'start_error_number 1
                         6          6
     enact'stopped_services 1
                         3          0
     enact'terminate_service 1
                         1          0
```

Best Upper Multi-set Bounds

```
     enact'assigned_BNMAs 1
                     1`(1,1,1,1)++
1`(1,2,1,2)++
1`(1,3,1,3)
     enact'assigned_monitors 1
                     1`(1,1,1,1)++
1`(1,2,1,1)++
1`(1,3,1,1)
     enact'disabler 1    empty
     enact'enable_disruptive_partner_change 1
                     1`1
     enact'enable_enactment 1
                     1`1
     enact'enable_non_disruptive_local_contract_change 1
                     1`(1,1)++
1`(1,2)++
1`(1,3)
     enact'enable_non_disruptive_partner_change 1
                     1`(1,1,1,1,true)++
1`(1,2,1,1,true)++
1`(1,3,1,1,true)
     enact'enacted_service_counter 1
                     1`(1,2)++
1`(1,3)
     enact'enacting_services 1
                     1`(1,1,1,1)++
1`(1,2,1,1)++
1`(1,3,1,1)
```

```
          enact'limiter 1       4‘1
          enact'local_contracts 1
                                1‘(1,1,1,1,true,true)++
1‘(1,1,2,2,true,true)++
1‘(1,1,3,3,true,true)
          enact'policies 1      1‘(1,1,1)++
2‘(1,2,1)++
1‘(1,3,1)
          enact'policy_violation 1
                                1‘(1,1,1)++
1‘(1,2,1)++
1‘(1,3,1)
          enact'service_start_error 1
                                empty
          enact'service_violation 1
                                1‘(1,1,1,1)++
1‘(1,2,1,1)++
1‘(1,3,1,1)
          enact'start_error_number 1
                                1‘1++
1‘2++
1‘3++
1‘4++
1‘5++
1‘6
          enact'stopped_services 1
                                1‘(1,1,1,1)++
1‘(1,2,1,1)++
1‘(1,3,1,1)
          enact'terminate_service 1
                                1‘(1,0)++
1‘(1,1)
```

Best Lower Multi-set Bounds

```
          enact'assigned_BNMAs 1
                                1‘(1,1,1,1)++
1‘(1,2,1,2)++
1‘(1,3,1,3)
          enact'assigned_monitors 1
                                1‘(1,1,1,1)++
1‘(1,2,1,1)++
1‘(1,3,1,1)
          enact'disabler 1      empty
          enact'enable_disruptive_partner_change 1
                                empty
          enact'enable_enactment 1
                                empty
          enact'enable_non_disruptive_local_contract_change 1
                                empty
          enact'enable_non_disruptive_partner_change 1
                                empty
          enact'enacted_service_counter 1
                                empty
```

```
     enact'enacting_services 1
                         empty
     enact'limiter 1     empty
     enact'local_contracts 1
                         1`(1,1,1,1,true,true)++
1`(1,1,2,2,true,true)++
1`(1,1,3,3,true,true)
     enact'policies 1    1`(1,2,1)
     enact'policy_violation 1
                         empty
     enact'service_start_error 1
                         empty
     enact'service_violation 1
                         empty
     enact'start_error_number 1
                         1`1++
1`2++
1`3++
1`4++
1`5++
1`6
     enact'stopped_services 1
                         empty
     enact'terminate_service 1
                         empty
```

H.3 Home Properties

Home Markings None

H.4 Liveness Properties

Dead Markings 75 [9,82,81,80,8,...]
Dead Transition Instances

```
     enact'report_start_error 1
```

Live Transition Instances None

H.5 Fairness Properties

```
     enact'disruptive_partner_change 1
                        No Fairness
     enact'enact_service 1   Impartial
     enact'non_disruptive_local_contract_change 1
                        No Fairness
     enact'non_disruptive_partner_change 1
                        No Fairness
     enact'policy_violation 1
                        No Fairness
     enact'report_start_error 1
                        Fair
```

