



ブロックチェーンを用いた組織間ワークフロー管理 システムに関する研究

長野, 裕史

(Degree)

博士 (システム情報学)

(Date of Degree)

2025-09-25

(Date of Publication)

2026-09-01

(Resource Type)

doctoral thesis

(Report Number)

甲第9330号

(URL)

<https://hdl.handle.net/20.500.14094/0100498370>

※ 当コンテンツは神戸大学の学術成果です。無断複製・不正使用等を禁じます。著作権法で認められている範囲内で、適切にご利用ください。



博士論文

ブロックチェーンを用いた組織間
ワークフロー管理システムに関する研究

2025 年 7 月

神戸大学大学院システム情報学研究科

長野 裕史

目次

第1章	緒論	1
1.1	研究の背景	1
1.2	関連研究	9
1.2.1	組織間ワークフロー管理システムのアーキテクチャに関する関連研究	9
1.2.2	組織間ワークフロー実行エンジンの関連研究	10
1.2.3	組織間ワークフロー管理システムにおける情報開示範囲の限定とデータ量増大への対応に関する関連研究	10
1.3	研究の方針	11
1.4	本論文の構成	13
第2章	SPoT を排除した組織間ワークフロー管理システムのアーキテクチャ	15
2.1	緒言	15
2.2	組織間ワークフロー管理システムの要件	16
2.2.1	ワークフロー処理の概要	16
2.2.2	組織間ワークフローの前提	19
2.2.3	組織間ワークフロー管理システムの要件	20
2.3	SPoT を排除した組織間ワークフロー管理システムの概要	25
2.3.1	SPoT を排除した組織間ワークフロー管理システムのアーキテクチャ	25
2.3.2	トランザクション処理フロー	29
2.4	提案システムアーキテクチャの評価	32
2.4.1	組織間ワークフローのユースケース	32
2.4.2	提案システムアーキテクチャのセキュリティ評価	33
2.5	結言	38
第3章	ブロックチェーンを活用した組織間ワークフロー管理システムの実行エンジン	39
3.1	緒言	39
3.2	組織間ワークフロー管理システムの実装課題	39
3.3	逐次データ保証型ワークフロー実行エンジン	41
3.3.1	逐次データ保証型ワークフロー実行エンジンの考え方	41
3.3.2	SDV ワークフロー実行エンジンの詳細	45

3.4	SDV ワークフロー実行エンジンの評価	47
3.4.1	データサイズの評価	48
3.4.2	性能評価.....	51
3.5	考察	53
3.6	結言	54
第 4 章	情報開示範囲の限定と冗長性のないデータ構造による組織間ワークフロー管理システム	55
4.1	緒言	55
4.2	安全なワークフロー管理システムのためのアプローチ	56
4.3	SDVCD ワークフロー実行エンジン.....	58
4.3.1	データモデル	58
4.3.2	システムアーキテクチャとトランザクションの流れ	59
4.4	SDVCD ワークフロー実行システムの評価.....	62
4.4.1	評価条件.....	62
4.4.2	データ量評価	65
4.4.3	性能評価.....	66
4.4.4	SDV 方式と SDVCD 方式の比較評価	69
4.5	考察	72
4.6	結言	73
第 5 章	結論	75
5.1	本研究のまとめ	75
5.2	今後の課題	76
参考文献		79
研究業績		85
本論文の構成		87
謝 辞		89

图目录

FIGURE 1-1 EXAMPLE OF WORKFLOW.....	2
FIGURE 1-2 EXAMPLE OF CROSS ORGANIZATIONAL WORKFLOW	3
FIGURE 1-3 CHALLENGES OF THIS THESIS.....	9
FIGURE 2-1 LIFE CYCLE AND RELATED USERS OF WORKFLOW	16
FIGURE 2-2 OVERVIEW OF WORKFLOW MANAGEMENT SYSTEM	17
FIGURE 2-3 EXAMPLE OF WORKFLOW SEQUENCE.....	18
FIGURE 2-4 OVERVIEW OF CROSS ORGANIZATIONAL WORKFLOW.....	20
FIGURE 2-5 SYSTEM ARCHITECTURE OF CROSS ORGANIZATIONAL WORKFLOW ON BLOCKCHAIN (I)	26
FIGURE 2-6 SYSTEM ARCHITECTURE OF CROSS ORGANIZATIONAL WORKFLOW ON BLOCKCHAIN (II)	27
FIGURE 2-7 SYSTEM COMPONENTS IN BLOCKCHAIN NODE OF SYSTEM ARCHITECTURE (II).....	28
FIGURE 2-8 TRANSACTION FLOW OF CREATE/UPDATE PROCESSING.....	30
FIGURE 2-9 TRANSACTION FLOW OF REFERENCE PROCESSING	31
FIGURE 2-10 IMPLEMENTED USE CASE OF CROSS ORGANIZATIONAL WORKFLOW	32
FIGURE 2-11 ATTACK TARGET ASSET FROM DATA INTEGRITY PERSPECTIVE	34
FIGURE 2-12 ATTACK SCENARIO THROUGH WORKFLOW LIFE CYCLE	35
FIGURE 3-1 TRANSACTION FLOW OF WORKFLOW EXECUTION PROCESSING.....	40
FIGURE 3-2 DATA STRUCTURE OF STATE DB (A).....	42
FIGURE 3-3 DATA STRUCTURE OF STATE DB (B).....	42
FIGURE 3-4 READ/WRITE SET IN DATA STRUCTURE (A).....	43
FIGURE 3-5 READ/WRITE SET IN DATA STRUCTURE (B).....	44
FIGURE 3-6 TRANSACTION FLOW OF THE SDV WORKFLOW EXECUTION ENGINE.....	46
FIGURE 3-7 RESPONSE TIME BY DATA SIZE	52
FIGURE 4-1 TRANSACTION FLOW USING PDC OF HYPERLEDGER FABRIC	57
FIGURE 4-2 DATA MODEL FOR SDVCD WORKFLOW EXECUTION ENGINE	59
FIGURE 4-3 TRANSACTION FLOW IN PROPOSED ARCHITECTURE	60
FIGURE 4-4 EVALUATED USE CASE OF CROSS ORGANIZATIONAL WORKFLOW	63
FIGURE 4-5 DATA SIZE OF EACH DATA STORE	65

FIGURE 4-6 TOTAL STORAGE SIZE 66

FIGURE 4-7 RESPONSE TIME BY DATA SIZE 67

FIGURE 4-8 BREAKDOWN OF PROCESSING TIME 68

表目次

TABLE 2-1 EXAMPLE OF WORKFLOW DEFINITION 18

TABLE 2-2 EXAMPLE OF TRANSACTION RECORD 19

TABLE 2-3 REQUIREMENTS AND SPoTs IN WORKFLOW DEFINITION PHASE 22

TABLE 2-4 REQUIREMENTS AND SPoTs IN WORKFLOW EXECUTION PHASE 23

TABLE 2-5 REQUIREMENTS AND SPoTs IN AUDITING PHASE..... 24

TABLE 2-6 COMPARISON OF SECURITY BETWEEN TWO SYSTEM ARCHITECTURES..... 37

TABLE 3-1 RECORD SIZE (BYTES) 48

TABLE 3-2 DATA SIZE (BYTES) 50

TABLE 3-3 SPECIFICATIONS OF EVALUATION ENVIRONMENT FOR SDV 52

TABLE 4-1 DATA MODEL OF CHANNEL STATE IN KEY VALUE STORE..... 61

TABLE 4-2 SPECIFICATIONS OF EVALUATION ENVIRONMENT FOR SDVCD 64

TABLE 4-3 COMPARISON OF DATA SIZE BETWEEN SDV AND SDVCD (1) (BYTES) 71

TABLE 4-4 COMPARISON OF DATA SIZE BETWEEN SDV AND SDVCD (2) (BYTES) 71

第1章

緒論

1.1 研究の背景

情報技術の進展に伴い、企業におけるビジネスプロセスは情報システムによって管理されるようになってきた。ソフトウェア開発を目的に開発されたモデリング言語である UML (Unified Modeling Language) [1]をベースに、ビジネスプロセスの記述向けに拡張した記法 [2]や、BPMN(Business Process Modeling Notation) [3]等のモデリング言語を用いてビジネスプロセスを記述し、SOA (Service Oriented Architecture) [4]や EAI(Enterprise Application Integration) [5]等のアーキテクチャを用いて ERP(Enterprise Resource Planning)システムと接続することにより、複数のシステムをまたがるビジネスプロセスを柔軟に設計・実装するとともに、実行状態や実行結果を効率的かつ厳正に管理する方法が一般的となっている [6] [7] [8]。BPMN 等のモデリング言語で記述されたビジネスプロセスを管理する手法は、ビジネスプロセス管理 (Business Process Management) [9]と呼ばれ、この手法を実装した情報システムはビジネスプロセス管理システム(Business Process Management System, 以下 BPMS) [10]として広く普及している [11]。

ビジネスプロセスの構成要素であるワークフローに対しては、ワークフロー管理システム [12] [13]が提供されており、ワークフローの定義や実行、監査を効率的かつ厳正に行うことが可能となっている [14] [15] [16]。ワークフロー管理システムを用いたワークフロー処理の例を Figure 1-1 に示す。ワークフロー処理は、起案や変更、承認、却下など、役割の異なる処理者が行う複数のトランザクションから構成される。ワークフロー管理システムでは、予め登録されたワークフロー定義に基づいて、ワークフロー実行エンジンが処理対象者に処理依頼を行い、処理結果に基づいてワークフローの状態を更新することでワークフローの実行状態を管理する。

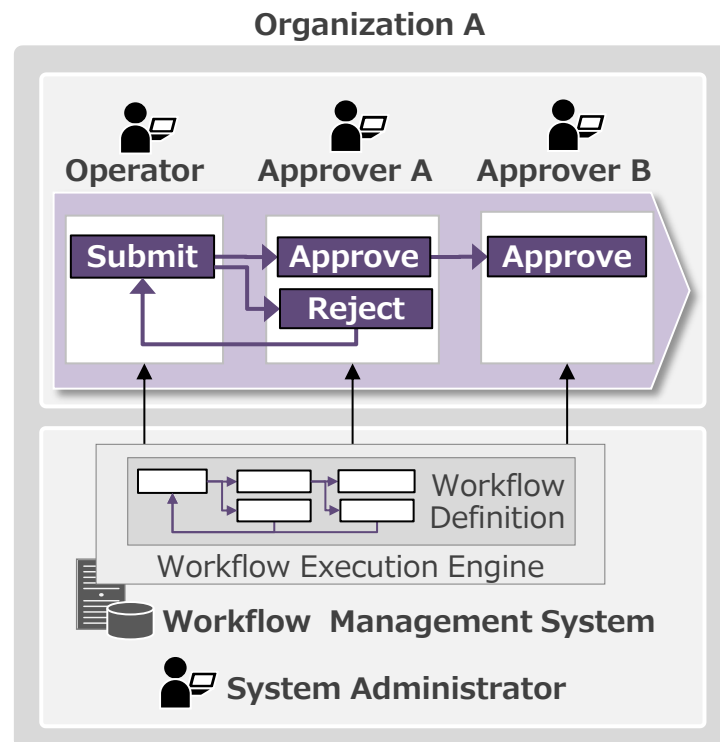


Figure 1-1 Example of workflow

しかしながら、このようなシステムや手法は、単一組織内のワークフローを対象としており、ワークフロー管理システム同士の相互運用性に課題がある [17]。このため、組織や企業をまたがる組織間のワークフローは、依然として文書を用いて手作業で処理されている。例えば、Figure 1-2 に示すように、代表的な組織間ワークフローである契約締結プロセスでは、組織内の稟議で承認された契約内容を相手の組織に提案したのち、訂正、署名などのプロセスにおいて複数回のやり取りが発生し、その都度電子メールや紙の文書として送受信されている。このため、組織間ワークフローの処理には多大な時間と労力がかかっている。しかし、既存のワークフロー管理システムは、Figure 1-1 に示した通り、単一組織内に閉じたシステム構成となっており、他組織からのアクセスを受け付ける機能や、他組織に情報を送信する機能は備えられていないため、組織間のワークフローには適用できない。

