Conflict-Resolution Lifecycles for Governed Decentralized Autonomous Organization Collaboration

Alex Norta
Tallinn University of Technology
12618 Akadeemia tee 15A
Tallinn, Estonia
alex.norta.phd@ieee.org

Anis Ben Othman
Tallinn University of Technology
12618 Akadeemia tee 15A
Tallinn, Estonia
anis.ben@gmail.com

Kuldar Taveter
Tallinn University of Technology
12618 Akadeemia tee 15A
Tallinn, Estonia
kuldar.taveter@ttu.ee

ABSTRACT

Recent blockchain-technology related innovations enable the governance of collaborating decentralized autonomous organizations (DAO) to engage in agile business-network collaborations that are based on the novel concept of smart contracting. DAOs utilize service-oriented cloud computing in a loosely coupled collaboration lifecycle with the main steps of setup, enactment, possible rollbacks and finally, an orderly termination. This lifecycle supports the selection of services provided and used by DAOs, smart contract negotiations, and behavior monitoring during enactment with the potential for breach management. Based on a sound understanding of the collaboration lifecycle in a Governance- as-a-Service (GaaS)-platform, a new type of conflict management must safeguard business-semantics induced consistency rules. This conflict management involves breach detection with recovery aspects. To fill the detected gap, we employ a formal design-notation that comprises the definition of structural and behavioral properties for exploring conflict-related exception- and compensation management during a decentralized collaboration. With the formal approach, we generate a highly dependable DAO-GaaS conflict model that does not collapse under left-behind clutter such as orphaned processes and exponentially growing database entries that require an unacceptable periodic GaaS reset.

CCS Concepts

• Information systems • Computer systems organization.

Keywords

Decentralized autonomous organization, conflict resolution, e-governance, smart contract, open cloud ecosystem, service orientation, business process, Industry 4.0

1. INTRODUCTION

The emergence of service-oriented cloud computing (SOCC) [36] promises for companies an accelerated e-governance for cross-enterprise-collaboration (CEC) [5] with a seamless, ad hoc integration and coordination of information- and business-process flow. The latter orchestrate and choreograph heterogeneous legacy-system infrastructures [20]. Such governance automation of cross-enterprise collaboration enhances overall efficiency and effectiveness. Additionally, a trend-reinforcement occurs with the concept of so-called decentralized autonomous organizations (DAO) that are powered by smart contracts [4, 30] to form agreements with people via the block chain [16]. The ontological concepts and properties for the design of smart-contracting systems [26] we derived from legal principles, economic theory, and theories of reliable and secure protocols. The smart contract itself is a computerized transaction protocol [29] that executes the terms of a contract. The blockchain is a distributed database for independently verifying the chain of ownership of artifacts in hash values that result from cryptographic digests [25].

The SOCC-paradigm facilitates a loose coupling and highly dynamic establishment in the governance of business collaboration. Services are self-describing, business-process grouped logical manifestations of physical resources, i.e., as a set of actions [35, 31] that an organization executes and exposes to the web. To achieve non-repudiation in CEC, the registration of business transactions is of major legal importance for organizations. A business transaction [14] is a well-defined business-function driven consistent change in the state of a business relationship.

While each DAO holds its own business transaction [12, 32], for CEC-governance a transaction conflict-resolution concept is important to ensure collaboration reliability. With the complexity involved, no single transaction model is able to meet all requirements. Instead, it is necessary to cross- organizationally establish transaction frameworks in a way that does not force companies into disclosing an undesirable amount of business internals [5]. While conflict management is addressed on a collaboration-model level [17], it is also important to cater on a Governance-as-a-Service (GaaS) level for conflict management and -resolution. This paper fills the gap by investigating the research question how to govern in a dependable way the flow of business semantics in a meaningfully automated CEC-governance lifecycle? Note that meaningful automation in this sociotechnical context recognizes complex organizational work design with interaction between people and technology in workplaces. Thus, we assume humans want automation for tedious work but retain final decision-making power. Furthermore, dependable [2] means the components that are part of the governance- lifecycle are relied upon to perform exclusively and correctly the system task(s) under defined operational and environmental conditions over a defined period of time. Based on this main research question, we deduce the following sub- questions to establish a separation of concerns. What is the underlying CEC-governance

EGOSE '15, November 24-25, 2015, St.Petersburg, Russian Federation
© 2015 ACM. ISBN 978-1-4503-4070-0/15/11…$15.00

DOI: http://dx.doi.org/10.1145/2846012.2846052

lifecycle and which business semantics flows along it? When exceptional governance scenarios occur, what mechanisms exist in automated CEC for an orderly conflict resolving compensation-rollback and partial-, or complete lifecycle termination? What are the relevant system properties for successfully realizing the CEC-governance lifecycle with a Governance-as-a-Service (GaaS) platform in a Cloud?

The remainder of the paper is structured as follows. Section 2 provides additional information relevant for understanding the business-collaboration context. Section 3 shows the top-level of the formalized CEC-governance lifecycle in which service protocols are visible with their data-exchanges. Furthermore, in Section 4 we show the successful return of business semantics to earlier stages for compensating exceptions in transactional rollbacks within a CEC-governance lifecycle. This section also shows varying types of lifecycle terminations that leave the Cloud ecosystem behind in a clean state so that no clutter-accumulation necessitates a total reset. The latter is not an option when simultaneously business-critical collaborations are in progress. Section 5 shows the feasibility of the approach by listing the results from model checking that are instrumental for implementing a sound CEC-governance lifecycle with conflict-resolution provisions. Section 6 gives related work and finally, Section 7 concludes this manuscript by summarizing the re-search work, giving the contributions achieved and showing directions for future work.

2. BUSINESS CONTEXT

For comprehending the governance-lifecycle in the sequel, the following frameworks are important to comprehend. The essential way for DAOs-relationships is peer-to-peer (P2P) and therefore, we clarify the corresponding collaboration model in Section 2.1. Furthermore, as contracts are the foundation of business collaboration, we present in Section 2.2 pre-existing concepts and properties for smart contracting.

2.1 P2P-collaboration model

Pertaining to DAO-collaboration, Figure 1 conceptually depicts a configuration. The in-house process of a service consumer is a so-called business-network model (BNM) [27] in the P2P-case. A BNM captures choreographies that are relevant for a business scenario. A BNM contains legally valid template contracts that are service types with assigned roles. Together with the BNM, the service types with their roles are available in a collaboration hub that houses business processes as a service (BPaaS-HUB) [22] in the form of subset process views [5]. The latter address the need to semi-automatically find collaboration parties and learn about their identity, services, and reputation. A BPaaS- HUB enables speedy business-partner discovery and support for on-the-fly background checking with a matching of services.

On the external layer of Figure 1, service offers match with service types from the BNM identically to the contractual sphere of collaborating parties. Additionally, a collaborating partner must match into the role specifics associated with a respective service type. We refer the reader to [5] for details about the tree-based process-view matching to establish a DAO-collaboration configuration.

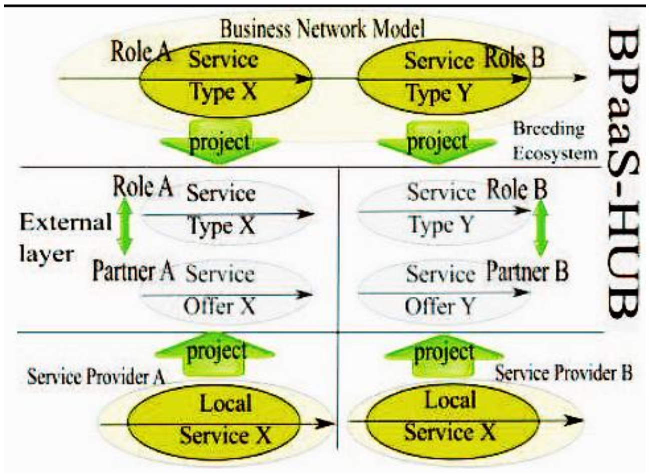
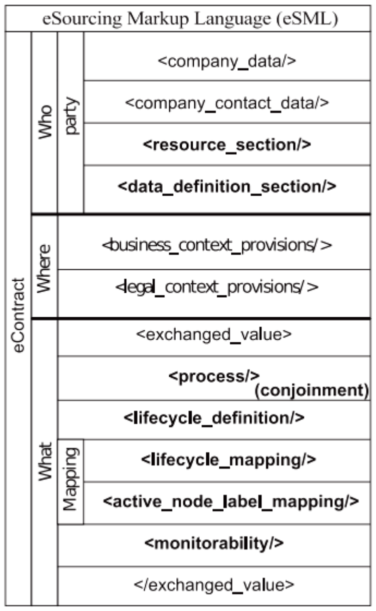


Figure 1. P2P-collaboration with eSourcing



eSourcing Markup Language (eSML)			
eContract	Who	party	<company_data/>
			<company_contact_data/>
			<resource_section/>
			<data_definition_section/>
	Where		<business_context_provisions/>
			<legal_context_provisions/>
	What		<exchanged_value>
			<process/> (conjoinment)
			<lifecycle_definition/>
		Mapping	**<lifecycle_mapping/>**
			<active_node_label_mapping/>
			<monitorability/>
			</exchanged_value>

Figure 2. Smart contracting concepts and properties [23]

2.2 Smart contract

We show the top-level structure of a smart contracting language termed eSourcing Markup Language (eSML) [23] in Figure 2. The bold typed eSML-definitions in Figure 2 are extensions and modifications that are not part of the Electronic Contracting Markup Language (ECML) [1] foundation.