```
enact'start_service 1   No Fairness
enact'stop_service 1    No Fairness
enact'termination 1     No Fairness
```

Appendix I

terminate

Contents

I.1 Statistics 153
I.2 Boundedness Properties 153
I.3 Home Properties 157
I.4 Liveness Properties 157
I.5 Fairness Properties 157

I.1 Statistics

```
State Space
    Nodes:   956
    Arcs:    3079
    Secs:    0
    Status:  Full

Scc Graph
    Nodes:   956
    Arcs:    3079
    Secs:    0
```

I.2 Boundedness Properties

Best Integer Bounds

```
                                Upper        Lower
    governance_removal'enable_channel_removal 1
                                1            0
    policy_removal'enable_pick 1
                                1            0
    policy_removal'picked_partner_counter 1
                                1            0
    roles_removal'enable_terminate 1
                                1            0
```

```
     roles_removal'roles_counter 1
                              1            1
     terminate'assigned_BNMAs 1
                              3            0
     terminate'assigned_monitors 1
                              3            0
     terminate'channel_number 1
                              1            0
     terminate'channels_tentatively_established 1
                              2            0
     terminate'communication_channel_counter 1
                              1            0
     terminate'contract_partner_counter 1
                              1            0
     terminate'eContract_outcome 1
                              1            0
     terminate'eContract_proposal 1
                              1            0
     terminate'enacted_service_counter 1
                              1            0
     terminate'enacting_services 1
                              2            0
     terminate'endpoints_established 1
                              3            0
     terminate'gathered_partners 1
                              3            0
     terminate'local_eContracts 1
                              3            0
     terminate'partner_policy_counter 1
                              3            0
     terminate'partner_reassign_ready 1
                              0            0
     terminate'policies 1     4            0
     terminate'roles 1        1            0
     terminate'roles_number 1
                              1            0
     terminate'service_offers_assigned 1
                              3            0
     terminate'terminate_governance_enabled 1
                              1            0
     terminate'terminate_service 1
                              1            0
     terminate'terminated_eCommunity 1
                              1            0
```

Best Upper Multi-set Bounds

```
     governance_removal'enable_channel_removal 1
                         1`(1,0)
     policy_removal'enable_pick 1
                         1`1
     policy_removal'picked_partner_counter 1
                         1`(1,1)++
1`(1,2)++
1`(1,3)
     roles_removal'enable_terminate 1
                         1`1
     roles_removal'roles_counter 1
                         1`1++
1`2++
1`3
     terminate'assigned_BNMAs 1
                         1`(1,1,1,1)++
1`(1,2,1,2)++
1`(1,3,1,3)
     terminate'assigned_monitors 1
                         1`(1,1,1,1)++
1`(1,2,1,1)++
1`(1,3,1,1)
     terminate'channel_number 1
                         1`(1,1)++
1`(1,2)
     terminate'channels_tentatively_established 1
                         1`(1,1,2,0)++
1`(1,1,3,1)
     terminate'communication_channel_counter 1
                         1`(1,1)++
1`(1,2)++
1`(1,3)
     terminate'contract_partner_counter 1
                         1`(1,1)++
1`(1,2)++
1`(1,3)
     terminate'eContract_outcome 1
                         1`(1,1,true,true)
     terminate'eContract_proposal 1
                         1`(1,1,false)
     terminate'enacted_service_counter 1
                         1`(1,1)++
1`(1,2)
     terminate'enacting_services 1
                         1`(1,2,1,1)++
1`(1,3,1,1)
     terminate'endpoints_established 1
                         1`(1,1,1,1)++
1`(1,2,1,1)++
1`(1,3,1,1)
     terminate'gathered_partners 1
                         1`(1,1,1,true)++
1`(1,2,2,true)++
1`(1,3,3,true)
     terminate'local_eContracts 1
                         1`(1,1,1,1,true,true)++
1`(1,1,2,2,true,true)++
1`(1,1,3,3,true,true)
     terminate'partner_policy_counter 1
                         1`(1,1,1)++
1`(1,2,1)++
1`(1,2,2)++
1`(1,3,1)
     terminate'partner_reassign_ready 1
                         empty
     terminate'policies 1
                         1`(1,1,1)++
2`(1,2,1)++
1`(1,3,1)
     terminate'roles 1   1`(1,0)
     terminate'roles_number 1
                         1`(1,1)++
1`(1,2)++
1`(1,3)
     terminate'service_offers_assigned 1
                         1`(1,1,1,true)++
1`(1,2,2,true)++
1`(1,3,3,true)
     terminate'terminate_governance_enabled 1
                         1`(1,0)
     terminate'terminate_service 1
                         1`(1,0)
     terminate'terminated_eCommunity 1
                         1`1
```

Best Lower Multi-set Bounds