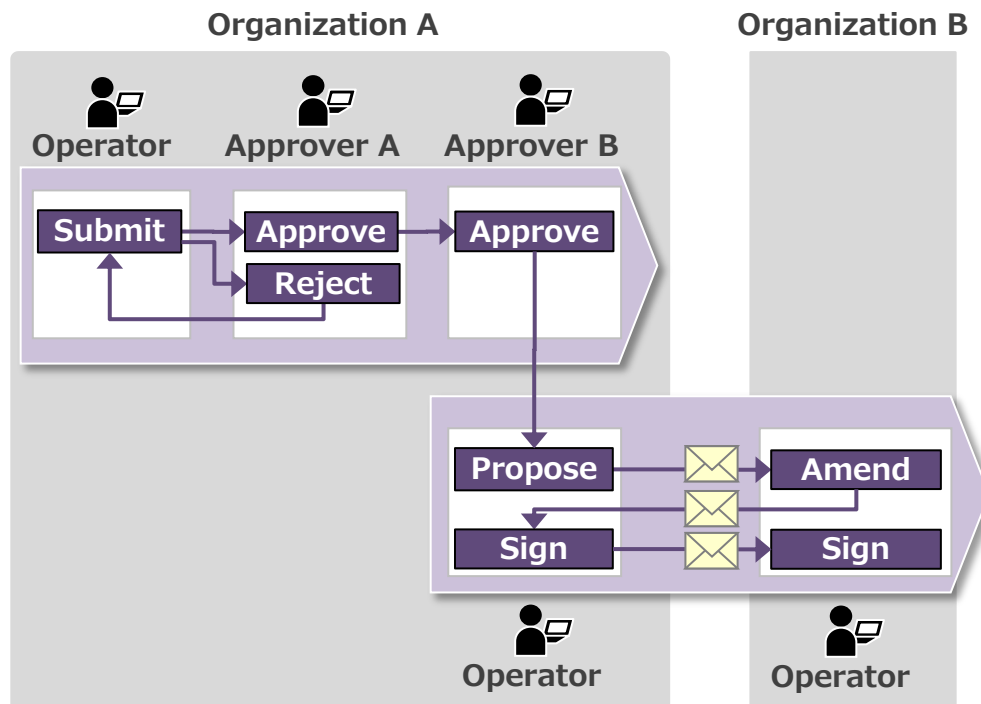


Figure 1-2 Example of cross organizational workflow

これに対し、WWW, CORBA [18], XML [19], Java などの分散処理技術を活用し、異なる組織間のシステムを連動させる試みが幾つか行われている。例えば、商用ワークフロー管理システムを連動させることで、企業間のワークフローシステムを実現する取り組みが行われており [20] [21] [22], WfMC(Workflow Management Coalition)¹ によるデータフォーマットの標準化などと併せて検討が進められている [23] [24] [25]。しかし、それぞれのワークフロー管理システムはそれぞれの企業が管理するため、相互の信頼が十分でない組織間のトランザクションにおいて、ワークフローのプロセスやデータが改ざんされていないことを保証する方法に課題が残る。例えば、ワークフローの実行プロセスにおいて、ある企業が承認したという記録が他の企業のワークフロー管理システムで改ざんされた場合、どちらのワークフロー管理システムの記録が正しいかを確認する方法はない。さらに、ワークフローにはワークフロー定義から実行、監査までのライフサイクルがある。このため、例えば組織間ワークフローの定義そのものが改ざんされ、企業間で合意した通りのプロセスが実行されない、実行後に実行履歴が改ざんされ、監査時に齟齬が発生する、などの問題が発生する可能性もある。このため、単に実行プロセスだけでなく、ワークフロー定義から実行、監査にわたるライフサイクル全体

1 Workflow Management Coalition, "Workflow Management Coalition", <https://wfmc.org>

にわたって、相互の信頼が不足する組織間で安全にワークフローを管理・実行する方法が必要となる。

このような分散環境下での安全なトランザクション処理技術として、ブロックチェーン技術がある。ブロックチェーンは、元々ビットコイン [26]のプラットフォーム技術として考案されたものであり、複数のトランザクションをまとめたブロックと呼ばれるデータのハッシュ値を、次のブロックに含めるハッシュチェーンと呼ばれるデータ構造により、過去のトランザクションデータの改ざんを困難にするとともに、トランザクションの確定を特定の参加者に依存せず、複数の参加者間での合意に基づいて行う分散合意形成技術により、特定の中央管理者に依存しない、分散型による信頼できるトランザクション処理を実現している。ブロックチェーン技術は、決済トランザクションを分散型で実行するための基盤技術として開発されたものであり、主に単一の決められた処理を繰り返し実行するような業務を対象としている。

従来、前述のような相互の信頼が不足する組織間での取引では、金融機関などの仲介者が介在することにより、リスクを最小化してきた。これに対して、ブロックチェーンの重要な特徴の一つである、改ざんが困難な情報の記録・共有機能を用いることにより、相互の信頼が不足する組織間で信頼できる記録を共有することが可能となり、仲介者が不在でも取引リスクを低減することができる [27]。このようなブロックチェーンの特徴を活用し、貿易取引に関わる文書を輸出者と輸入者の間で安全に共有する方法 [28]や、農産物等のサプライチェーンにおけるトレーサビリティ情報をサプライチェーン参加企業間で共有する方法 [29] [30]、建設プロジェクトにかかわる文書やデータを関係者間で安全に共有する方法 [31] [32]などが提案されている。また、このような仕組みを、上記のようなサプライチェーン管理や貿易金融等のビジネスケースに適用し、実現可能性を評価するための多くの実証実験が行われている [33] [34] [35]。

このような適用領域の拡大に伴い、ブロックチェーンの技術的な課題を解決する様々な研究が行われてきた。ブロックチェーンの主要な技術的課題として、スケーラビリティ、セキュリティ、データの不可逆性への対応がある [36] [37] [38]。

まず、一つ目の課題はスケーラビリティである。ブロックチェーンでは、ブロックの生成処理と分散合意形成処理がトランザクション処理量の制約となる。ブロックチェーンにおけるトランザクションは、複数のトランザクションをまとめたブロックを生成した段階で確定するため、このブロックのサイズによって時間あたりのトランザクション

処理量が決定される。その際、複数ノード間でブロックデータが正しく生成されたことを確認する合意形成処理を行うため、このアルゴリズムも処理性能に影響する。特に、ビットコインやイーサリアム [39]などの暗号資産を扱うパブリックブロックチェーンでは、不特定多数のノードが自由にブロックチェーンネットワークに参加できるため、悪意のある参加ノードが一定数存在する前提で厳格な合意形成アルゴリズムを用いる必要がある。このような合意形成処理は、処理性能に大きな影響を与える。

このような問題に対し、より高速かつ安全な合意形成アルゴリズムによって処理性能を向上させる取り組みが行われている [37]。特に、近年では合意形成処理によるエネルギー消費量が問題となっており、これを削減することを併せて考慮する方式が提案されている。また、同一ネットワークで処理できるトランザクション数を増大させる方法として、レイヤー2と呼ばれるオフチェーンでトランザクションを処理し、その結果をブロックチェーンに記録する方法が提案されている [40]。このような方法によれば、ブロックチェーン自体の処理量の制約の範囲内で、トランザクション処理量を増大させることが可能となる。一方で、参加ノードを限定するプライベートブロックチェーンにおいては、ノード保有組織間にある程度の信頼関係がある前提で、より簡易な合意形成アルゴリズムを用いることができる。また、ブロックチェーンの他に、キーと値のペアでデータを保存し、高速なアクセスを可能にするデータベースであるキーバリューストアを備え、ブロックチェーンと同時に記録していく構成が提案されている。これらの方法により、処理量や処理性能の問題が比較的起こりにくくなっている。

次に、セキュリティの課題がある。分散合意形成処理については、上述の通り、セキュリティや処理性能、エネルギー消費量の観点で様々な検討がなされてきた。その後、ブロックチェーン上で柔軟なプログラム処理を実行する仕組みであるスマートコントラクト [41]が提案されると、スマートコントラクトの脆弱性を突いた様々な攻撃が問題となった。これに対し、コードの検証やデザインパターン、セキュリティテストにより、安全性を確保するための対応策が提案されている [42]。ブロックチェーン上のデータについては、例えばビットコインでは取引主体の匿名性が確保される一方、取引データ自体は開示される。これに対し、ゼロ知識証明を活用してプライバシー保護を実現する方式 [43]や、オフチェーンストレージを活用し、取引内容自体は外部に保存してアクセスを制御する方式が提案されている [44]。また、参加組織を限定するプライベートブロックチェーンを用いた業務用途が拡大するにつれて、取引データを秘匿化するニーズ

が高まってきている。これに対して、開示範囲を限定する機能やユーザのアクセス権を制御する方式が提案されている。

最後に、データの不可逆性による問題がある。ブロックチェーンでは、一度記録されたデータは削除や変更ができないという特徴がある。このため、GDPR (General Data Protection Regulation)に代表される個人情報保護規制への対応が困難であるという問題がある。また、データの削除ができないため、ブロックチェーンに記録されるデータ量が増大し続けるという問題がある。これらの問題に対して、スケーラビリティの課題への対応策として挙げたオフチェーンにより、ブロックチェーン上のデータ量を削減する方法や、IPFS(InterPlanetary File System)²のような分散ストレージシステムと連携し、機微情報や大量データについてはオフラインストレージに格納し、そのハッシュ値だけをブロックチェーンに格納する方法などが提案されている [44]。

このように、適用領域の拡大に伴い、ブロックチェーンの技術的な成熟が進んでいるが、ブロックチェーンは、基本的には決済処理のような予め定義・実装された固定的な処理プロセスを対象に、単一トランザクション毎の処理結果を、改ざんが困難な記録方式で全ての参加者にオープンに共有する技術である。一方で、本研究が対象とするワークフロー処理は、①処理プロセスの定義自体に変更が起こりうる可変的な処理プロセスであり、プロセスの定義から実行、監査に至る一連のライフサイクルが存在する。このため、参加組織全てが信頼できる組織間ワークフロー管理システムを実現するためには、実行フェーズだけでなく、これらのライフサイクル全体に渡って、特定の組織に依存せずに処理を実行する方法が必要となる。また、②ワークフローの処理プロセスは複数トランザクションの組み合わせで構成されるため、単一トランザクションの処理結果だけでなく、その処理プロセスにも改ざんがないこと、つまり予め合意したプロセス通りに実行されていることが要求される。さらに、③ワークフロー処理は企業間取引等を対象とするため、全ての情報を全ての参加者に開示することはできず、情報の開示範囲を限定する必要がある。

まず、①に挙げた通り、ワークフロー管理システムにおいては、運用中に新たなワークフローが定義される場合がある、ワークフローの定義が変更になる場合がある等、可変的な処理である。このため、信頼性の高い組織間ワークフロー管理システムを実現するためには、定義から実行、監査に至るワークフローのライフサイクル全体に渡って、

2 IPFS, "IPFS", <https://ipfs.io>.

特定のプロセスやデータを単一の組織に依存する状態を排除することが必要となる。このような状態は単一信頼点（Single Point of Trust, 以下 SPoT） [45]と呼ばれ、悪意のある組織が関連プロセスやデータを改ざんする攻撃対象となりうる。ブロックチェーンでは、あらかじめ定義された処理プロセスを実行する際に、参加ノード間で合意形成を行った後にトランザクションを確定することにより、SPoTを排除している。しかし、上記のようなワークフローのライフサイクルを考慮した場合、あらかじめ定義されたとおりワークフローを処理する実行フェーズだけでなく、ワークフロー定義を作成・合意する定義フェーズから、実行フェーズを経て、ワークフロー実行後の処理履歴を第三者が確認する監査フェーズに至るまで、ワークフローのライフサイクル全体を通じて発生しうる全ての SPoT を排除する必要がある。しかし、定義や実行、監査といった各フェーズの処理は、それぞれ操作者が異なるため、異なるアプリケーションプログラムで処理される。このため、フェーズ間を移行する際に、複数のアプリケーションプログラムを跨ったデータの移行作業が発生する。通常、このような作業は、特定のシステム管理者やワークフロー管理者に依存するため、SPoT となる。例えば、SPoT を排除した状態でワークフロー定義を合意したとしても、その定義を実行環境に実装する作業は特定のシステム管理者に依存するため、SPoT となる。このように、ワークフローのライフサイクルにおいて、処理プログラムや操作者、処理対象となるデータが変化する中で、発生しうる全ての SPoT を排除する方法を立案することは困難である。

次に、②に挙げたように、ワークフロー処理では、処理結果だけでなく、処理プロセスにも改ざんがないことを保証する必要がある。例えば、ワークフロー処理では、複数の承認者が存在する場合に、他の承認者が承認済みかどうかや、他の承認者が追加した情報や変更した情報を確認したうえで、次の承認者が承認を行うなど、関連するトランザクションの処理結果を正しく参照する必要がある。しかし、ブロックチェーン技術は、単一のトランザクションについて、書き込みデータを処理ノード間で合意しながら書き込むことで改ざんを防止する技術であり、その際に、関連する他のトランザクションの処理結果など、参照したデータが正しいことを保証することは困難である。このため、ブロックチェーン技術を単に適用しただけでは、複数のトランザクションから構成されるワークフロー処理において、処理プロセスが正しく実行されていることを保証することは困難である。

最後に、③の特徴について、本研究が対象とする組織間ワークフローにおいては、複

数の組織が参加するブロックチェーンネットワークを構成する必要があるが、それぞれのワークフローに含まれる組織グループは、ワークフローのケースごとに異なる。つまり、あるワークフローに含まれる組織は、別のワークフローには含まれない可能性がある。しかし、ブロックチェーン基盤では、基本的にすべてのワークフローデータが全てのブロックチェーンノードに保存されるため、各ノードには、自組織が含まれていないワークフローのデータも保存され、データの機密性が問題となる。また、全てのデータがノード数分重複して格納されるため、システム全体のデータ容量も問題となる。このように、参加組織が拡大した場合に、データの機密性や蓄積されるデータ量の増大の問題を解決する必要があるが、ワークフロー毎に変化する参加組織に対する情報の開示範囲を限定しつつ、データ容量を抑制する方法を実現することは容易ではない。