The core structure of a smart contract we organize according to the interrogatives Who for defining the contracting parties together with their resources and data definitions, Where to specify the business- and legal context, and What for specifying the exchanged business values. For achieving a consensus, we assume the What-interrogative employs matching process views that require cross-organizational alignment for monitorability We refer to [23] for more information about the smart-contracting ontology. Also note that we refer from hereon to a smart contract as an eContract.

We next discuss the DAO-lifecycles for the setup-, enactment- and orderly termination stages in terms of control- and data flow.

3. COLLABORATION LIFEYCLE

We formalize with Colored Petri Nets (CPN) [11] the governance lifecycle. CPN is a language for the design, specification, simulation and verification of systems and has a graphical representation with a set of modules, each containing a network of places, transitions and arcs. The modules interact through well-defined interfaces and the data elements of the overall governance lifecycle are declared for all refinement levels. We use CPN Tools[1] for designing, simulating, performance testing and verifying the models in this paper. Note that way we solve many dependability issues in the design of such a complex system [2]. Table 1 lists relevant token colors [18] with their hierarchic service-refinement availability mentioned in the left column (1 for the top level and 6 for the most detailed refinement). Token colors are present for all lower but not for any higher CPN-refinement-hierarchy levels. The fourth column explains the purpose of a token color for a lifecycle. The integer-type tokens mostly represent an identification number and string-type tokens are either eContract-negotiation outcomes or eContract proposals. Boolean-type tokens represent decision points in the lifecycle.

For the remainder, Section 3.1 shows the top-level formalization of the eContract-setup phase. Section 3.3 elaborates on the enactment phase of the DO-lifecycle and Section 3.2 gives the rollout of a decentralised governance infrastructure. Finally, Section 3.4 shows the governance termination. Note that due to page limitation, we only show a subset of models while we make the complete CPN-model available[2] and also fully document [18] all models.

3.1 Setup Phase

The depiction in Figure 3 shows the governance lifecycle in the form of a CPN for forming an eCommunity [13] and starts with the creation of a BNM that contains service offers for validation against service types, and additionally, roles are assigned to the services. Concrete collaborating partners fill these roles during the eContract negotiation.

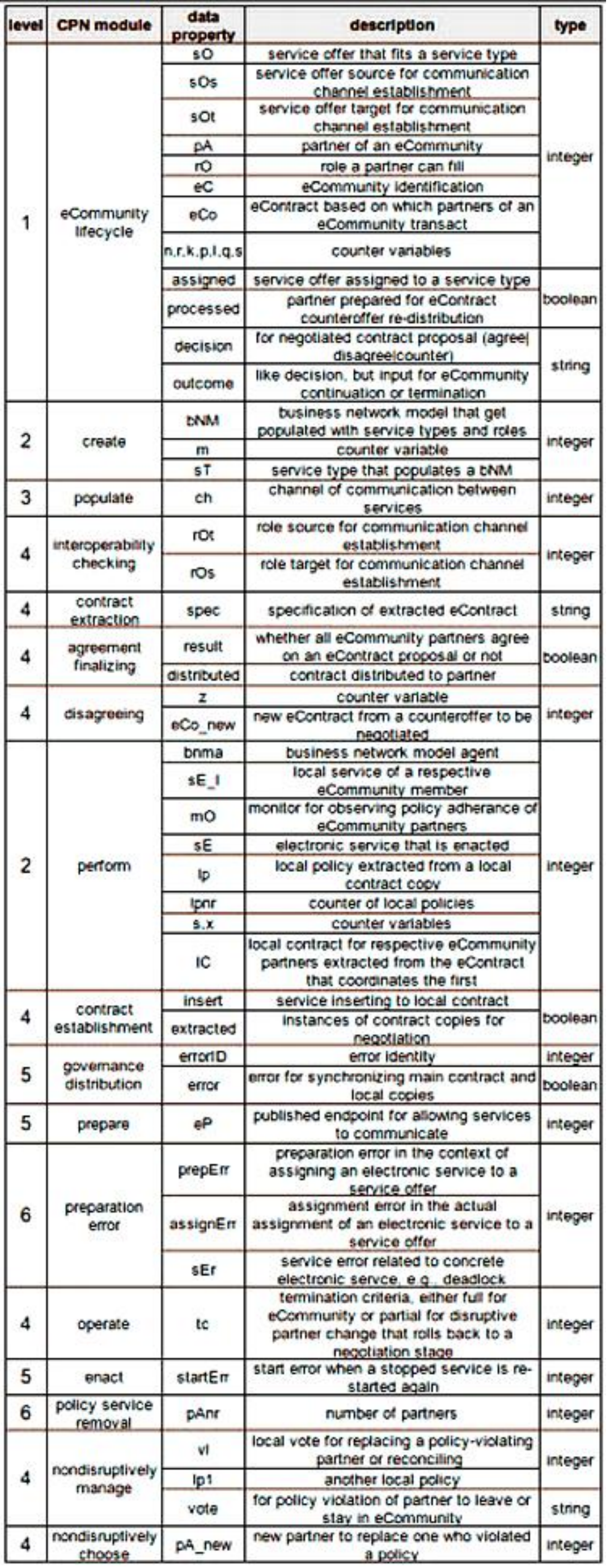
As Figure 4 shows in a limited screenshot of the actual CPN-module, the partners that slip into roles must vote on agreeing, or rejecting an eContract proposal that is based on a picked BNM. Rejection terminates the eCommunity while having all partners agree, results in a consensual eContract passed on to the next service. A third option during the negotiation phase is the proposal of a contract alternative.

If all partners agree during the negotiate stage, the eContract comes into existence that serves as a coordinating agent [28]. In the enterprise infrastructure distribution, local eContract copies come into existence for every eCommunity-partner together with business network model agents (BNMA) and monitors. The extraction stage from the eContract creates sets of policies from the local contracts and assigns each a BNMA and monitor. The final preparation stage populates the lowest technical collaboration-level with matching services and corresponding endpoints for communication channels before enactment.

[1] http://cpntools.org/

[2] CPN-DAO collaboration: http://tinyurl.com/kq7qjvl

Table 1. Data properties used in the DAO-lifecycle [18]



level	CPN module	data property	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

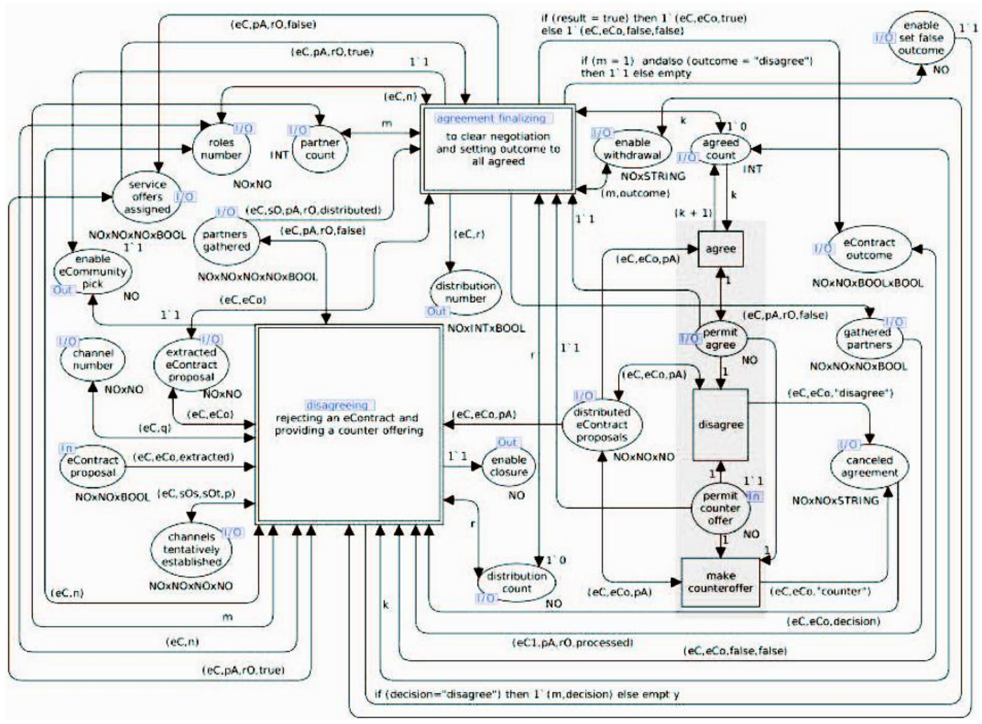


Figure 4: The decision phase in the *negotiate* module [18]

3.2 Decentralized governance infrastructure

Due to page limitations we give a short overview of the decentralized governance infrastructure (DGI) rollout and refer to [18] for further details. Briefly, every eCommunity partner receives a local eContract copy that the agreed upon eContract governs. Thus, the latter functions as a controlling agent. Every local eContract copy is the source to extract a respective set of policies, monitors and 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.