```
     governance_removal'enable_channel_removal 1
                         empty
     policy_removal'enable_pick 1
                         empty
     policy_removal'picked_partner_counter 1
                         empty
     roles_removal'enable_terminate 1
                         empty
     roles_removal'roles_counter 1
                         empty
     terminate'assigned_BNMAs 1
                         empty
     terminate'assigned_monitors 1
                         empty
     terminate'channel_number 1
                         empty
     terminate'channels_tentatively_established 1
                         empty
     terminate'communication_channel_counter 1
                         empty
     terminate'contract_partner_counter 1
                         empty
     terminate'eContract_outcome 1
                         empty
     terminate'eContract_proposal 1
                         empty
     terminate'enacted_service_counter 1
                         empty
```

```
    terminate'enacting_services 1
                            empty
    terminate'endpoints_established 1
                            empty
    terminate'gathered_partners 1
                            empty
    terminate'local_eContracts 1
                            empty
    terminate'partner_policy_counter 1
                            empty
    terminate'partner_reassign_ready 1
                            empty
    terminate'policies 1
                            empty
    terminate'roles 1       empty
    terminate'roles_number 1
                            empty
    terminate'service_offers_assigned 1
                            empty
    terminate'terminate_governance_enabled 1
                            empty
    terminate'terminate_service 1
                            empty
    terminate'terminated_eCommunity 1
                            empty
```

I.3 Home Properties

Home Markings None

I.4 Liveness Properties

Dead Markings 11 [956,946,945,944,910,...]
    Dead Transition Instances None
    Live Transition Instances None

I.5 Fairness Properties

No infinite occurrence sequences.

Bibliography

[1] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guï£¡zar, N. Kartha, C. K. Liu, R. Khalaf, Dieter Koenig, M. Marin, V. Mehta, S. Thatte, D. Rijn, P. Yendluri, and A. Yiu. Web services business process execution language version 2.0 (OASIS standard), 2007. pages 5

[2] Samuil Angelov, Jochem Vonk, Krishnamurthy Vidyasankar, and Paul W. P. J. Grefen. Supporting cross-organizational process control. In Luis M. Camarinha-Matos, Iraklis Paraskakis, and Hamideh Afsarmanesh, editors, *PRO-VE*, volume 307 of *IFIP Conference Proceedings*, pages 415–422. Springer, 2009. pages 81

[3] M. M. Astrahan, Ht. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System r: Relational approach to database management. *ACM Transactions on Database Systems*, 1:97–137, 1976. pages 11

[4] V. Bjork and C.T. Davis. Data processing spheres of controldata, 1978. pages 14

[5] Erik Boertjes, Paul W. P. J. Grefen, Jochem Vonk, and Peter M. G. Apers. An architecture for nested transaction support on standard database systems. In *Proceedings of the 9th International Conference on Database and Expert Systems Applications*, DEXA ’98, pages 448–459, London, UK, 1998. Springer-Verlag. pages 13

[6] D. Box, D. Ehnebuske, and G. Kakivaya et al. *Simple Object Access Protocol (SOAP) 1.1*. http://www.w3.org/TR/SOAP/, 2003. pages 5

[7] Yuri Breitbart, Hector Garcia-Molina, and Avi Silberschatz. Overview of multidatabase transaction management. *The VLDB Journal*, 1:181–240, October 1992. pages 11

[8] D. Bunting, M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenso. *Web Service Context (WS-CTX)*, July 2003. pages 85