以上から、本研究では組織間ワークフローにおける以下の課題を解決することを目的とする。

- (1) ワークフローのライフサイクル全体で SPoT を排除する組織間ワークフロー管理システムのアーキテクチャの実現
- (2) ワークフローの実行フェーズにおけるデータとプロセスの改ざん耐性を備えた組織間ワークフロー実行エンジンの実現
- (3) 組織間ワークフロー管理システムの大規模化における情報開示範囲の限定とデータ容量増大への対応

各課題の位置づけを Figure 1-3 に示す。

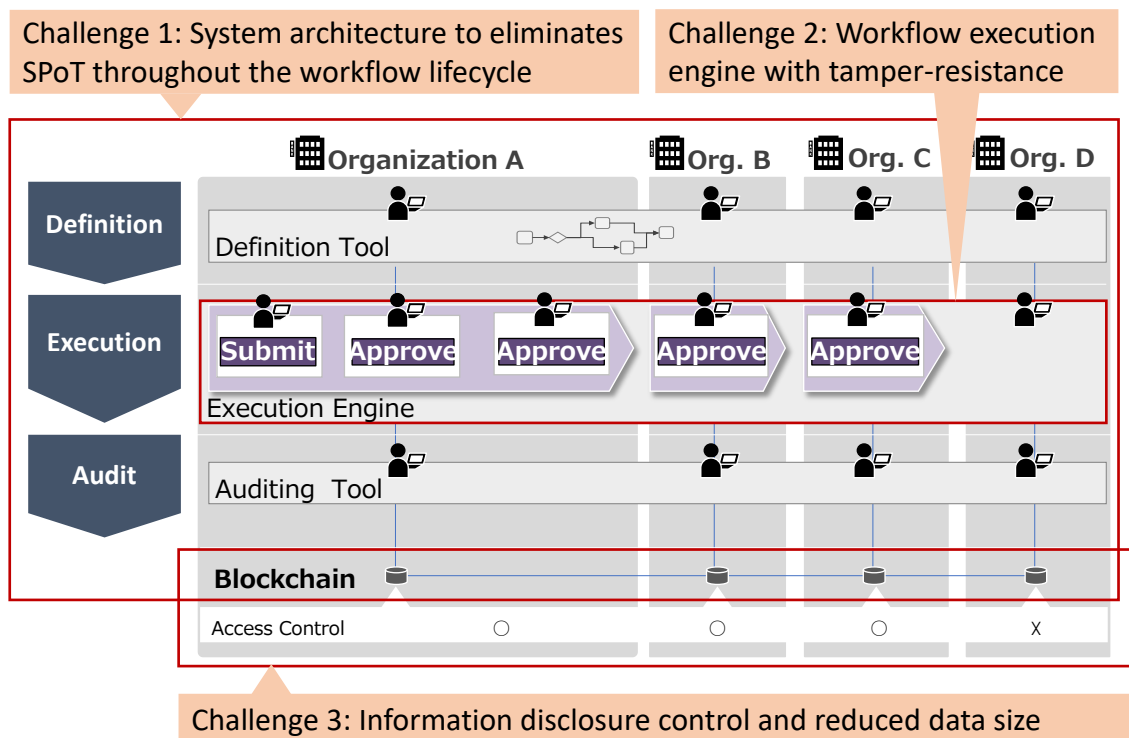


Figure 1-3 Challenges of this thesis

1.2 関連研究

1.2.1 組織間ワークフロー管理システムのアーキテクチャに関する

関連研究

前述したサプライチェーン管理や貿易取引のようなアプリケーションを実現するための組織間ワークフロー管理システムの実現に向けて、ブロックチェーン技術を活用したいくつかのアーキテクチャが提案されている。例えば、文献 [46]では、組織間でのビジネスプロセスにおける実行状態やデータをブロックチェーン上に共有することで、組織間ビジネスプロセスを安全に実行する方法が示されている。また、文献 [47]では、参加者間の情報の透明性と改ざん防止を保証するために、ワークフローのようなビジネスプロセスを定義通りに実行するための BPMS と、その実行記録を安全に蓄積するためのブロックチェーン基盤を統合するアーキテクチャが提案されている。しかし、ブロックチェーンが分散型の信頼できるデータ共有基盤を提供するのに対し、BPMS とブロックチェーン間の相互通信や BPMS は集中管理となっており、これらが SPoT となる。

また、ワークフローには、定義から実行、記録の保管や監査といった一連のライフサイクルがある。実行フェーズで用いられるワークフローの定義は、各組織のビジネス要件の変化に基づいて動的に変更可能である必要がある。ブロックチェーン上では、ワークフロー処理をスマートコントラクトによって実行するが、ワークフロー定義を変更する場合にはシステム管理者による運用作業が必要となる。このようなシステムの運用作業についても、スマートコントラクトを用いて実行することにより、SPoT を排除する方法が近年提案されている [48]。しかし、ワークフローの定義、実行、監査はそれぞれ操作者が異なるプロセスであり、フェーズ間の移行はスマートコントラクトの入れ替えだけでは対応できない。このようなワークフローのフェーズ間の移行を含めて、組織間ワークフローのライフサイクル全体に渡って SPoT を排除し、全ての参加組織が信頼できるワークフロー処理システムを実現するためのアーキテクチャは示されていない。

1.2.2 組織間ワークフロー実行エンジンの関連研究

ワークフローの実行フェーズにおける SPoT を排除する方法として、ワークフロー処理を行う BPMS の機能をブロックチェーン上で動作する分散処理型のプログラムであるスマートコントラクトとして実装することにより、相互の信頼が不足する組織間でワークフローが定義通りに処理されることを保証する方法が提案されている [49] [50]。また、文献 [51] [52]では、ブロックチェーン基盤としてイーサリアムを用い、貿易金融のビジネスプロセスを仲介者無しで実現する方法が示されている。さらに、文献 [53]や文献 [54]では、イーサリアム上のスマートコントラクトでワークフロー実行エンジンを実装する方法が示されている。しかし、ワークフローの複数のステップ間で正しくデータを参照して処理を行ったかなど、複数のステップからなるワークフロー処理の特徴を考慮した安全なデータ管理と実行エンジンの実装方法は示されていない。

1.2.3 組織間ワークフロー管理システムにおける情報開示範囲の限定とデータ量増大への対応に関する関連研究

ブロックチェーン基盤上でビジネスプロセスを実行する際のデータと処理の機密性の問題は文献 [55]で指摘されており、ゼロ知識証明などの暗号技術の活用 [56] [57]や

TEE(Trusted Execution Environment)の活用 [58] [59]が提案されている。しかし、これらはスマートコントラクトとデータの機密性を処理依頼先に対して確保するために利用する方法であり、ワークフロー処理においてはワークフロー参加組織間では秘匿する必要はなく、参加組織外に対する機密性が必要とされているため、活用できない。

ブロックチェーンネットワーク上のデータの漏洩を抑制するために、特定のノードにのみ生データを保存し、参照権限のないノードにはハッシュ値のみを格納することにより、データの機密性を確保する方法として、コンソーシアム型ブロックチェーン基盤の一つである Hyperledger Fabric³では、Private Data Collection⁴と呼ばれる機能が提供されている。しかし、データの整合性が保証されるのは単一のトランザクションに対してのみであり、ワークフロー案件を構成する一連のトランザクションに対しては保証されない。

また、ブロックチェーン上のデータ量を削減する方法として、IPFSのような外部ファイルシステムにデータ本体を保存し、ブロックチェーン上にはハッシュ値のみを格納する方法が提案されている [60]。しかし、このような方法をワークフローに適用した場合、ワークフローデータの改ざんを検知することは出来るが、ワークフローデータの管理を第三者に依存するため、改ざんそのものを防止したりデータの削除を防止したりすることはできない。別の方法として、トランザクションの変更点をまとめてブロックチェーンに記録することで、データ量を削減したノードを構成する Block Summarization と呼ばれる技術が提案されているが [61]、ステップごとの記録が必要なワークフローには適さない。

このように、ワークフローデータの一貫性を確保しつつ、データの機密性とデータ量の削減に取り組んだ先行研究は見つからない。

1.3 研究の方針

前節までに述べた課題を踏まえた、本研究の方針を説明する。

(1) SPoT を排除した組織間ワークフロー管理システムのアーキテクチャ

³ The Linux Foundation, "Hyperledger Fabric - Hyperledger", <https://www.lfdecentralizedtrust.org/projects/fabric>.

⁴ The Linux Foundation, "Private Data - Hyperledger Fabric Docs", <https://hyperledger-fabric.readthedocs.io/en/latest/private-data/private-data.html>.

複数のフェーズから構成されるワークフローのライフサイクルにおいて、各フェーズ内の処理における SPoT を排除するだけでなく、フェーズ間を移行する際のアプリケーションプログラムを跨るデータの受け渡しにおいても、SPoT を排除する必要がある。これに対し、各フェーズの処理をブロックチェーン基盤上のプログラム処理機能であるスマートコントラクトで実装するだけでなく、これらのスマートコントラクト同士を連携させることで、アプリケーションプログラム間のデータ移行作業そのものを排除する。定義から実行フェーズへの移行においては、ワークフロー定義スマートコントラクトを用いて合意したワークフロー定義を、ワークフロー実行スマートコントラクトが参照可能な状態で直接実行環境に格納する。このようなアーキテクチャにより、従来、特定のシステム管理者が行っていたフェーズ間の移行作業自体をなくし、フェーズ間の移行における SPoT を排除する。

(2) ブロックチェーンを活用した組織間ワークフロー管理システムの実行エンジン

組織間ワークフローの実行エンジンであるスマートコントラクトの実装においては、ワークフロー処理の特性を考慮し、一連の処理プロセスを通じてデータの改ざんがないことを保証する必要がある。これに対し、ワークフロー処理が処理ステップごとのトランザクションの組み合わせであることに着目し、ステップ間でのデータの一貫性を保証するためのデータ構造とスマートコントラクトの実装方式を提案する。また、このデータ構造と実装方式を採用した場合に問題となる可能性のあるデータ量の増大と処理性能の低下について、具体的なワークフロー例を実装し、評価する。

(3) 情報開示範囲の限定と冗長性のないデータ構造による組織間ワークフロー管理システム

複数組織が参加するブロックチェーンネットワークにおいて、データの開示範囲を限定するための機能として、Hyperledger Fabric では Private Data Collection が提供されている。この機能を組織間ワークフローに適用する場合、(2)に示したように、ワークフローのステップ間でのデータの一貫性も同時に保証する必要がある。これに対し、過去の全ステップのデータを毎ステップで逐次比較して格納する方式ではなく、Private Data Collection において他ノードに格納されるトランザクションのハッシュ値を取得し、これを用いて全ステップのデータを比較したうえで差分データのみを格納する方式により、格納データの冗長性を排除する方式を提案する。

1.4 本論文の構成

本論文は全 5 章から構成され、第 2 章以降を以下のように構成する。

第 2 章では、文献 [62] [63] に基づき、組織間での利用を想定したコンソーシアム型ブロックチェーンを用い、組織間ワークフロー管理におけるライフサイクル全体にわたって存在する SPoT、特に、ワークフローの定義フェーズから実行フェーズに移行する際に存在する SPoT を排除するシステムアーキテクチャを提案し、攻撃シナリオに基づく改ざんリスクの評価により、提案方式のセキュリティ評価を行う。

第 3 章では、文献 [64] [65] に基づき、組織間ワークフロー管理システムのアーキテクチャで動作するワークフロー実行スマートコントラクトの実装方式として、ワークフロー案件を通じてデータの改ざんを防止するワークフロー実行エンジンの実装方式を提案し、処理性能とデータ容量の観点で実用性を評価する。

第 4 章では、文献 [66] [67] に基づき、最小限のデータでワークフローデータの整合性を保証し、参加者間での情報の開示範囲を限定する組織間ワークフロー実行エンジンを提案し、処理性能とデータ容量の観点で実用性を評価する。

第 5 章では、総論として本研究で得られた成果を要約し、今後の課題を述べる。

第2章

SPoT を排除した組織間ワークフロー管理 システムのアーキテクチャ

2.1 緒言

本章では、ワークフローの定義から実行、監査に至るワークフローのライフサイクル全体を通して SPoT を排除する、組織間ワークフロー管理のシステムアーキテクチャを提案する。提案アーキテクチャにより、組織間ワークフローにおける処理プロセスとデータの改ざんリスクを回避することを狙いとする。

従来のワークフロー管理システムでは、ワークフローのライフサイクルである定義、実行、監査のそれぞれの機能を、集中型システムで提供している。互いに信頼関係が不足する複数組織間でのワークフロー管理を安全に実現するためには、これらの機能を、SPoT を排除した構成で実現する必要がある。これに対し、ブロックチェーン基盤を用いることにより、それぞれの処理プロセスを、SPoT を排除した構成で実行することができる。しかし、例えば、ワークフロー定義のプロセスにおいて SPoT を排除したとしても、通常の方法では、合意した定義を実行環境に実装する処理において特定システム管理者が介在するため、この処理が SPoT となり、定義自体が改ざんされるリスクがある。本章では、ワークフローの各フェーズをブロックチェーン上のスマートコントラクトを用いて実装するとともに、各フェーズの処理を行う複数のスマートコントラクトを連動させることにより、フェーズ間を移行する際のデータ連携をシステム管理者に依存せずに実現するシステムアーキテクチャを提案する。このアーキテクチャにより、各フェーズ内の処理だけでなく、フェーズ間の移行も含めて、ワークフローのライフサイクル全体に渡って全ての SPoT を排除する。