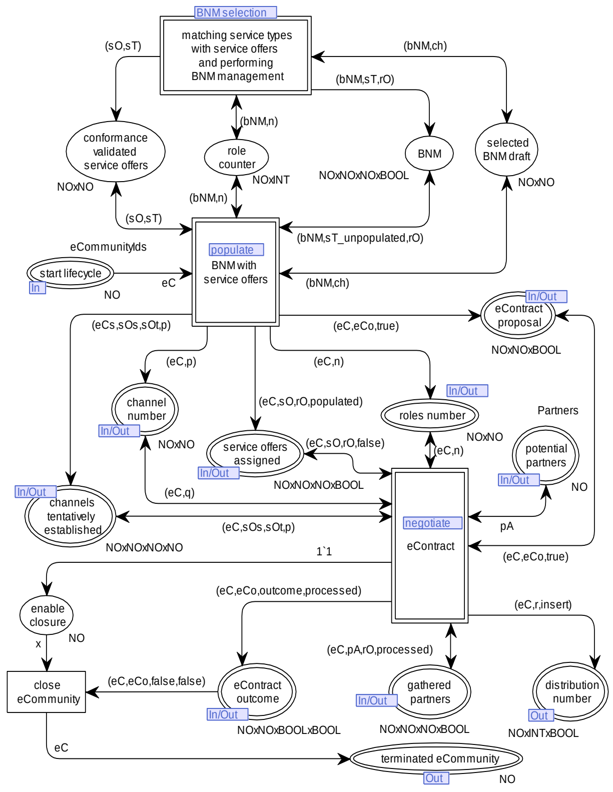


Figure 3. The setup phase of a smart contract in the create module [18]

3.3 Enactment phase

The electronic services that fill the service-offer templates in the eContract are complemented by in-house services. In Figure 5 these latter services reside in the central state *enacting services*. We assume for the gray-shaded part of Figure 5, 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 [5, 19].

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.

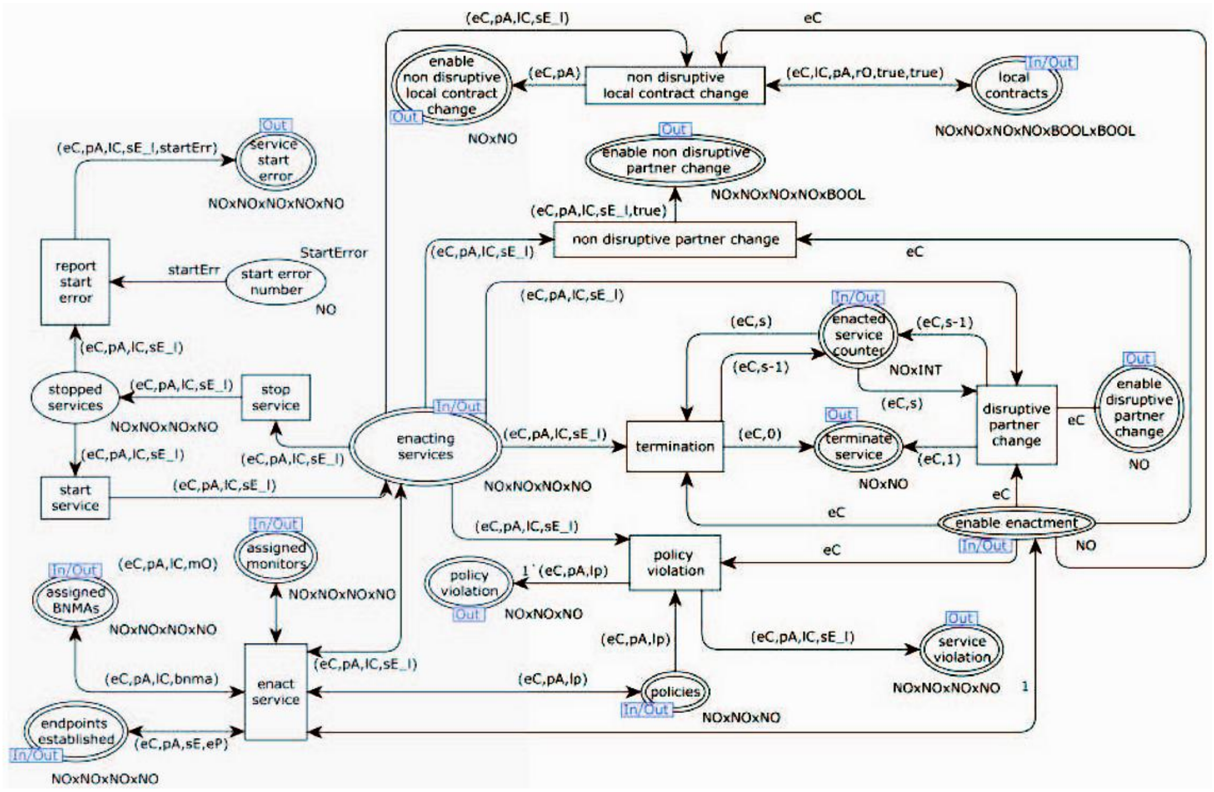


Figure 5. The enact module in gray shade [18]

3.4 Termination phase

The DAO-governance dissolves in three stages, as Figure 6 shows. The depicted terminate-module not only eliminates the technical-level setup of local eContract copies, but also the remainder of the distributed collaboration infrastructure. Correspondingly, first the policy removal commences with the termination of all electronic services from the technical level in enactment. Next, the counter of the amount of eCommunity-partners triggers the removing of all policies that are prior extracted from local eContract copies.

The elements for observing the enactment of an eContract are the behavior-monitors and BNMAs. These infrastructure elements for observing the eCommunity-partner behavior are removed as part of the DAO-governance infrastructure together with related local eContract copies. Finally, it is possible to remove the communication channels represented by the established endpoints.

The roles removal 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.

4. CONFLICT ROLLBACK

The top-level conflict rollback resembles a Saga [6] transaction being an idea adopted from chained transactions [33] of including a compensation mechanism to roll back. Traditional 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 a whole transaction back to the very beginning with compensations. Note that failures refer to traditional database settings. However, in the governance lifecycle in this research, it is behavior-controlling policy violations that result in the undoing by sub-transactions. More recently, blockchain technology promises to be the foundation for the next generation of very large and distributed database systems [15], e.g., the blockchain-based DB called Guardtime3[3]. Since blockchains as a specific transaction type solve the Byzantine generals' problem [7], the feature of non- repudiation of recorded collaboration events enables effective decentralized P2P trust management. With this foundation, we specify below a novel e-governance framework that controls the business-semantics ow in a very targeted way.

Three business-semantics rollback scenarios exist that may either be disruptive or calming, and that govern the transition of an eCommunity from one epoch to another. A conceptual depiction of these rollbacks Figure 7 shows. Briefly, disruptive rollbacks imply an eContract renegotiation must start from scratch again. Calming rollbacks imply that the DAOs of an eCommunity see scope to reconcile collaboration issues. In both cases, the eCommunity experiences epoch changes.

Pertaining to Figure 7, there are one disruptive and three conflict calming business-semantics rollbacks. The first type of disruptive business-semantics rollback that Section 4.1 discusses, 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-

[3] Guardtime: https://guardtime.com/blog

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.

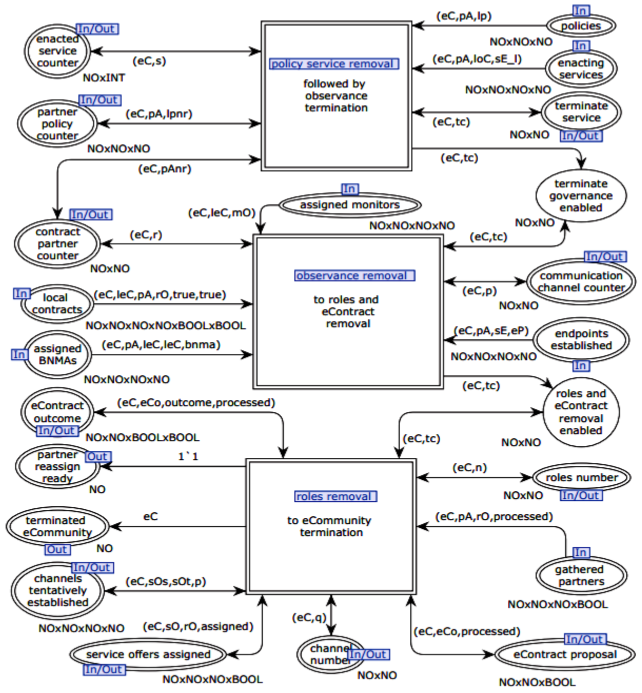


Figure 6. The terminate module [18].