[9] D. Bunting, M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenso. *Web Service Coordination Framework (WS-CF)*, July 2003. pages 85

[10] D. Bunting, M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenso. *Web Services Composite Application Framework (WSCAF)*, July 2003. pages 85, 86

[11] D. Bunting, M. Chapman, O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenso. *Web Services Transaction Management (WS-TXM)*, July 2003. pages 85

[12] L.F Cabrera, G. Copeland, M. Feingold, R.W. Freind, and T. Freund. *WS-AtomicTransaction 1.0*. ftp://www6.software.ibm.com/software/developer/library/WS-AtomicTransaction.pdf, 2005. pages 84, 85

[13] L.F Cabrera, G. Copeland, M. Feingold, R.W. Freind, and T. Freund. *WS-BusinessActivity 1.0*. ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity., 2005. pages 84

[14] L.F Cabrera, G. Copeland, M. Feingold, R.W. Freind, and T. Freund. *WS-Coordination 1.0*. ftp://www6.software.ibm.com/software/developer/library/WS-Coordination.pdf, 2005. pages 84, 85

[15] A. Ceponkus, S. Dalal, T. Fletcher, P. Furniss, A. Green, and B. Pope. Business transaction protocol version 1.0, June 2002. pages 84

[16] R. Chinnici, M. Gudgin, J.J. Moreau, and Sanjiva Weerawarana. *Web Services Description Language (WSDL) Version 1.2*. http://www.w3.org/TR/2003/WD-wsdl12-20030611, 2003. pages 5

[17] Panayiotis K. Chrysanthis and Krithi Ramamritham. Acta: a framework for specifying and reasoning about transaction structure and behavior. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, SIGMOD '90, pages 194–203, New York, NY, USA, 1990. ACM. pages 78

[18] Panos K. Chrysanthis and Krithi Ramamritham. Delegation in acta to control sharing in extended transactions. *IEEE Data Eng. Bull.* pages 78

[19] Panos K. Chrysanthis and Krithi Ramamritham. *ACTA: the SAGA continues*, pages 349–397. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992. pages 13, 78

[20] Panos K. Chrysanthis and Krithi Ramamritham. Synthesis of extended transaction models using acta. *ACM Trans. Database Syst.*, 19:450–491, September 1994. pages 78

[21] Sanjay Dalal, Sazi Temel, Mark Little, Mark Potts, and Jim Webber. Coordinating business transactions on the web. *IEEE Internet Computing*, 7:30–39, January 2003. pages 84

[22] T. Freund and T. Story. Transactions in the world of web services, part 1., 2002. pages 85

[23] H. Garcia-Molina and K. Salem. Sagas. In *In: Procs. of the 1987 ACM SIGMOD International Conference on Management of data (SIGMOD'87)*, pages 249–259, San Francisco, California, United States, 1987. pages 13, 66, 79

[24] J. Gray and A. Reuter. *Transaction Processing : Concepts and Techniques*. Morgan Caufmann, San Francisco, 1993. pages 11

[25] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993. pages 12, 79

[26] P. Grefen. Service-oriented support for dynamic interorganizational business process management. In M. Papazoglou D. Georgakopoulos, editor, *Service Oriented Computing*, pages 83–110. MIT Press, 2008. pages 4

[27] P. Grefen, H. Ludwig, and S. Angelov. A Three-Level Framework for Process and Data Management of Complex E-Services. *International Journal of Cooperative Information Systems*, 12(4):487–531, 2003. pages vii, 3, 17

[28] Paul Grefen, Karl Aberer, Heiko Ludwig, and Yigal Hoffner. Crossflow: Cross-organizational workflow management in dynamic virtual enterprises. *International Journal of Computer Systems Science & Engineering*, 15:277–290, 2000. pages 81

[29] Paul Grefen, Jochem Vonk, and Peter Apers. Global transaction support for workflow management systems: from formal specification to practical implementation. *The VLDB Journal*, 10:316–333, December 2001. pages 13

[30] Paul W. P. J. Grefen, Jochem Vonk, Erik Boertjes, and Peter M. G. Apers. Two-layer transaction management for workflow management applications. In *Proceedings of the 8th International Conference on Database and Expert Systems Applications*, DEXA '97, pages 430–439, London, UK, 1997. Springer-Verlag. pages 13, 84