以下、まず 2.2 節で従来のワークフロー管理システムの構成を説明したうえで、組織間ワークフローのライフサイクルにおける SPoT を洗い出し、組織間ワークフロー管理システムの要件を整理する。この要件に基づき、2.3 節で、組織間ワークフローにおけ

る SPoT を排除した組織間ワークフロー管理システムのアーキテクチャを提案し、提案アーキテクチャを用いた場合の処理フローを説明する。その後、2.4 節において、ワークフローのライフサイクルに沿った攻撃シナリオに基づくセキュリティ評価の結果について述べる。

2.2 組織間ワークフロー管理システムの要件

2.2.1 ワークフロー処理の概要

本節では、ワークフロー処理、およびその実行を管理するワークフロー管理システムの概要について説明する。ワークフローは、企業のビジネスプロセスの一部であり、提案の作成、修正、却下、承認、最終的な提案の合意など、複数のステップから構成される。ワークフローには、定義、実行、監査の 3 つのフェーズから構成されるライフサイクルがある。Figure 2-1 は、ワークフローの 3 つのフェーズと、各フェーズにおける関連ユーザを示している。最初のフェーズは定義フェーズであり、ワークフロー管理者は、ビジネスルールに基づいてワークフロー定義を作成する。次に、実行フェーズでは、ワークフロー定義に基づき、ワークフロー処理者がワークフローを開始、処理する。ワークフローの実行履歴はワークフロー管理システムに記録され、監査フェーズにおいて監査者により参照される。ワークフロー管理システムは、これらのワークフロー処理フェーズのライフサイクル全体を管理するシステムであり、システム管理者によって管理される。

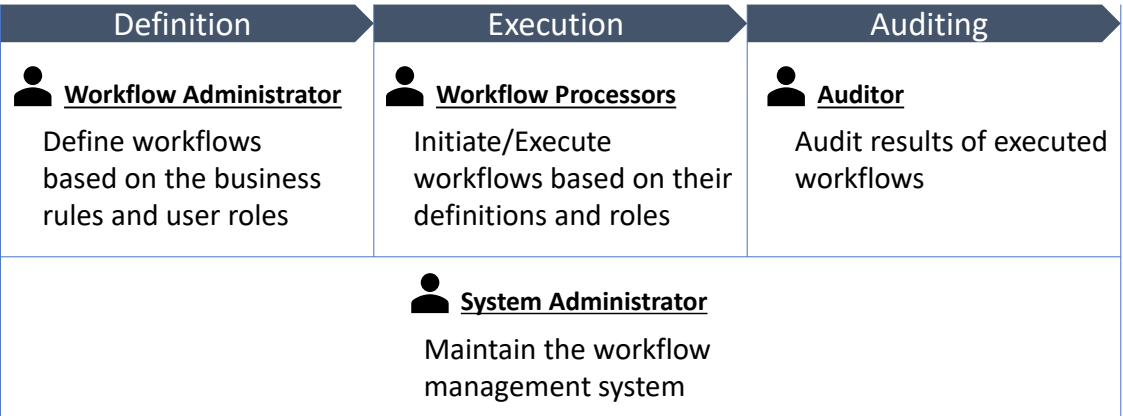


Figure 2-1 Life cycle and related users of workflow

Figure 2-2 に従来のワークフロー管理システムの概要を示す。ワークフロー管理シス

テムは、ワークフロー処理者、ワークフロー管理者、および監査者用のクライアントアプリケーション、ワークフロー管理者用のワークフロー設計ツール、ワークフローの状態管理機能を用いてワークフロー定義に基づくワークフロー実行処理を制御し、実行結果を監査証跡として保存する、ワークフロー実行エンジンから構成される。定義フェーズでは、ワークフロー管理者がワークフロー設計ツールを用いてワークフロー定義を作成し、ワークフロー管理システムに格納する。実行フェーズでは、ワークフロー実行エンジンの状態管理機能により、データベースに保存されたワークフローの状態を参照し、ワークフロー処理者の処理に基づいて更新する。実行結果は、実行記録管理機能によりデータベースに保存され、クライアントアプリケーションを通じて監査者への参照機能を提供する。

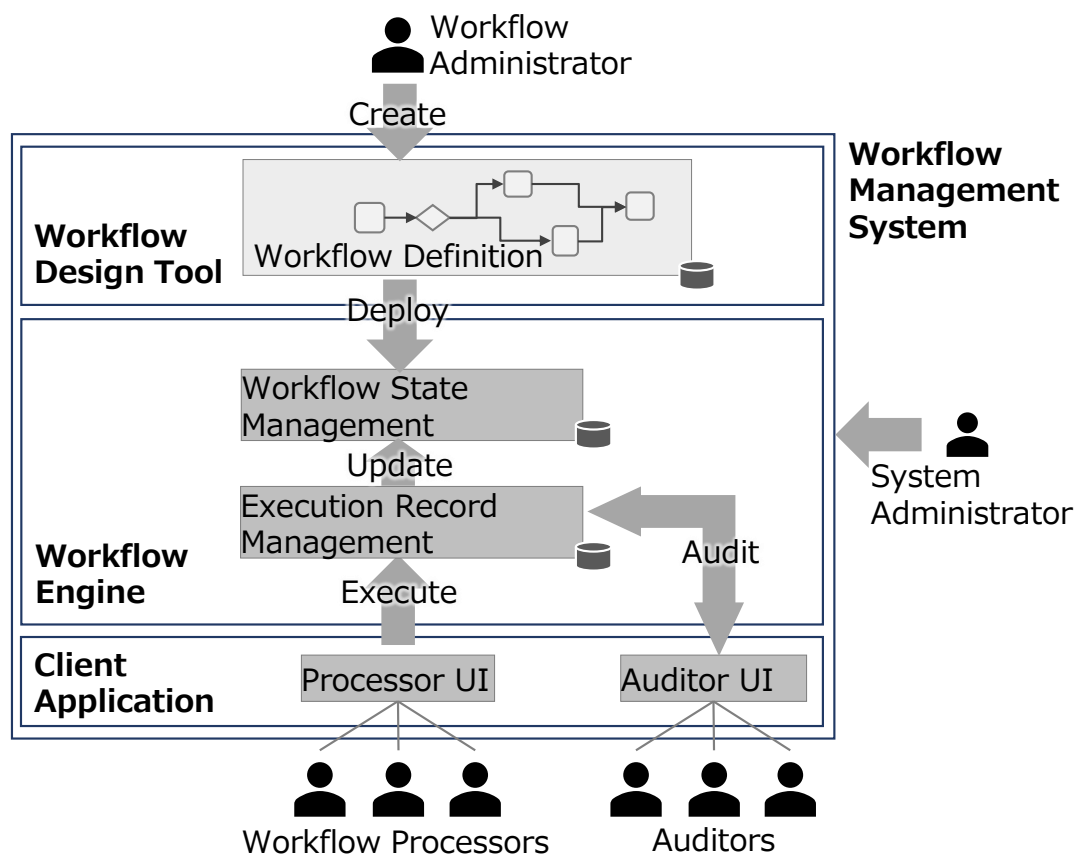


Figure 2-2 Overview of workflow management system

ワークフロー定義の構造には多くのパターンがあり [68], Table 2-1 に示すようなデータ構造として定義される。この例では、ワークフローは 5 つのステップから構成さ

れ、Figure 2-3 に示すようなワークフローのシーケンスは、前提条件(Prerequisite)と次のステップ(Next Step)フィールドに設定されたステップ間の関係として定義される。

Table 2-1 Example of workflow definition

Step	Operator	Operation	Prerequisite	Next Step
Initiate request	Proposer	Set item1 and item2	-	Update request
Update request	Approver1	Approve item1 and item2. Set item3 and item4	Initiate request	Approve request1 Approve request2
Approve request1	Approver2	Approve item1, item2, item3 and item4	Update request	Approve request3
Approve request2	Approver3	Approve item1, item2, item3 and item4	Update request	Approve Request3
Approve request3	Approver4	Approve item1, item2, item3 and item4	Approve Request1 Approve Request2	

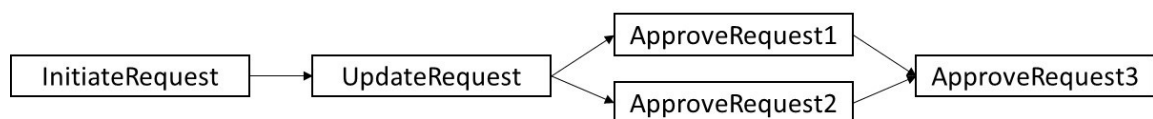


Figure 2-3 Example of workflow sequence

このワークフローが実行フェーズで処理されると、処理履歴は Table 2-2 の通り記録され、実行フェーズではワークフローのステータスを確認するために、また監査フェーズでは監査記録として利用される。

Table 2-2 Example of transaction record

Time	Operator	Step	Operation
8/9/2020 09:05	AAA	Initiate request	Initiate
8/9/2020 10:12	BBB	Update request	Approve
8/9/2020 11:23	CCC	Approve request1	Approve
8/9/2020 13:41	DD	Approve request2	Reject

従来のワークフロー管理システムは、業務プロセスの効率化と管理コストの削減を目的に、単一組織内での内部利用を想定して設計されている。従って、組織のシステム管理者が管理すべき単一のデータベースと単一のサーバ上で動作する。

2.2.2 組織間ワークフローの前提

組織間ワークフローは、Figure 2-4 に示されるように、内部および組織間のワークフローの組み合わせと考えることができる。組織間ワークフローの各ステップは、対応する組織内ワークフローに基づいて処理される。組織内ワークフローは、2.2.1 で説明した方法と同様に管理される。しかし、組織間ワークフローは、複数の組織のワークフロー管理者が組織間の合意に基づき定義し、複数の組織のワークフロー処理者が処理し、第三者である監査者が監査する。このため、組織内ワークフローとは異なる管理方法が必要となる。

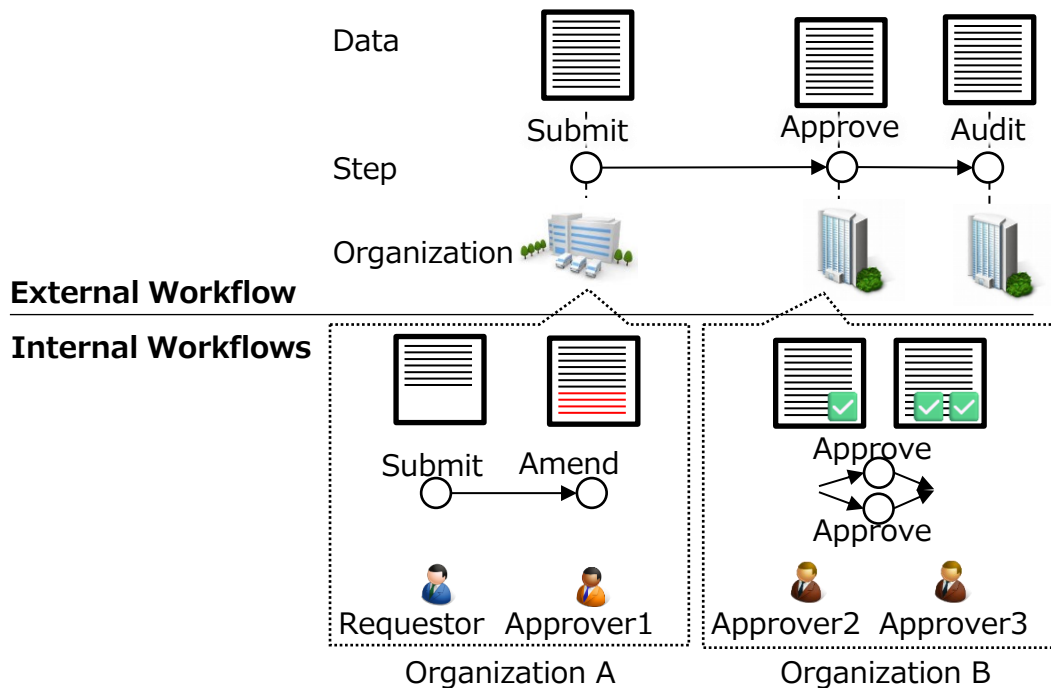


Figure 2-4 Overview of cross organizational workflow

2.2.3 組織間ワークフロー管理システムの要件

2.2.2 で述べたように、組織間ワークフローは、互いの信頼関係を前提にできない組織間で定義、処理、管理される。したがって、組織間のワークフローを安全に処理するためには、組織間で信頼できる処理やデータ管理を行うシステムが必要となる。

ワークフローの定義、実行、監査フェーズにおける処理プロセスにおいて、Figure 2-1 に示したユーザが、Figure 2-2 に示したワークフロー管理システム内の各データにどのような操作(CRUD(Create, Read, Update, Delete))を行うかを整理し、これらの処理プロセスを組織間ワークフローにおいて実行する際に求められる要件と、それぞれの処理プロセスで発生しうる SPoT を洗い出した結果を Table 2-3, Table 2-4, Table 2-5 に示す。

まず、定義フェーズでは、組織間ワークフローの参加組織のいずれかに属するワークフロー管理者がワークフロー定義を開始(Initiate)し、その後、複数の参加組織からの更新(Update)リクエストを処理したのち、ワークフロー定義が合意(Agree)される。さらに、合意された定義ファイルがシステム管理者によって実行環境に実装(Deploy)される。定義の開始、更新、合意プロセスにおいては、ワークフロー定義とその合意の記録が、