Of the remaining three conflict calming business-semantics rollbacks, one that we explain in Section 4.2, also replaces an eCommunity-partner in a way where it does not dis- mantle and recreate the DGI. Likewise, the other conflict calming business-semantics rollbacks equally leave the existing DGI intact. Section 4.4 gives a calming rollback type where a policy violation an eCommunity does not consider severe enough to justify a dismantling and re-creation of a DGI.

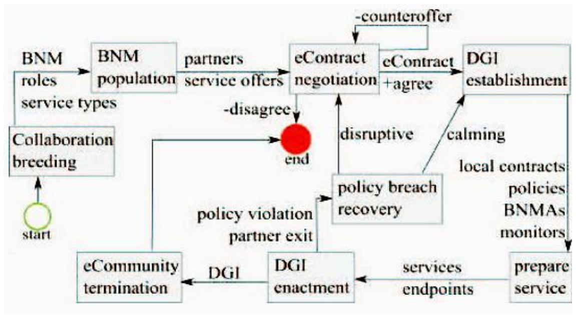
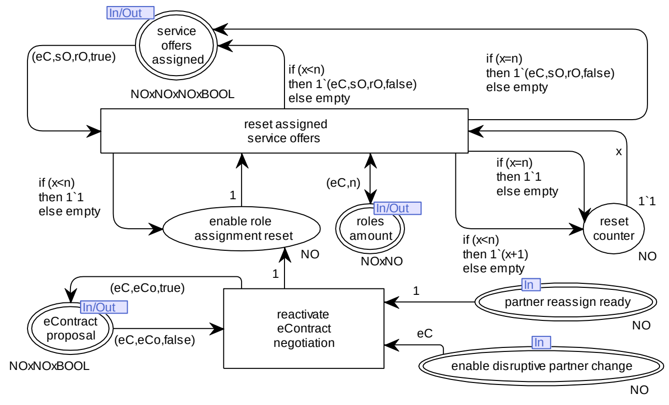


Figure 7. A conceptual lifecycle for rolling back collaboration conflicts [18]

If the eCommunity-partners vote to ignore a respective policy violation, the related eContract enactment resumes without any modification. The third type of conflict calming business-semantics rollback in Section 4.3, allows the complete replacement of a local eContract copy with a new one as part of the existing DGI that remains otherwise unchanged. Thus, the new local eContract and related policies, BNMA and monitor must adhere to the main DGI coordinating eContract agent.

4.1 Disruptive collaboration reset

The delivery of the identification number of the eCommunity from the *enact*-service in Figure 5 by a corresponding transition labeled *disruptive partner change* triggers the service for a *disruptive* collaboration reset depicted in Figure 8. The same transition also triggers a partial termination and removal of the DGI. When the partial collaboration termination completes, an enabling token enters the input node labeled partner reassign ready that repeats the eContract negotiation by the correspondingly named service in Figure 8. Before the re-negotiation starts, it is necessary to reset the service offers for extracting contract proposals. Since every eCommunity-partner has a service offer to populate, the amount of resets equally results from the partial DGI-termination.



In/Out
service offers assigned
(eC,sO,rO,true)
NOxNOxNOxBOOL
if (x<n) then 1`(eC,sO,rO,false) else empty
if (x=n) then 1`(eC,sO,rO,false) else empty
reset assigned service offers
if (x<n) then 1`1 else empty
if (x=n) then 1`1 else empty
(eC,n)
enable role assignment reset
roles amount
reset counter
if (x<n) then 1`(x+1) else empty
(eC,eCo,true)
eContract proposal
(eC,eCo,false)
reactivate eContract negotiation
partner reassign ready
enable disruptive partner change

Figure 8: Triggering a DGI-dismantling and re- negotiation of an eContract. (disruptively reset) [18].

4.2 Conflict-calming replacement

The service of Figure 9 consists of two parts. On the one hand, a part for removing temporarily all elements related to a local eContract copy, i.e., policies, BNMA, endpoint, monitor. The reason for this temporary local removal is detecting the identification key for all related elements that comprise a composite of the unique number of the eCommunity and contained partner requiring change. Thus, besides inserting a new eCommunity-partner itself, a replication happens in the composite identification keys of the related local eContract-copy elements. The second nested service in Figure 9 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.

4.2.1 Local DGI removal

For locally replacing an eCommunity-partner in Figure 10, a token enters in the state labeled enable local partner change comprising the identifiers of the eCommunity, old partner to be replaced, local eContract copy and local service. The arrival of that token commences a stepwise, temporary removal of a uniquely identified local eContract copy and related elements starting with the respective BNMA.

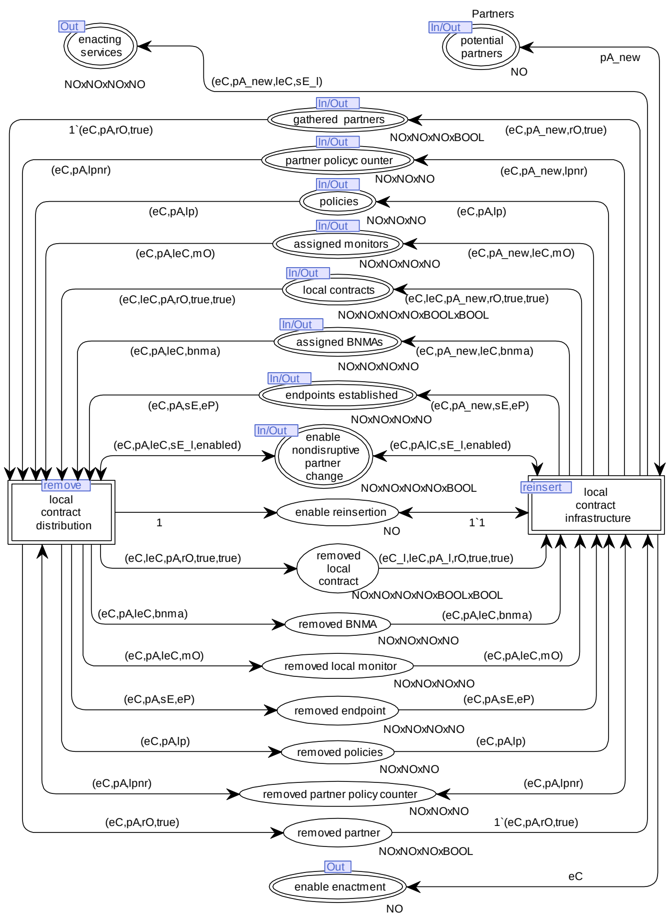


Figure 9. Distributing local eContract copies, policies and populating with local electronic services (*nondisruptively choose*) [18]

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

4.2.2 *Local DGI reinsert*

When the remove-service completes and adds a token into the state labeled enable reinsertion, the service in Figure 11 commences with reinserting into the existing DGI the local eContract copy and related elements. To enable reinserting a local eContract copy and related other DGI-elements, a new partner is chosen based on the acceptance of the remaining eCommunity to be a new replacement. The latter 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 eContract copy 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 eContract copy and associated endpoint, BNMA, monitor, policies and related amount counter.

4.3 Calming DGI change

A business-semantics rollback-option in Figure 12 that leaves the existing DGI intact and replaces a local eContract copy with a modified version. Thus, a composed identification key comprises the unique key of the eCommunity and partner who's local eContract copy requires modification. As a modification trigger, we consider, e.g., a minor change in the business environment of a party that is not significant enough to justify an entirely newly created DGI. Figure 12 shows that a nested subservice first stepwise temporarily removes a local eContract copy and related elements from an existing DGI that requires modification. When the local removal completes, the enabling of the insertion-service repopulates the DGI with a modified local eContract copy by rolling back into the Figure 7 depicted DGI establishment.

4.3.1 *DGI change removal*

This service in Figure 13 is organized in a cycle around the state labeled *enable non disruptive local eContract copy change*. The enabling token for local eContract copy removal comprises a tuple with colors representing a composed identification key, i.e., the unique number of the eCommunity and party who's local eContract copy requires modification, all elements related to the local eContract copy are stepwise removed while leaving the rest of the DGI intact.