[31] P.W.P.J. Grefen, H. Ludwig, and S.A. Angelov. A framework for e-services: A three-level approach towards process and data management, 2002. pages 14

[32] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15:287–317, December 1983. pages 79

[33] A.R. Hevner, S. Ram, S.T. March, and J. Park. DESIGN SCIENCE IN INFORMATION SYSTEM RESEARCH. *MIS Quarterly*, 28(1):75–105, 2004. pages vii, 5, 6

[34] Willem jan Van Den Heuvel and Sergei Artyshchev. Developing a three-dimensional transaction model for supporting atomicity spheres. In *International Workshop on Web Services Research, Standardization, and Deployment*, 2002. pages 14

[35] Kurt Jensen, Lars Michael, Kristensen Lisa Wells, K. Jensen, and L. M. Kristensen. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. In *International Journal on Software Tools for Technology Transfer*, page 2007, 2007. pages 24

[36] Tao Jin and Steve Goschnick. Utilizing web services in an agent based transaction model (abt. In *In AAMAS 2003 - Workshop on Web Services and Agent-based Engineering*, pages 1–9, 2003. pages 86

[37] L. Kutvonen, J. Metso, and T. Ruokolainen. Inter-enterprise Collaboration Management in Dynamic Business Networks. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, volume 3760 of *Lecture Notes in Computer Science*, page 593ï£¡611, Agia Napa, Cyprus, October 2005. LNCS Springer. pages 25

[38] Lea Kutvonen, Janne Metso, and Sini Ruohomaa. From trading to ecommunity population: Responding to social and contractual challenges. In *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 199–210, Washington, DC, USA, 2006. IEEE Computer Society. pages 15

[39] Philip M. Lewis, Arthur Bernstein, and Michael Kifer. Databases and transaction processing: an application-oriented approach. *SIGMOD Rec.*, 31:74–75, March 2002. pages 11

[40] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall, N.J., 2000. pages 14

[41] M. Little. Business transaction protocol: Transactions for a new age. *Web Services Journal*, 11:50–55, 2002. pages 84

[42] M. Little and T.S. Freund. A comparison of web services transaction protocols. Technical report, http://www-106.ibm.com/developerworks/webservices/library/ws-comproto/y, 2003. pages 2, 13

[43] M. Little and J. Webber. Introducing WS-CAF - more than just transactions. *Web Services Journal*, 3(12):52–55, 2003. pages 85, 86

[44] N. Mehandjiev and P. Grefen, editors. *Dynamic Business Process Formation for Instant Virtual Enterprises*. Springer, 2010. pages 2, 9

[45] Jan Mendling. *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*. Springer Publishing Company, Incorporated, 1 edition, 2008. pages 80

[46] E. B. Moss. Nested transactions: An approach to reliable distributed computing. Technical report, Cambridge, MA, USA, 1981. pages 11

[47] N.Antonopoulos and L. Gillam, editors. *Cloud Computing: Principles, Systems and Application*. Springer, 2010. pages 4

[48] A. Norta. *Exploring Dynamic Inter-Organizational Business Process Collaboration*. PhD thesis, Technology University Eindhoven, Department of Information Systems, 2007. pages 2, 10, 14, 17, 51

[49] A. Norta and P. Grefen. Discovering Patterns for Inter-Organizational Business Collaboration. *International Journal of Cooperative Information Systems (IJCIS)*, 16:507 – 544, 2007. pages 3

[50] A. Norta, P. Grefen, S. Angelov, and L. Kutvonen. A Reference Architecture for Electronic Business-to-Business Collaboration Setup and Enactment Systems. Technical Report: C-2010-2, University of Helsinki, Finland, 2010. http://www.cs.helsinki.fi/u/anorta/publications/nortagrefen10esra.pdf. pages 17