組織間で安全に保管されることが要件となる。したがって、定義ファイルや合意の記録が、特定の組織のワークフロー管理者やシステム管理者に依存する状態 [SPoT-1]を排除する必要がある。また、ワークフロー定義ファイルを実行環境へ実装するプロセスにおいては、最終的に合意したワークフロー定義を、改ざんなく実行環境に実装することが要件となる。したがって、この実装処理を特定のシステム管理者に依存する状態 [SPoT-2]を排除する必要がある。

次に、実行フェーズにおいては、定義フェーズにおいて合意したワークフロー定義に基づいて、参加組織のいずれかに属するワークフロー処理者がワークフローを開始(Initiate)し、その後、定義に従って各組織のワークフロー処理者が更新(Update)処理を行った後、ワークフローが完了(Close)する。この一連の処理において、ワークフロー処理が定義フェーズで合意した定義通りに実行されていること、及び実行ステータスと実行履歴が改ざんなく組織間で共有されていることが組織間ワークフローの要件となる。したがって、ワークフロー定義の管理を特定組織のシステム管理者に依存している状態 [SPoT-3]や、実行ステータスや実行履歴の管理を特定組織のシステム管理者に依存している状態[SPoT-4]を排除する必要がある。

最後に、監査フェーズにおいては、監査者が各参加組織からワークフロー定義と実行履歴のデータを収集(Collect)し、それらのデータが整合しているかどうかを確認(Audit)する。データの収集処理においては、定義ファイルが実行フェーズを通じて改ざんされていないこと、また実行フェーズで処理された結果である実行履歴が、ワークフロー完了後に改ざんされていないことが要件となる。したがって、複数の参加組織からのデータ収集を、特定の組織のシステム管理者に依存した状態[SPoT-5]を排除する必要がある。

Table 2-3 Requirements and SPoTs in workflow definition phase

Phase		Definition			
Process		Initiate	Update	Agree	Deploy
Data (<u>C</u> reate, <u>R</u> ead, <u>U</u> ppdate, <u>D</u> elete)	Workflow Definition	C	R, U	R, U	C, R
	Execution Record				
	Workflow Status				
Operator		Workflow Administrator	Workflow Administrator	Workflow Administrator	System Administrator
Requirements		The workflow definition and the record of agreements are stored securely			The workflow definition is deployed to the execution environment as agreed in the definition phase
SPoT		The workflow definition or the record of agreements are managed by single workflow administrator or single system administrator [SPoT-1]			The workflow definition is deployed by a single system administrator [SPoT-2]

Table 2-4 Requirements and SPoTs in workflow execution phase

Phase		Execution		
Process		Initiate	Update	Close
Data (<u>C</u> reate, <u>R</u> ead, <u>U</u> ppdate, <u>D</u> elete)	Workflow Definition	R	R	R
	Execution Record	C	R, U	R, U
	Workflow Status	C	R, U	R, U
Operator		Workflow Processors	Workflow Processors	Workflow Processors
Requirements		- The Workflow processes are executed based on the workflow definition as agreed in the definition phase. - The execution record and the workflow status are stored securely and shared among organizations.		
SPoT		- The workflow definition is managed by a single system administrator [SPoT-3] - The execution record or the workflow status are managed by a single system administrator [SPoT-4]		

Table 2-5 Requirements and SPoTs in auditing phase

Phase		Auditing	
Process		Collect	Audit
Data (<u>C</u> reate, <u>R</u> ead, <u>U</u> ppdate, <u>D</u> elete)	Workflow Definition	R	R
	Execution Record	R	R
	Workflow Status		
Operator		Auditors	Auditors
Requirements		The execution records are not falsified after the case was closed.	The execution records are consistent among organizations
SPoT		The data from multiple organizations are collected by a single system administrator [SPoT-5]	

組織間ワークフロー管理システムにおける改ざんリスクを回避し、全ての参加組織が信頼できるシステムを実現するためには、ワークフロー処理のライフサイクルを通じて、上記 5 つの SPoT を排除する必要がある。

2.3 SPoT を排除した組織間ワークフロー管理システムの概要

2.3.1 SPoT を排除した組織間ワークフロー管理システムのアーキテクチャ

2.2.2 で述べたように、組織間ワークフローは、組織内ワークフローと組織間ワークフローの 2 つの部分から構成される。Figure 2-2 のような従来のワークフロー管理システムは、幅広い企業で利用されている。従って、組織間ワークフロー管理システムは、既存の組織内ワークフロー管理システムとの接続性を考慮する必要がある。また、2.2.3 で述べた通り、組織間ワークフローについては、5 つの SPoT を排除する必要がある。

5 つの SPoT のうち、SPoT-1 と SPoT-4 は、それぞれ定義フェーズ、実行フェーズにおいて操作者が処理する対象のデータの管理方法に関する SPoT である。このような SPoT は、対象データをブロックチェーン上に格納する Figure 2-5 のようなアーキテクチャによって排除できる。このアーキテクチャ(I)では、従来のワークフロー管理システムを組織間ワークフローの実行エンジンとして利用し、その実行記録をブロックチェーンに格納する。これにより、定義フェーズにおける定義データや合意記録の改ざん、実行フェーズにおける実行記録の改ざんを防止することができる。しかし、ワークフロー定義に基づいてワークフローの実行を管理する実行エンジンは、参加組織のいずれか、あるいは第三者が管理する集中型システムとして運用されるため、SPoT-3 として挙げた通り、実行エンジンを管理する組織のシステム管理者が SPoT となり、ワークフロー定義を改ざんしたり、ブロックチェーンノードに送信する前に実行記録を改ざんしたりすることができる。

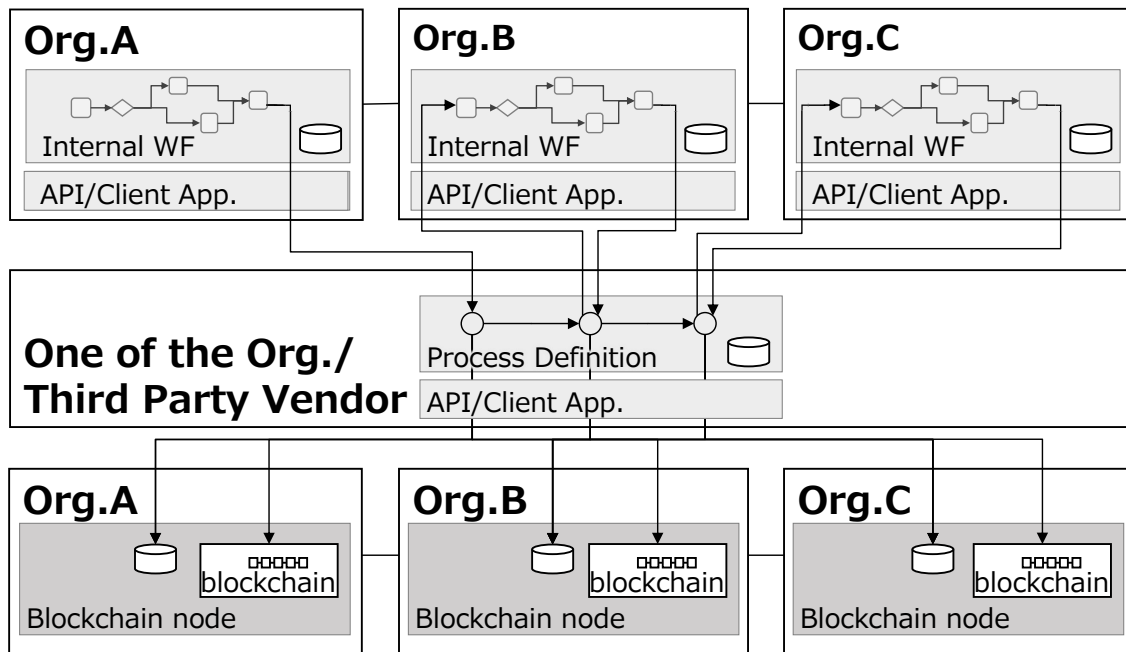


Figure 2-5 System architecture of cross organizational workflow on blockchain (I)

この SPoT-3 を排除するために、アーキテクチャ(II)を提案する。このアーキテクチャでは、ワークフロー定義はブロックチェーン上のスマートコントラクトとして実装され、参加組織の合意なしには変更できない。また、ワークフローのステータスや実行記録の更新もこのスマートコントラクトによって管理されるため、特定の組織のシステム管理者が改ざんすることはできない。このアーキテクチャによれば、ワークフロー実行エンジンが用いるワークフロー定義と、実行フェーズを通じて更新される実行履歴は、共に実行フェーズが終了した後も保持される。このため、監査フェーズにおいても、実行フェーズと同様の方法で格納されたデータを参照することにより、参加組織間で照会した結果が応答されるため、改ざんがあった場合には検知可能である。これにより、監査フェーズにおける SPoT-5 も排除される。このアーキテクチャにおいて、組織内ワークフロー管理システムの組織内ユーザやシステム管理者が、ブロックチェーン上に実装された組織間ワークフロー管理システムに提案する前の実行記録を改ざんすることは可能である。ただし、これは組織内の問題であり、各組織で管理することで対応できる。

上記の通り、アーキテクチャ(II)によれば、組織間ワークフローの定義フェーズにおける SPoT-1、実行フェーズ、および監査フェーズにおける SPoT-3、SPoT-4、SPoT-5 を排除することができる。しかし、定義フェーズにおけるスマートコントラクトが

Table 2-3 に示したワークフロー定義に関わるプロセスを処理するプログラムであるのに対し、実行フェーズと監査フェーズにおけるスマートコントラクトは、ワークフロー実行エンジンとして、Table 2-4 と Table 2-5 に示した実行フェーズと監査フェーズの処理プロセスを処理する別のプログラムである。このため、定義フェーズから実行フェーズに移行する際に、Table 2-3 に示した実装(Deploy)の処理を通じて、ワークフロー定義を実行フェーズのスマートコントラクトにロードする必要がある、SPoT-2 が残る。

そこで、本研究では、アーキテクチャ(II)を用いて、このようなフェーズ間のデータの受け渡しも含めて SPoT を排除する方法として、Figure 2-7 に示すような、定義から実行までのワークフローのライフサイクルを処理する 2 種類のスマートコントラクトをアーキテクチャ(II)の中に併存させ、これらを連動させることによって SPoT-2 を排除する方法を提案する。

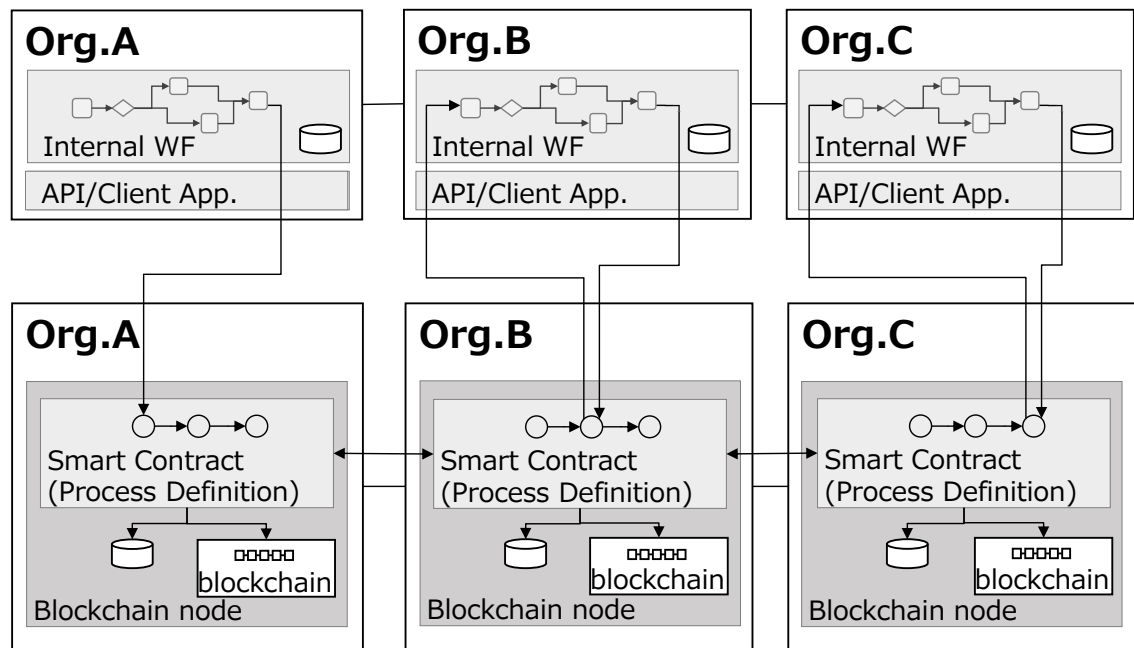


Figure 2-6 System architecture of cross organizational workflow on blockchain
(II)