4.3.2 *DGI change insertion*

In Figure 14, the DGI re-population 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 *eContract copys* that is currently without further refinement.

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

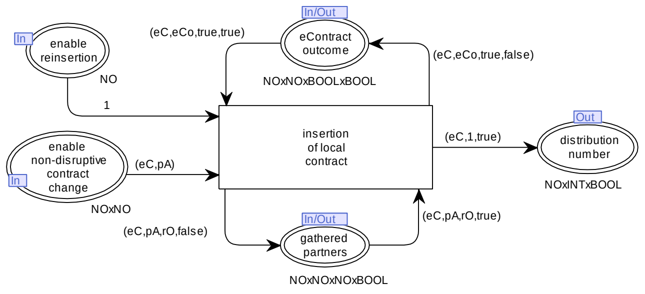


Figure 14. Reinsertion of a changed local eContract copy with related elements (insertion) [18].

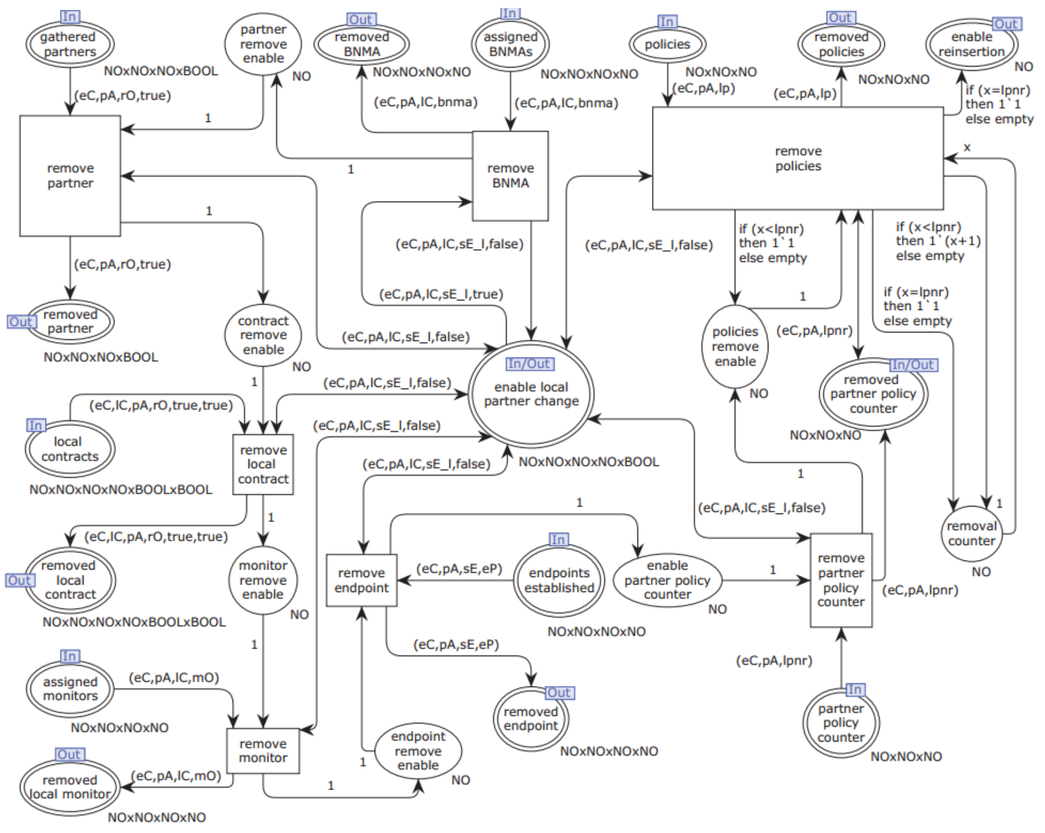


Figure 10. Removing local eContract copies, partners, policies, monitors, service endpoints and BNMAs (remove) [18]

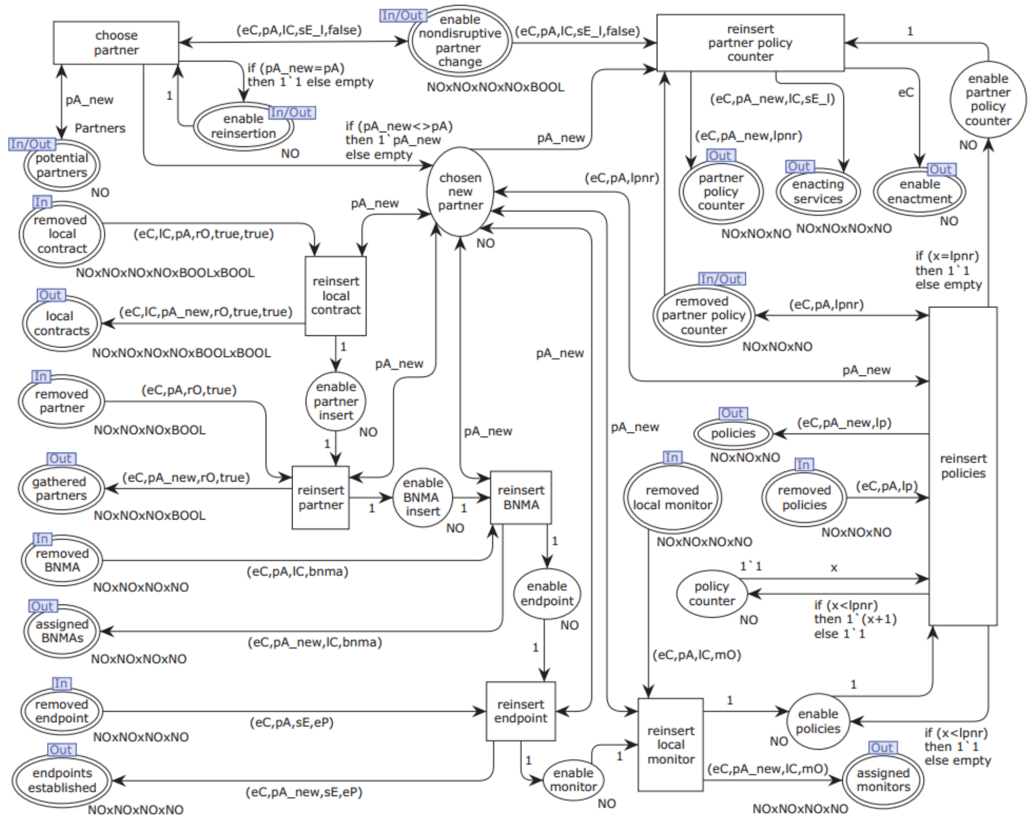


Figure 11. Reinserting local eContract copies, partners, policies, monitors, service endpoints and BNMAs (reinsert) [18

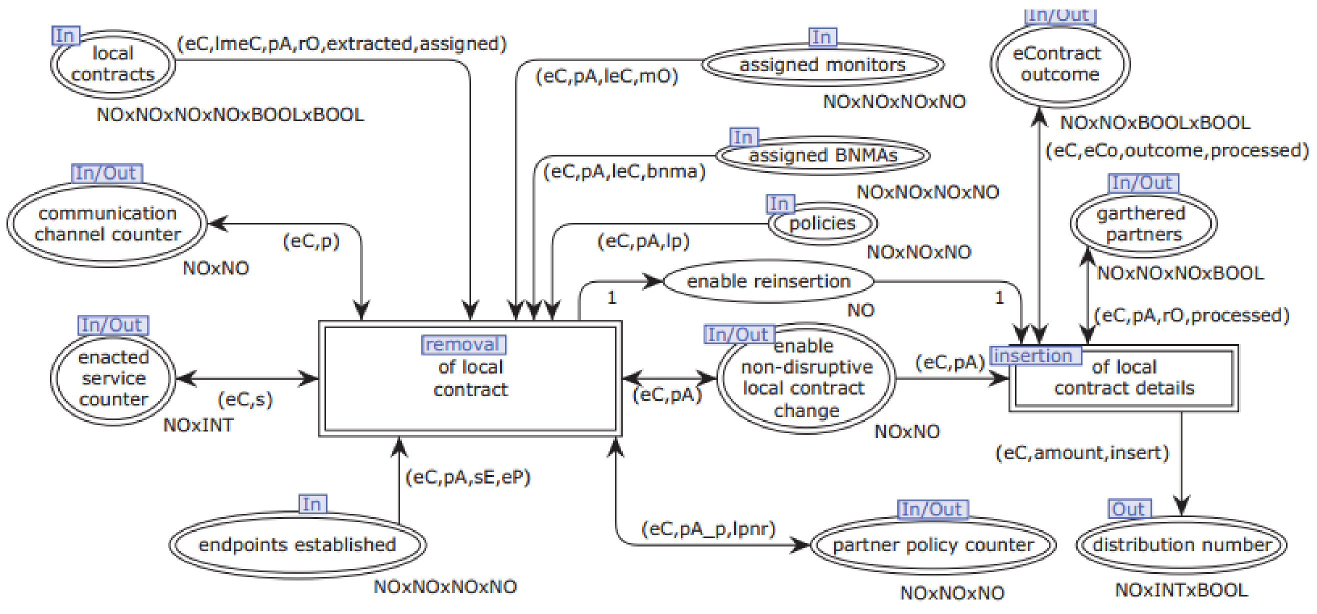


Figure 12. Protocol for enacting and terminating an eContract (*nondisruptively change*) [18].