[51] Michael P. Papazoglou. Web services and business transactions. *World Wide Web*, 6:49–91, March 2003. pages 14

[52] Michael P. Papazoglou, Piet Ribbers, and Aphrodite Tsalgatidou. Integrated value chains and their implications from a business and technology standpoint. *Decis. Support Syst.*, 29:323–342, November 2000. pages 14

[53] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing. *Communications of the ACM*, 46:25–28, 2003. pages vii, 4

[54] Mike P. Papazoglou and Benedikt Kratz. B.: A business-aware web services transaction model. In *ICSOC 2006. LNCS*. Springer, 2006. pages 78

[55] M.P. Papazoglou. Web Services and Business Transactions. *World Wide Web: Internet and Web Information Systems*, 6:49–91, 2003. pages 15

[56] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998. pages 6

[57] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets II: Applications*, volume 1492 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998. pages 6

[58] Sini Ruohomaa and Lea Kutvonen. Trust and distrust in adaptive inter-enterprise collaboration management. *Journal of Theoretical and Applied Electronic Commerce Research, Special Issue on Trust and Trust Management*, 5(2):118–136, August 2010. pages 7

[59] G. Shanks. Semiotic approach to understanding representation in information systems. In *Proceedings of the Information Systems Foundations Workshop: Ontology, Semiotics and Practice*. Macquarie University - Sydney, 1999. pages 7

[60] Yiyun Shen, Markus Miettinen, Pirjo Moen, and Lea Kutvonen. Privacy preservation approach in service ecosystems. In *Proceedings of the 15th IEEE International EDOC Conference Workshops*, pages 283–292, Helsinki, Finland, August 2011. IEEE Computer Society. pages 6

[61] J.D. Tygar. Atomicity in electronic commerce. In *Proc. of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC'96)*, pages 8–26, New York, 1996. pages 15

[62] J. Vonk and P.W.P.J. Grefen. Cross-organizational transaction support for e-services in virtual enterprises. *Distributed and Parallel Databases*, 14(2):137–172, 2003. pages 13, 82, 83

[63] Jochem Vonk, Paul W. P. J. Grefen, Erik Boertjes, and Peter M. G. Apers. Distributed global transaction support for workflow management applications. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, DEXA '99, pages 942–951, London, UK, 1999. Springer-Verlag. pages 13

[64] T. Wang. *A TxQoS-aware Business Transaction Framework*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2011. pages viii, 78, 79, 80, 82, 83

[65] T. Wang, P. Grefen, and J. Vonk. Abstract transaction construct: Building a transaction framework for contract-driven, service-oriented business processes. In A. Dan and W. Lamersdorf, editors, *Service-Oriented Computing - ICSOC 2006, 4th International Conference*, volume 4294 of *Lecture Notes in Computer Science*, pages 352–364, Chicago, USA, December 2006. LNCS Springer. pages 78

[66] Ting Wang, Jochem Vonk, Benedikt Kratz, and Paul Grefen. A survey on the history of transaction management: from flat to grid transactions. *Distrib. Parallel Databases*, 23:235–270, June 2008. pages 78, 82, 84

[67] J. Weber and S. Parastatidis. Demystifying service-oriented architecture. *Web Services Journal*, 3(11):40–44, 2003. pages 2

[68] Gerhard Weikum and Hans-J. Schek. Concepts and applications of multilevel transactions and open nested transactions. In *Database Transaction Models for Advanced Applications*, pages 515–553. Morgan Kaufmann, 1992. pages 10, 11

[69] E. Wustenhoff. Service level agreement in the data center, 2002. pages 79

[70] Ltd X/Open Company. Distributed transaction processing: Reference model, version 3, 1996. pages 11

[71] Eric Yu, Philippe Du Bois, Eric Dubois, and John Mylopoulos. From organization models to system requirements - a "cooperating agents" approach, 1998. pages 14