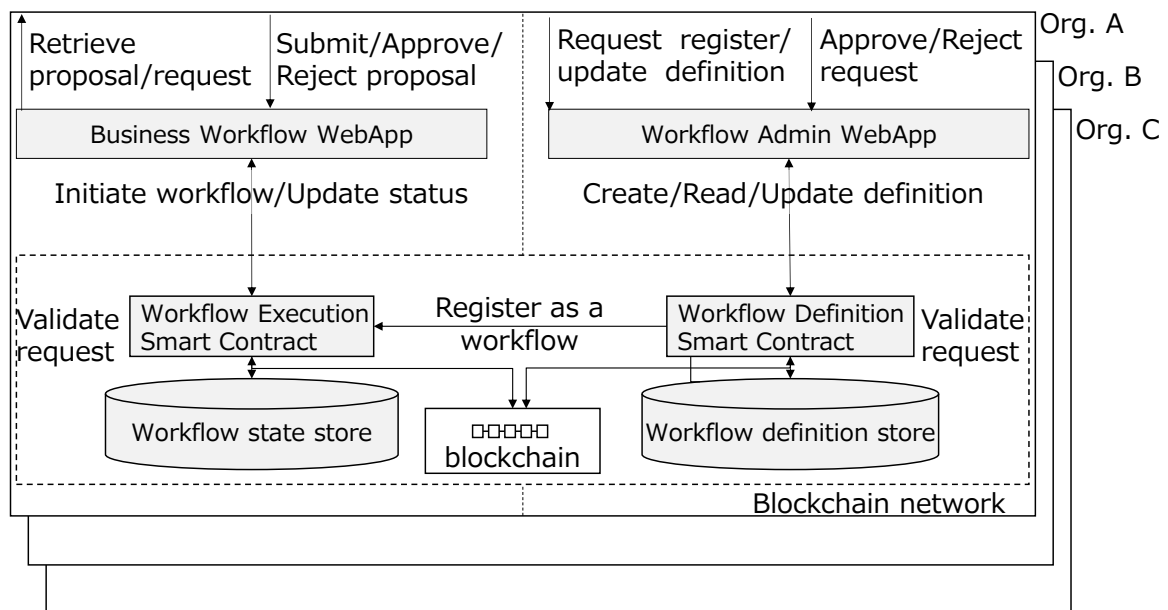


Figure 2-7 System components in blockchain node of system architecture (II)

Figure 2-7 に示すブロックチェーン基盤の構成要素において、1 つ目のスマートコントラクトはワークフロー定義を管理するものであり、もう 1 つはワークフロー処理を管理するものである。ワークフロー定義を管理するスマートコントラクトは、ワークフロー管理者のウェブアプリケーションを通じて、各組織のワークフロー管理者からのリクエストを受け付ける。このスマートコントラクトは、データ項目やステップ、承認者などの要件など、各組織の視点からの要求を受け付け、ワークフロー定義を作成していく。各組織からのリクエストは他のノードに送信され、各ノードのスマートコントラクトがデータ形式やユーザ権限の検証、リクエスト間の競合チェックなどを行う。この検証の結果が一致する場合は、スマートコントラクトはその結果を各ノードのワークフロー定義ストアに書き込む。このように、ワークフロー定義の作成プロセスは、ワークフロー管理者からの要求に基づいてスマートコントラクトによって処理されるため、ワークフロー管理者の一人が全ノードの定義を変更することは不可能である。このような処理により、ワークフロー定義フェーズにおける SPoT-1 は排除され、参加組織が合意したワークフロー定義が改ざんされることなくワークフロー定義ストアに書き込まれる。

実行フェーズでは、ワークフロー実行スマートコントラクトが、ワークフロー定義ストアに書き込まれた合意済み定義に基づいてワークフローを処理する必要がある。しかし、ワークフロー実行スマートコントラクト自身を更新する、スマートコントラクトが

参照するファイルを追加するなど、Figure 2-7 に示した構成要素に変更を加えるような操作はシステム管理者に依存するため、SPoT-2 が問題となる。また、スマートコントラクトがアクセスできるデータストアは、それぞれの処理における更新処理を行うデータストアに限定されるため、ワークフロー実行スマートコントラクトはワークフロー定義スマートコントラクトが管理するワークフロー定義ストアにアクセスすることはできない。そこで、定義フェーズから実行フェーズに移行する際に、ワークフロー実行スマートコントラクトが、ワークフロー定義スマートコントラクトを介してワークフロー定義ストアにアクセスし、新たなワークフロー定義として登録する構成とした。このような構成によれば、特定のシステム管理者に依存せず、スマートコントラクトの処理により、合意済みのワークフロー定義を参加組織間で照合したうえでワークフロー実行エンジンに登録できるため、ワークフローの定義フェーズから実行フェーズに移行する際の SPoT-2 を排除し、合意された定義に基づいたワークフローの実行を保証することが可能となる。

2.3.2 トランザクション処理フロー

本節では、2.3.1 で説明したアーキテクチャによって、SPoT を排除しながら、ワークフローの処理トランザクションがどのように処理され、データがどのように保存されるかを説明する。Figure 2-8 は、ワークフロー処理者が、ワークフローを開始するためにトランザクションを送信する際のトランザクションの流れを示している。まず、参加組織のブロックチェーンノード上のウェブアプリケーションは、処理者が開始できるワークフロー・テンプレートのリストを表示する。処理者はテンプレートの 1 つを選択し、必要な値を設定して提案を送信する。その後、ノード上のワークフロー処理スマートコントラクトが他のノードに提案内容を配布し、各ノードがユーザ権限を含む提案内容を検証し、提案内容を処理してステートデータベースに格納される Read/Write セットを作成する。この Read/Write セットには、提案内容のデータそのものと、処理された結果が含まれる。他のノードから返信された Read/Write セットがこのノードの Read/Write セットと一致する場合は、他のノードに結果の書き込みを要求する。ステートデータベースには、ワークフロー提案内容が、リクエスト ID と、承認者またはレビューアが処理する次のステップとしてのアクションとともに格納される。

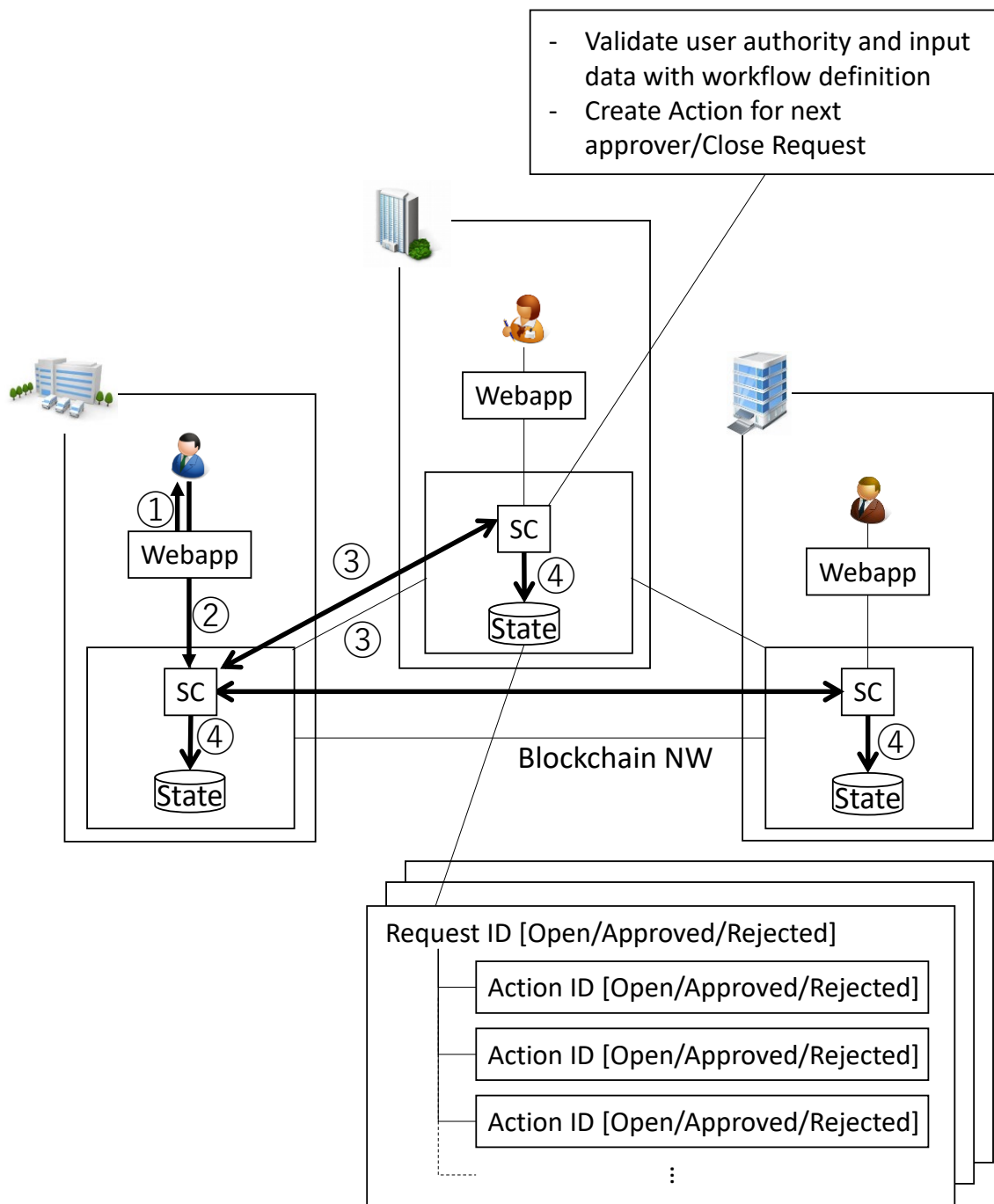


Figure 2-8 Transaction flow of create/update processing

次に、Figure 2-9 を用いて、監査プロセスのトランザクションフローを説明する。ユーザがワークフロー案件の状態を確認するためにレコードの参照を要求する場合や、ワークフロー案件完了後に案件の監査を要求する場合、スマートコントラクトは上述の処

理要求と同様の処理を行う。そして、各ノード上のスマートコントラクトは、検証処理を通じて、各ノード上のステートデータベース間の参照結果が整合していることを検証するため、参照要求の結果をステートデータベースに書き込む。このように、提案システムアーキテクチャのトランザクションフローでは、案件完了後の監査フェーズにおいても、保存されたレコードのノード間での整合性が確保される。

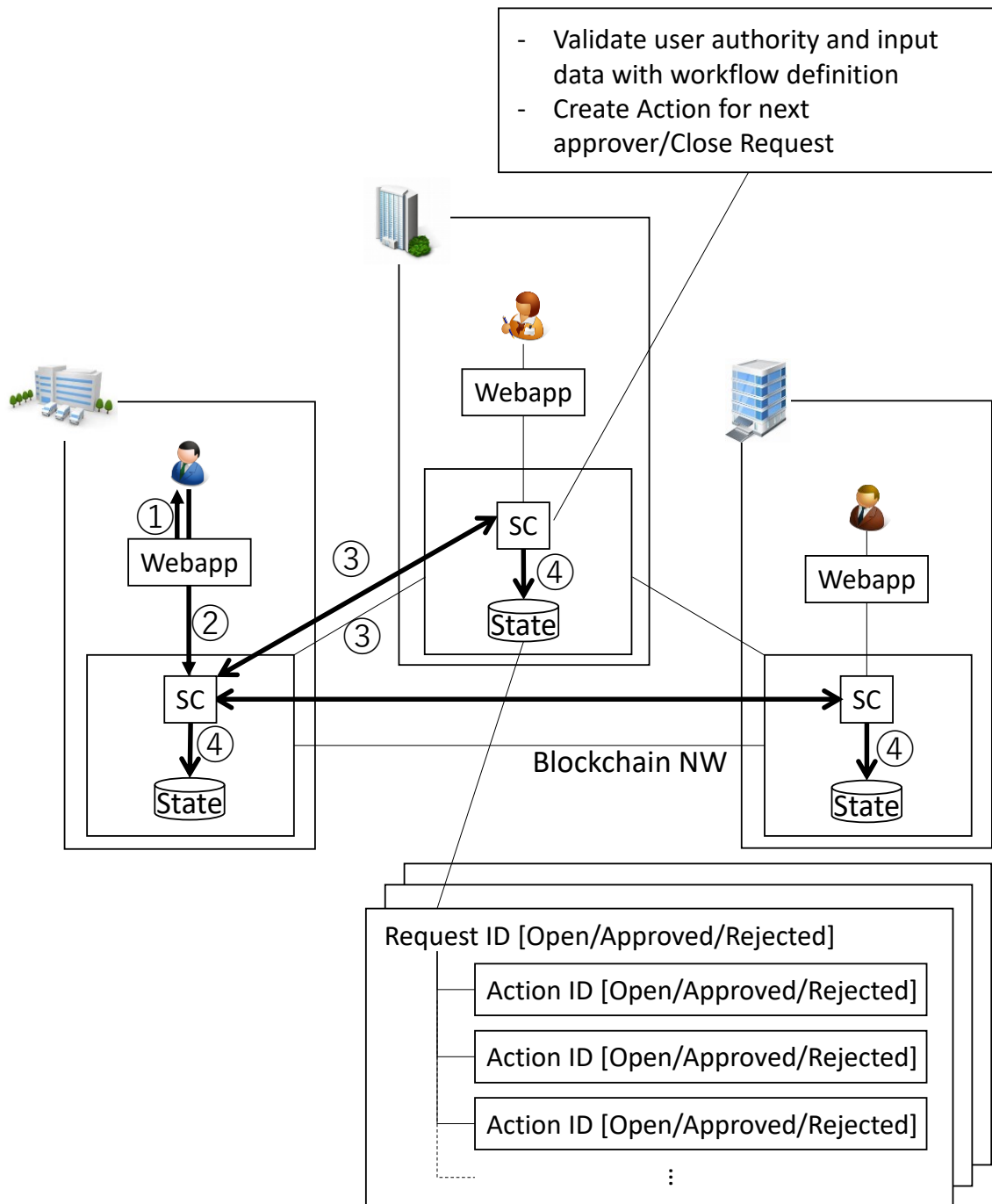


Figure 2-9 Transaction flow of reference processing

2.4 提案システムアーキテクチャの評価

2.4.1 組織間ワークフローのユースケース

文献 [68]に示されているように、ワークフローの構造と制御には多くのパターンが存在するが、すべてのパターンは、トランザクションの提案、承認、却下、およびレビューの組み合わせによって構成される。組織間ワークフローにおいては、組織間でやり取りされるトランザクションは、承認やレビューなどのリクエストとその応答のいずれかである。このため、組織間ワークフローの評価ユースケースとして、ここでは Figure 2-10 に示すようなシンプルな 3 者間ワークフローを想定した。本ワークフローは、複数組織で提供するサービスにおける顧客登録のためのワークフロー処理であり、参加組織の一つから顧客登録リクエストが送信されると、他の二つの組織はこの内容を順次確認した上で承認や却下を行い、最終的に 3 つの組織が全て承認することにより、顧客として登録され、ワークフローが終了する。