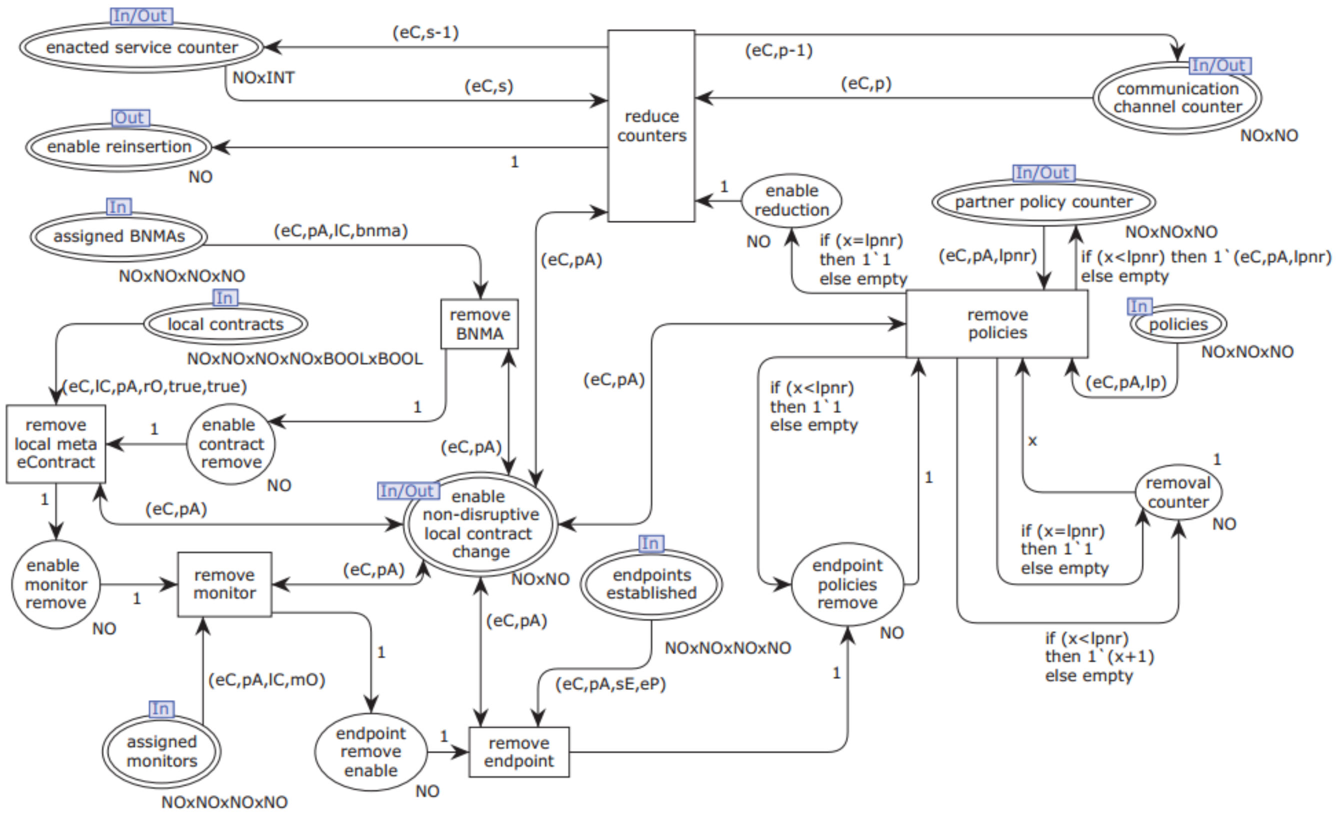


Figure 13. Stepwise DGI removal for calming-change preparation (removal) [18].

4.4 Conflict-calming voting management

If a policy violation starts the service in Figure 15, 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 15, 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 second 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 15 does not refine the reconciliation option, we consider it too future work.

Next, we discuss a formal and practical proof-of-feasibility evaluation.

5. FEASABILITY EVALUATION

There are two parts of evaluation we conduct. First, in Section 5.1, a formal model-checking summary indicates the behavioral- and structural properties of the DAO-governance lifecycle. Furthermore, in Section 5.2, we map the modules of the collaboration lifecycle into the eSourcing Reference Architecture eSRA [20]. By doing so, it is possible to estimate the technical feasibility of a system implementation.

5.1 Model-checking results

From a developer perspective, the motivation for analyzing the DAO-governance lifecycle is to know if it terminates correctly and is thereby dependable. A simple way to testing is a simulation with a token game in CPN Tools. However, while manual- and semi-automatic simulations yield a deeper understanding of the models for the designer, they fail to test all paths and aspects of complex DAO-governance lifecycles.

The listed services, i.e., CPN modules, in Table 2 are pragmatically chosen with respect to their testability. Note, we term CPN-modules as services in this manuscript as it adheres to the service-oriented cloud-computing paradigm. 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 contains atomic transitions. The state space of the overall transaction lifecycle is too big for testing in a computationally feasible way. However, besides verifying by model-checking, we additionally simulate extensively the entire governance lifecycle by employing a token-game for evaluating the termination correctness in the state terminated eCommunity.

The formalized DAO-governance 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 are states and arcs are 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 [18], 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 state may hold considering all reachable markings. informally, a marking gives the amount of token distributions in the states of a CPN at a specific point in time. The best upper integer bound of a state specifies the maximal number of tokens that can reside in a state in any reachable marking. The best lower integer bounds for a state specifies the minimal number of tokens that can reside in a state in any reachable marking. The best upper multi-set bound of a state specifies for each color in the color set of this state, the maximal numbers of tokens that is present in this state with the given color in any reachable marking. The best lower multi-set bound of a state specifies for each color in the color set of a state the minimal number of tokens that is present in this state 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 not fair if it is not just. Impartial considers all infinite runs while fair and just only consider some infinite runs.

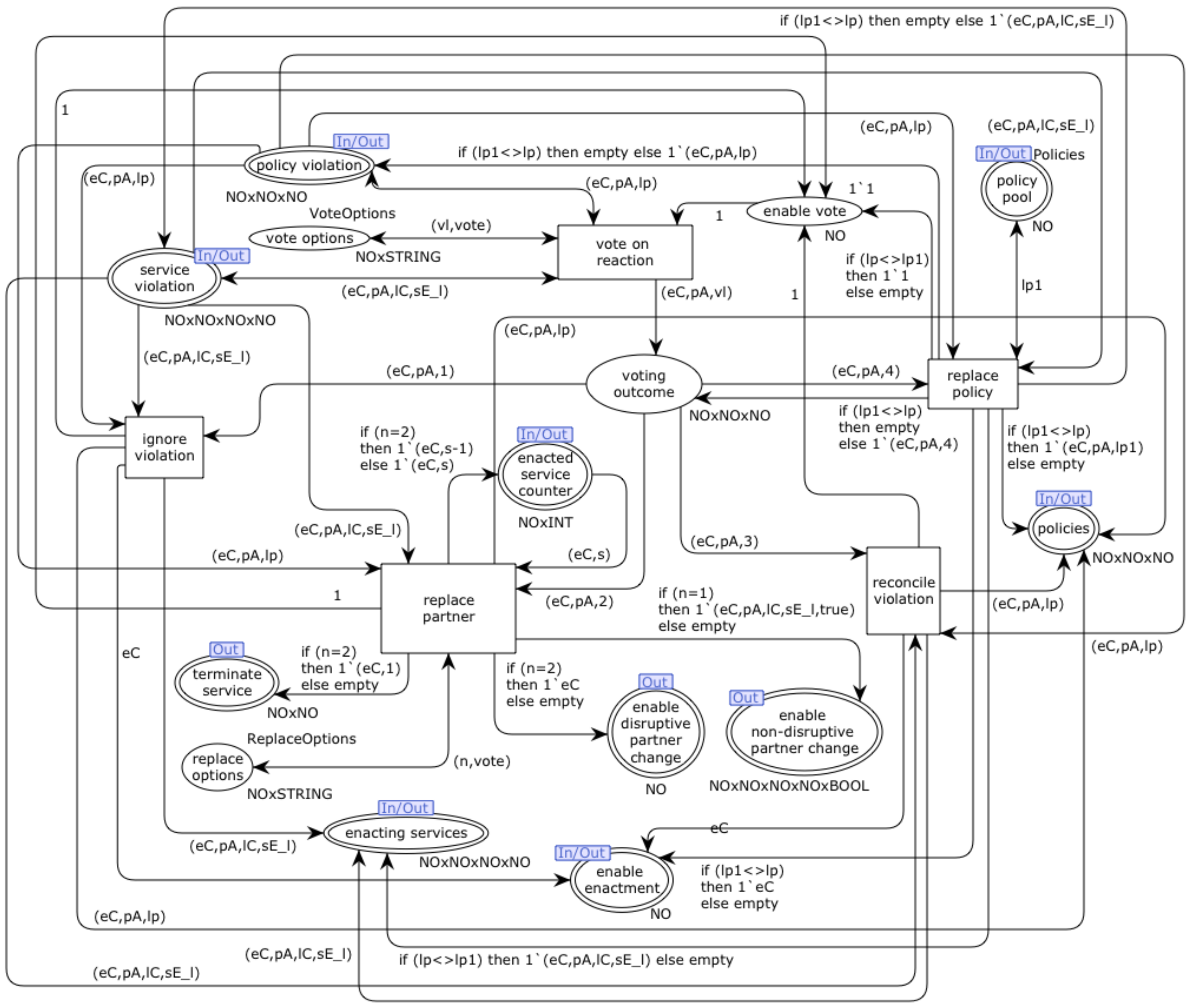


Figure 15. Voting options for calming eCommunity conflicts (nondisruptively manage) [18].

A summary of the analysis results we provide in Table 2 where the first column lists the services of the governance 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.

5.1.1 *Loops*

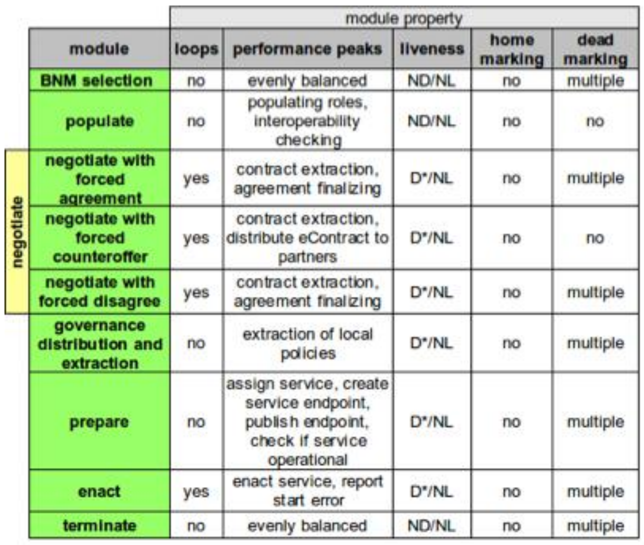
Detected loops in a model mean the system implementers must think carefully about enforceable termination criteria. Detected performance peaks mean, during runtime, provisions must be in place for elastic resource assignment, which is important for Cloud platforms.

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 2 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 [18] 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 2 show they do not contain loops.

5.1.2 *Performance peaks*

The column for performance peaks in Table 2 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 2 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.



		module property				
	module	**loops**	**performance peaks**	**liveness**	**home marking**	**dead marking**
	BNM selection	no	evenly balanced	ND/NL	no	multiple
	populate	no	populating roles, interoperability checking	ND/NL	no	no
negotiate	**negotiate with forced agreement**	yes	contract extraction, agreement finalizing	D*/NL	no	multiple
negotiate	**negotiate with forced counteroffer**	yes	contract extraction, distribute eContract to partners	D*/NL	no	no
negotiate	**negotiate with forced disagree**	yes	contract extraction, agreement finalizing	D*/NL	no	multiple
	governance distribution and extraction	no	extraction of local policies	D*/NL	no	multiple
	prepare	no	assign service, create service endpoint, publish endpoint, check if service operational	D*/NL	no	multiple
	enact	yes	enact service, report start error	D*/NL	no	multiple
	terminate	no	evenly balanced	ND/NL	no	multiple

Table 2. Model-checking results for the DAO- governance lifecycle modules [18]

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. The *governance distribution* and *extraction* of policies [18] 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.

5.1.3 Marking

While no tested service has any home marking, in Table 2, 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 2, testing the populate- service reduced to a counteroffer loop.

5.1.4 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 e orts of services.

The liveness column in Table 2 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? in Table 2 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.

With respect to liveness, all checked services comprise no live transitions. That means no transition is always red in any marking of a service. The test results for dead markings indicate no transition in a service is never red in any marking (which we indicate as ND). For a subset of services in Table 2, we assign D*, 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*-liveness in Table 2, the results show dead transitions are from marking paths we deliberately disabled.

5.2 Architecture representation

To enable the setup and enactment of process-view based collaboration evolution, a system must meet a set of requirements. First, there must exist a service that facilitates the matching of service offers from collaborating parties and service requests from consuming organizations. Second, the collaborating parties house internally a component for the distributed binding and enactment of emergency cases. Third, with tool support, the parties must rapidly develop service offers and concrete services. Fourth, each collaborating party is capable of orchestrating its own internal legacy system for automating the collaboration. Finally, due to the heterogeneity of the collaboration, a translation service must exist for bridging the differences (technical, syntactic, semantic, pragmatic) between collaborating parties.

The eSourcing Reference Architecture [20] supports these requirements. Figure 16 depicts the resulting architecture in UML-component diagram notation that takes into account the above listed requirements. The Service-HUB [22] as a trusted third party service in the middle that satisfies the first requirement and is suitable for the rapid setup phase of a DAO-governance collaboration. Each party has on an external layer an eSourcing Middleware for the technical binding after a successful setup that satisfies the second requirement. During the distributed collaboration-enactment, the eSourcing Middleware exchanges data via a security ensuring gateway with other DAOs. Thus, the eSourcing Middleware also comprises external workflow- and rules- enactment services that coordinate each other not only internally but also via the gateway with other parties.

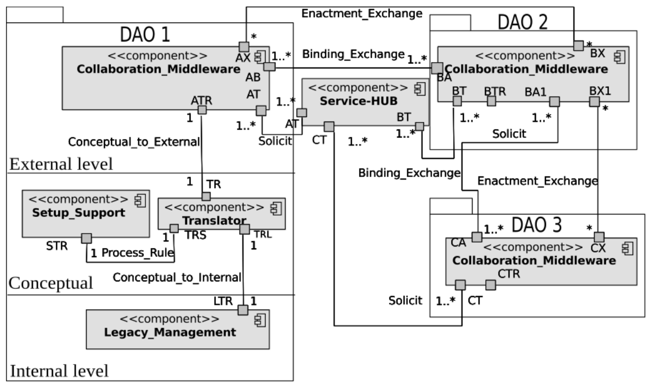


Figure 16. The eSourcing Reference Architecture eSRA [20] supports setting up eContracts

We assume there exists in each party a conceptual layer with a service for Setup Support that satisfies the third requirement and comprises tools for not only rapidly internally designing services and rules with the help of pattern libraries [21], but also includes a local verification- and simulation service. Next, each party has an internal layer with a service for Legacy Management that satisfies the fourth requirement and comprises local work ow- and rules enactment services that coordinate each other for the orchestration of Web-service wrapped internal legacy systems. Finally, the external- and internal enactment services exchange via a Translator service on the conceptual layer of each party to bridge the heterogeneous collaboration aspects. The Translator satisfies the final requirement and also connects on the conceptual layer with the Setup Support service. Note that we omit in Figure 16 the conceptual- and internal layer with the exception of the Hiring company due to space limitations.

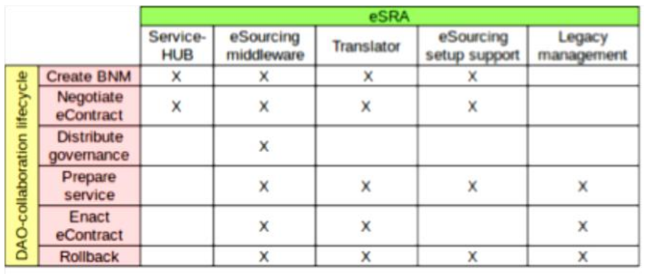
5.2.1 Governance-lifecycle mapping

The behavior in the eSRA-services of Figure 16 we determine by mapping in the DAO-governance lifecycle. Table 3 shows the highest-level assignment between services and the lifecycle stages.

By mapping the modules of the DAO-governance lifecycle into eSRA, we also yield an indication for the available technologies that are to technically realize an application-system implementation. We refer to [20] for details about the contained technological-feasibility study.