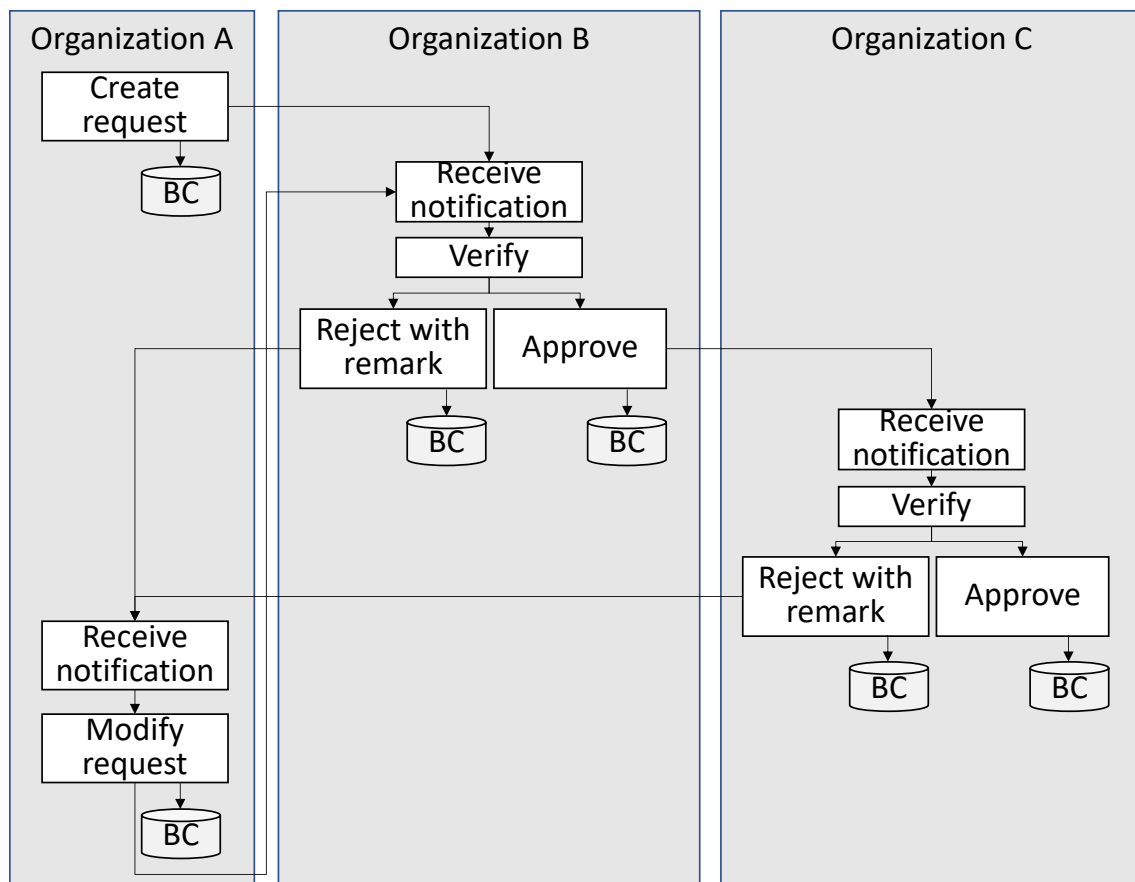


Figure 2-10 Implemented use case of cross organizational workflow

2.4.2 提案システムアーキテクチャのセキュリティ評価

次に、提案システムアーキテクチャをセキュリティの観点から評価する。Figure 2-11 は、2つのアーキテクチャにおける攻撃対象資産を示している。内部ワークフロー管理システムのクライアントアプリケーション(a)は、悪意のあるユーザが組織間ワークフロー管理システムに送信する前のトランザクションデータを改ざんする可能性があるため、攻撃対象となる。アーキテクチャ(I)の組織間ワークフロー管理システムのクライアントアプリケーション(c)も同様に攻撃対象となる。また、悪意のあるユーザがワークフローの定義や結果を改ざんすることを目的とする場合、ワークフロー定義 (b)、ステートデータベース (d)、ブロックチェーン (e) も攻撃対象となる。

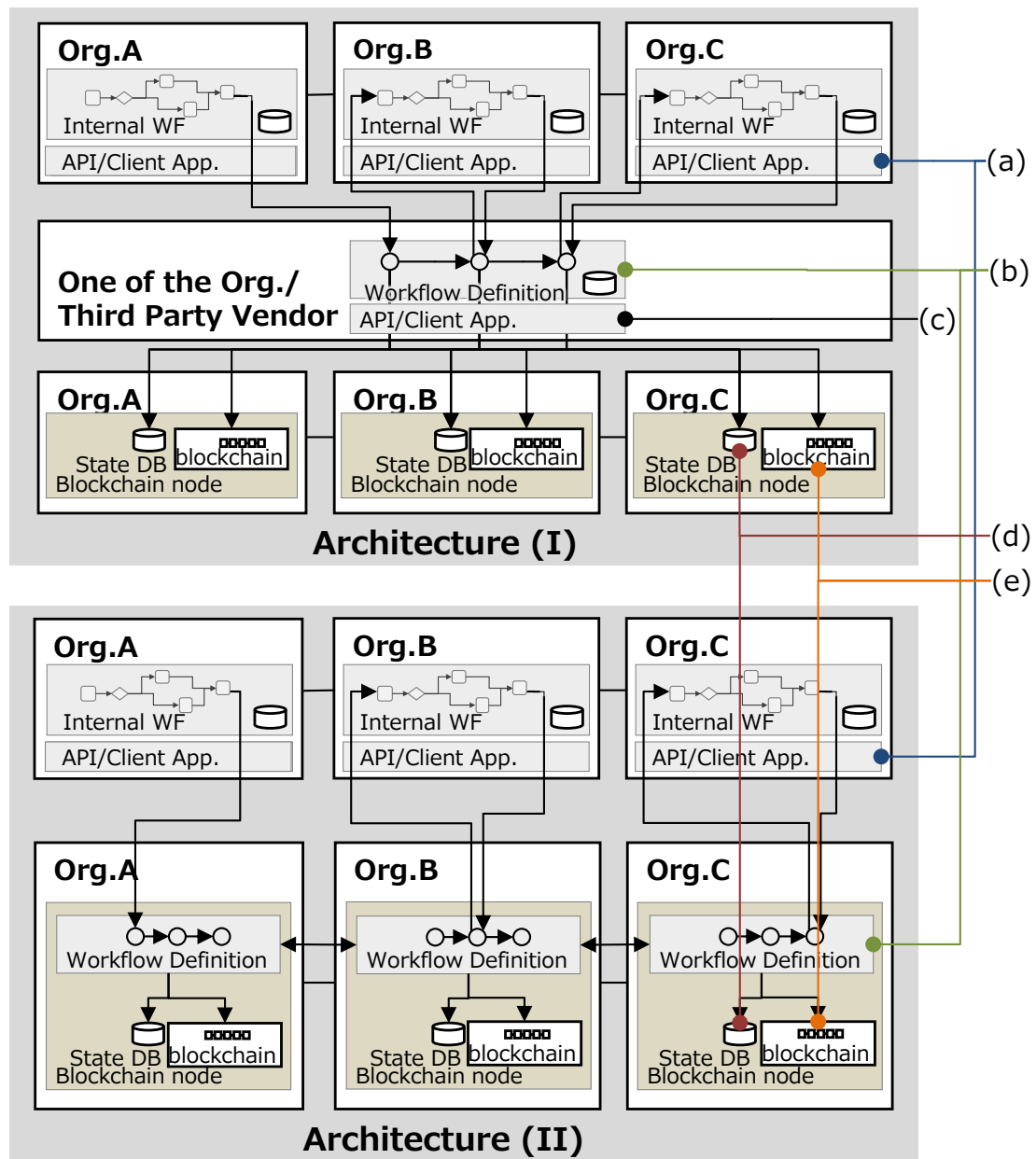


Figure 2-11 Attack target asset from data integrity perspective

Agreement on workflow definition		Open case	Close case
Definition		Execution	Auditing
(a)		(2) Falsification of submitted data by system administrator	
(b)	(1) Falsification of workflow definition by workflow Administrator or system Administrator	(3) Falsification of workflow definition by workflow administrator or system administrator	
(c)		(4) Falsification of submitted data by system administrator	
(d)		(5) Falsification of submitted data by system administrator	(7) Falsification of submitted data by system administrator
(e)		(6) Falsification of submitted data by system administrator	(8) Falsification of submitted data by system administrator

Figure 2-12 Attack scenario through workflow life cycle

Figure 2-12 に、ワークフローのライフサイクルに沿った攻撃対象資産への攻撃シナリオを示す。定義フェーズ、またはワークフロー案件の開始前においては、ワークフロー定義が攻撃される可能性がある。実行フェーズでは、全ての対象資産が攻撃対象となる。監査フェーズでは、実行結果であるステートデータベースとブロックチェーンが攻撃対象となる。

提案アーキテクチャ(II)は、各シナリオにおいて以下のように改ざんリスクを回避する。

シナリオ（1）は、ワークフロー案件を開始する前に、参加組織のワークフロー管理者の一人がワークフロー定義を改ざんすることである。例えば、ワークフロー管理者は、特定の参加組織による承認ステップをスキップするために、定義を改ざんする可能性がある。提案システムアーキテクチャでは、ワークフロー定義プロセス自体がスマートコントラクトとして実装され、参加組織の合意に基づいて作成されるため、定義フェーズにおいて作成中のワークフロー定義を改ざんすることはできない。また、あるノード上のワークフロー定義スマートコントラクトの 1 つが管理者によって改ざんされた場合、ノード間での比較結果が一致しないため、ワークフロー定義の更新トランザクションは

拒否される。さらに、作成されたワークフロー定義は、特定のシステム管理者に依存することなく、ワークフロー定義スマートコントラクトを介してワークフロー実行スマートコントラクトに実装されるため、定義から実行フェーズに移行する際にも改ざんのリスクはない。このように、提案システムアーキテクチャによれば、シナリオ（１）による改ざんリスクは回避される。

シナリオ（２）は、社内ワークフロー管理システムと組織間ワークフロー管理システムのインターフェースにおける提案データの改ざんである。これは社内システムの問題であり、社内のコンプライアンスによって管理されるべきである。

シナリオ（３）は、案件がオープン状態で、参加組織のワークフロー管理者の一人がワークフロー定義を改ざんすることである。悪意のある利用者の動機はシナリオ（１）と同様であり、提案システムアーキテクチャにおいてシナリオ（１）と同様に回避される。

シナリオ（４）は、組織間ワークフロー管理システムとブロックチェーンのデータストア間の送信データの改ざんである。アーキテクチャ(I)では、組織間ワークフロー管理システムは第三者またはいずれかの組織が管理しており、SPoTであるため改ざんリスクがある。しかし、提案システムアーキテクチャでは、これらはブロックチェーンノードで処理されるため、改ざんのリスクはない。

シナリオ（５）は、ワークフロー案件が完了する前に、システム管理者が提案データを改ざんすることである。提案システムアーキテクチャでは、各ノードがスマートコントラクトを実行し、その結果を相互に検証する。このため、あるノードの提案データが改ざんされた場合、他のノードの提案データと異なる結果となり、検証は失敗する。このように、提案システムアーキテクチャではこのリスクを回避することができる。

シナリオ（６）は、実行フェーズにおけるブロックチェーンの送信データの改ざんである。しかし、データ構造の特性上、ブロックチェーンのデータを改ざんすることはほぼ不可能である。

シナリオ（７）は、案件が完了した後に、システム管理者が処理結果を改ざんすることである。承認者全員が案件を承認した後に、特定のノード上でデータを改ざんする可能性がある。提案システムアーキテクチャでは、参照トランザクションでも、各ノードがスマートコントラクトを実行して結果を検証する。したがって、提案システムアーキテクチャでは、この改ざんリスクも回避される。