6. RELATED WORK

The need for formally backing socio-technical DAO-governance exists, as related work shows that stemps mostly from the virtual enterprise (VE) community. To enhance the development of VEs, the citation [8] presents a policy-based multi-agent management system. A hierarchical policy specification controls the behavior of agents. A detection algorithms and resolution strategies describe the conflicts that hierarchical policies may cause. A policy administration tool simplifies the operations of these policies. Each enterprise registered in this tool has an agent assigned that executes policies of enterprise.



		eSRA				
		Service-HUB	eSourcing middleware	Translator	eSourcing setup support	Legacy management
DAO-collaboration 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 3. Mapping the DAO-governance lifecycle into eSRA

The authors in [34] use buyer- and seller agents to form virtual enterprises that transact. The special focus lies on using ontologies and a fuzzy set theory based knowledge reuse method to bridge the gap between respective heterogeneous datasets of collaborating enterprises. Similarly, also [10] uses exclusively agents for distributed decision making in a VE for risk management by employing particle-swarm optimization. In [9], multiple categories of policies around agents only VE management are input for a conflict-management algorithm. All these works do not consider business-process oriented collaboration where agents perform conflict management.

Contracting is part of Web-service choreography in some research work that only takes a technical position. In [24], the authors describe various aspects of negotiating and agreeing contracts between software agents acting on behalf of enterprises or individuals. In [3], the authors consider contracts as labelled transition systems over located action names, representing operations at a certain location over a network. However, in this technicality focused approach, the authors study only the foundational aspects of contract compliance in a language independent way. The language independent representation of contracts allows for choreography projection in structured operational semantics, but does not take into account a satisfactory degree of cross-organizational collaboration suitability and expressiveness.

7. CONCLUSION

We investigate the lifecycle of cross-organizational business process aware collaborating governance that involves decentralized autonomous organizations. The means of governance setup are smart contracts that comprise machine readable code the parties in an eCommunity consent upon. The governance-lifecycle comprises a setup phase where a business network model contains service types to which partner roles are assigned. Concrete partners with their service offers populate the service types and a negotiation follows that results either in a dissent, counter-offer issuance, or a consent. The setup phase involves the establishment of a collaboration-governance infrastructure after which the decentralized enactment-phase commences.

During enactment, exceptional conflict-situations may occur that are addressed by the eCommunity partners in a sociotechnical way. That means, the parties have to establish together the degree of severity of a respective conflict. The decision can either be to disruptively roll back to the negotiation stage of the governance-lifecycle, or to pursue conflict-calming responses such as replacing an eCommunity party, or a policy, or a local eContract copy. We evaluate the governance lifecycle for decentralized autonomous organizations with means of model checking and also with mapping the governance behavior into a cross-organizational collaboration reference architecture that facilitates a technical feasibility evaluation for application-system implementation.

For future work, we plan to apply the governance lifecycle to cyber-physical systems in the realm of Internet-of- Things that require the choreography of several collaborating decentralized autonomous organizations and where also the orchestration of several types of smart-object flocks is necessary. Consequently, we will explore how blockchain technology can realize non-repudiation in process-aware smart contracting governance. Additionally, we aim to investigate how blockchain technology enables novel approaches for the effective management of distributed trust, reputation, privacy and security in cross-organizational cyber-physical system governance.

ACKNOWLEDGMENTS

This work was funded by the Trusted Business Transactions project of the Academy of Finland. Additionally, research funding involves the Estonian IT Akadeemia and also partly the research project SF0140013s10 ”Model-based Creation and Management of Evolutionary Information Systems” by the Estonian Ministry of Education and Research.

8. REFERENCES

[1] S. Angelov. Foundations of B2B Electronic Contracting. Dissertation, Technology University Eindhoven, Faculty of Technology Management, Information Systems Department, 2006.

[2] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing.

Dependable and Secure Computing, IEEE Transactions on, 1(1):11–33, 2004.

[3] M. Bravetti and G. Zavattaro. Contract compliance and choreography conformance in the presence of message queues. In R. Bruni and K. Wolf, editors, Web Services and Formal Methods, volume 5387 of Lecture Notes in Computer Science, pages 37–54. Springer Berlin Heidelberg, 2009.

[4] V. Butterin. A next-generation smart contract and decentralized application platform, 2014.

[5] R. Eshuis, A. Norta, O. Kopp, and E. Pitkanen. Service outsourcing with process views. IEEE Transactions on Services Computing, 99(PrePrints):1, 2013.

[6] 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.

[7] G. Giaglis and K. Kypriotaki. Towards an Agenda for Information Systems Research on Digital Currencies and Bitcoin. In W. Abramowicz and A. Kokkinaki, editors, Business Information Systems Workshops, volume 183 of Lecture Notes in Business Information Processing, pages 3–13. Springer International Publishing, 2014.

[8] J. Hu, Y. Song, and Y. Sun. Multi-agent oriented policy-based management system for virtual enterprise. Journal of Software, 7(10):2357–2364, 2012.

[9] J. Hu, Y. Song, and Y. Sun. Multi-agent oriented policy-based management system for virtual enterprise. Journal of Software, 7(10), 2012.

[10] M. Huang, F. Lu, W. Ching, and T. Siu. A distributed decision making model for risk management of virtual enterprise. Expert Systems with Applications, 38(10):13208 – 13215, 2011.

[11] K. Jensen, L. Michael, K. L. 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.

[12] B. Kratz. A model and language for business-aware transactions. Technical report, Tilburg University, 2012.

[13] L. Kutvonen, J. Metso, and S. 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.

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

[15] W. Mougayar. The blockchain is the new database, get ready to rewrite everything, http://startupmanagement.org/2014/12/27/the- blockchain-is-the-new-database-get-ready-to-rewrite- everything/, 2014.

[16] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Consulted, 1(2012):28, 2008.

[17] N. Narendra, A. Norta, M. Mahunnah, L. Ma, and F. Maggi. Sound Conflict Management and Resolution for Virtual-Enterprise Collaborations. Service-Oriented Computing and Applications, forthcoming.

[18] A. Norta. Safeguarding Trusted eBusiness Transactions of Lifecycles for Cross-Enterprise Collaboration. http://www.cs.helsinki.fi/u/anorta/publications/ TBT SOCC.pdf, 2012.

[19] A. Norta and R. Eshuis. Specification and verification of harmonized business-process collaborations. Information Systems Frontiers, 12:457–479, 2010. 10.1007/s10796-009-9164-1.

[20] A. Norta, P. Grefen, and N. Narendra. A reference architecture for managing dynamic inter-organizational business processes. Data & Knowledge Engineering, 91(0):52 – 89, 2014. [21] A. Norta, M. Hendrix, and P. Grefen. A pattern-knowledge base supported establishment of inter-organizational business processes. In R. Meersman and Z. Tari, editors, On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, and ODBASE, volume 4277 of Lecture Notes in Computer Science,page834¨ı£¡843,Montpellier,France,October 2006. LNCS Springer.

[22] A. Norta and L. Kutvonen. A cloud hub for brokering business processes as a service: A "rendezvous" platform that supports semi-automated background checked partner discovery for cross-enterprise collaboration. In SRII Global Conference (SRII), 2012 Annual, pages 293–302, July 2012.

[23] A. Norta, L. Ma, Y. Duan, A. Rull, M. Ko˜lvart, and K. Taveter. econtractual choreography-language properties towards cross-organizational business collaboration. Journal of Internet Services and Applications, 6(1):8, 2015.

[24] S. Ossowski. Agreement Technologies:Law, Governance and Technology. Springer, 2013.

[25] B. Panikkar, S. Nair, P. Brody, and V. Pureswaran. Adept: An iot practitioner perspective, 2014.

[26] T. Patron. The Bitcoin Revolution: An Internet of Money. Travis Patron.

[27] T. Ruokolainen, S. Ruohomaa, and L. Kutvonen. Solving service ecosystem governance. In Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2011 15th IEEE International, pages 18–25. IEEE, 2011.

[28] L. Sterling and K. Taveter. The art of agent-oriented modeling. MIT Press, 2009.

[29] M. Swan. Blockchain thinking: The brain as a dac (decentralized autonomous organization). In Texas Bitcoin Conference, pages 27–29, 2015.

[30] N. Szabo. Formalizing and securing relationships on public networks. First Monday, 2(9), 1997.

[31] L. Von Mises and B. Mayes. Human action. Classics on Tape, 1990.

[32] T. Wang. A TxQoS-aware Business Transaction Framework. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 2011.

[33] T. Wang, J. Vonk, B. Kratz, and P. Grefen. A survey on the history of transaction management: from flat to grid transactions. Distrib. Parallel Databases, 23:235–270, June 2008.

[34] X. Wang, T. Wong, and G. Wang. Service-oriented architecture for ontologies supporting multi-agent system negotiations in virtual enterprise. Journal of Intelligent Manufacturing, 23(4):1331–1349, 2012.

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

[36] Y. Wei and M. Blake. Service-oriented computing and cloud computing: Challenges and opportunities. Internet Computing, IEEE, 14(6):72–75, 2010.