シナリオ（８）は、監査段階でのブロックチェーン上の処理結果の改ざんである。しかし、シナリオ（６）と同様、改ざんはほとんど不可能である。

監査フェーズにおけるセキュリティの観点から、２つのシステムアーキテクチャを比較した結果を Table 2-6 に示す。両アーキテクチャとも、ブロックチェーンにデータが保存されるため、監査証跡の不変性という点では安全である。しかし、(I)のシステムアーキテクチャは、ワークフロー定義が組織や第三者によって管理され、悪意のあるユーザによって改ざんされる可能性があるため、ワークフロー定義の遵守という観点では安全ではない。これに対し、提案システムアーキテクチャ(II)では、ワークフロー定義がスマートコントラクトとして各参加組織で管理され、参加組織間の合意がなければ変更できないため、この点に関して安全である。また、システムアーキテクチャ(I)は、組織内ワークフロー管理システム(a)および組織間ワークフロー管理システム(c)のクライアントアプリケーションの脆弱性により、組織内ワークフローと組織間ワークフロー間の記録の整合性という点でも安全とは言えない。特に、攻撃対象（c）は各組織の管理下にない。一方、攻撃対象(a)のリスクは限定的であり、各組織内でコントロール可能である。このため、攻撃対象(c)が存在せず、組織内ワークフローの処理結果を SPoT のない組織間ワークフロー管理システムで直接処理する提案システムアーキテクチャ(II)は、組織内ワークフロー管理システムと組織間ワークフロー管理システムの間のデータの一貫性という意味で安全であるといえる。

Table 2-6 Comparison of security between two system architectures

	Architecture (I)	Architecture (II)
Consistency of audit record between organizations	Secure	Secure
Compliance with workflow definition	Not Secure	Secure
Consistency of execution record between internal and cross-organizational workflow management systems	Not Secure	Secured by internal audit

このように、提案システムアーキテクチャは、ワークフロー定義プロセスと実行プロセスを、それぞれスマートコントラクトとして実装するだけでなく、実行プロセスのスマ

ートコントラクトを定義プロセスのスマートコントラクトによって自動的に各ノードに配置することにより、組織間ワークフロー管理システムにおける SPoT をライフサイクル全体に渡って排除する。これにより、組織間ワークフローに関する改ざんリスクを全て回避し、全ての参加組織に対して信頼性の高い組織間ワークフロー管理システムを提供することができる。

2.5 結言

本章では、組織間ワークフロー管理システムを実現する信頼性の高いシステムアーキテクチャを提案した。提案システムアーキテクチャでは、参加組織それぞれが保有する組織内ワークフロー管理システムの処理結果をもとに、組織間ワークフロー処理システムへの処理トランザクションを発行する。組織間ワークフロー処理システムには、信頼できる組織間の安全なデータストアとしてブロックチェーンを利用し、安全なワークフロー処理エンジンとしてブロックチェーン上のスマートコントラクトを利用する。さらに、定義フェーズと実行フェーズを移行する際の処理についても、システム管理者に依存せず、スマートコントラクトを連動させることで SPoT を排除する。ワークフローのライフサイクルに沿った攻撃シナリオに基づく改ざんリスクの評価の結果、提案システムアーキテクチャによれば、組織間ワークフロー管理システムにおける SPoT を排除することで、すべての改ざんリスクを回避できることを確認した。

第3章

ブロックチェーンを活用した組織間ワークフロー管理システムの実行エンジン

3.1 緒言

2章では、ワークフローの定義処理と、実行・監査処理をそれぞれブロックチェーン上のスマートコントラクトとして実装し、これらを連動することにより、組織間ワークフロー管理における SPoT を排除するシステムアーキテクチャを提案し、攻撃シナリオに基づくセキュリティ評価を行った。しかし、ワークフロー実行エンジンであるスマートコントラクトの実装にあたっては、ワークフロー処理の特性を考慮した処理方式やデータ構造を用い、ワークフロー処理者が正しい参照データに基づいて処理を行うことを保証するとともに、処理性能を確保する必要がある。

本章では、上記アーキテクチャで動作するワークフロー実行スマートコントラクトの実装方式として、ワークフローのステップ間でのデータの一貫性に着目し、これを確保することにより情報の信頼性を保証する方式を提案する。提案方式では、データの一貫性を保証するために冗長性の高いデータモデルを用いるため、格納データサイズが増大するとともに、トランザクション処理性能が低下する可能性がある。このため、具体的なワークフロー例を実装し、データサイズと性能の観点で評価する。

以下、3.2節では、ブロックチェーンを活用した組織間ワークフロー管理システムにおけるスマートコントラクトの実装上の課題を整理し、3.3節でこれを解決する逐次データ保証型ワークフロー実行エンジンの実装方式を提案する。その後、3.4節で実装評価の結果を述べる。

3.2 組織間ワークフロー管理システムの実装課題

2章で提案したシステムアーキテクチャでは、予め組織間で合意し、各ノードに配置されたワークフロー定義に基づいて、ワークフロー実行スマートコントラクトがワーク

フロー処理を実行する。このシステムアーキテクチャにおけるワークフロー実行処理の概要を Figure 3-1 に示す。

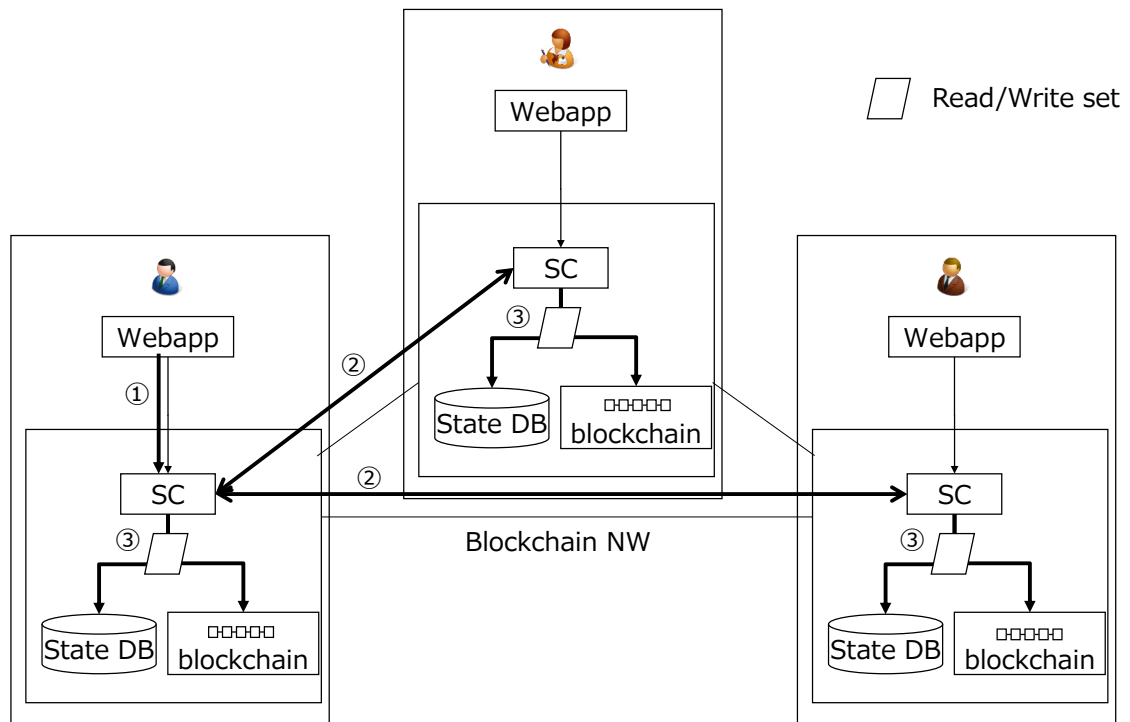


Figure 3-1 Transaction flow of workflow execution processing

まず①において、ワークフロー処理者は、承認・却下などのワークフロー処理リクエストを Web アプリケーション経由でワークフロー実行スマートコントラクトに送信する。ワークフロー実行スマートコントラクトは、②において、処理リクエストを他のノードに配信する。各ノードのワークフロー実行スマートコントラクトは、当該ランザクションを実行する前のワークフロー処理ステータスなど、更新前の State DB 参照データと、実行後の処理ステータスや次の処理ステップなど、State DB の更新データをセットにした Read/Write セットを作成し、送信元のノードに返信する。一定数の返信を受信した送信元ノードのワークフロー実行スマートコントラクトは、各ノードからの Read/Write セットを比較し、それらが一致する場合に、③において Read/Write セットを各ノードに配信し、各ノードにてブロックチェーンファイルに追加するとともに、State DB を更新する。

特定の組織のシステム管理者によって、ワークフロー実行スマートコントラクト、あ

るいは処理ステータスなどの格納データが改ざんされていた場合、他のノードと Read/Write セットが一致しないため、処理が完了しない。これにより、ワークフロー定義フェーズで合意したプロセス通りにワークフロー処理が実行され、その処理結果が各ノード間で一致していることを、全ての参加組織が信頼することができる。

このように、2 章で提案したシステムアーキテクチャによれば、単一トランザクションの実行においては、当該トランザクションで参照・更新するレコードを対象とした Read/Write セットを比較することによって、データの改ざんを検知することができる。しかし、ワークフロー処理は複数ステップから構成される処理であるため、ブロックチェーン上では複数のトランザクションにまたがって処理が行われる。このため、単一トランザクション単位の Read/Write セットの比較では、ワークフローの処理プロセス全体に渡るデータの一貫性を保証することはできない。例えば、あるステップにおける処理で、他のステップにおいて更新されたデータや承認履歴を参照したうえで承認の判断をする場合、正しいデータやその履歴を参照したことを保証するために、そのようなデータも Read/Write セットに含めて比較する必要がある。このように、ワークフロー処理プロセスの信頼性を担保するためには、ノード間で照合が必要なデータの範囲を定義し、これを Read/Write セットとして生成し、照合するためのデータ構造、及びスマートコントラクトの処理設計が課題となる。この際、各トランザクションで比較照合する Read/Write セットのデータ量は増大するため、データ量や処理性能についても考慮する必要がある。

以下、3.3 節では上記の課題を解決するワークフロー実行エンジンの実装方式について述べる。

3.3 逐次データ保証型ワークフロー実行エンジン

3.3.1 逐次データ保証型ワークフロー実行エンジンの考え方

トランザクション処理時にノード間で照合する Read/Write セットは、当該トランザクション実行時に更新する State DB のレコードの更新前後のデータである。State DB のデータ構造としては、起案、承認、却下などの処理毎にレコードを分け、これらのレコードをポインタでつなぐ Figure 3-2 に示す構成案(a)と、ワークフロー案件に対する処理全てを 1 つのレコード内にまとめる Figure 3-3 に示す構成案(b)が考えられる。

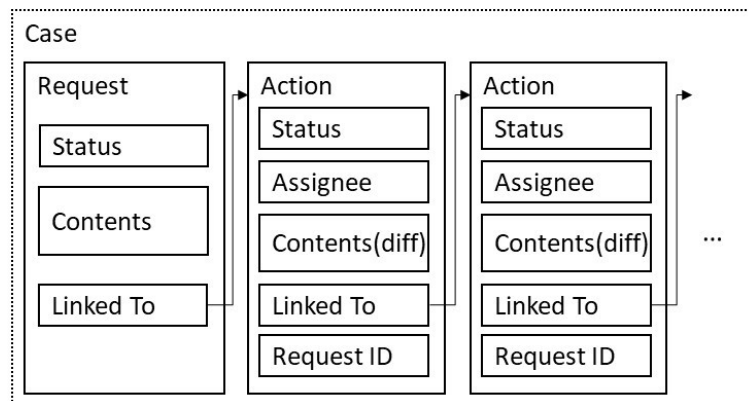


Figure 3-2 Data Structure of State DB (a)

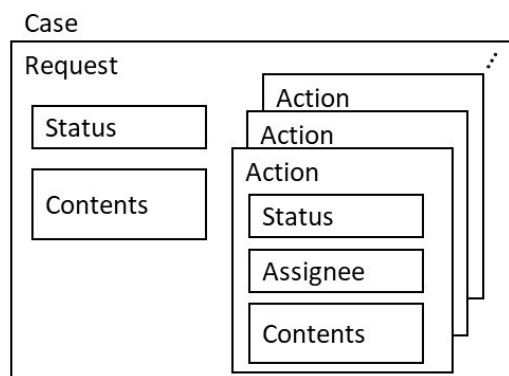


Figure 3-3 Data Structure of State DB (b)

構成案(a)では、ワークフロー案件(Case)の起案(Request)と、その後の承認、却下などの Action のたびに 1 レコードが生成される。各レコードには、案件の ID(Request ID) が付与されるとともに、レコード間は後続のレコードを示す Linked To の情報で関連付けられる。また、起案内容(Contents)については、Request レコードに提案時の内容が記録され、その後の Action では、更新がある場合にのみ、その差分が記録される。これにより、最小限のデータ量の追加でトランザクションを処理していくことが可能となる。

一方、構成案(b)では、各案件のデータは 1 レコード内に全て格納される。ワークフローの起案時にレコードが生成されると、その後の Action のたびに、当該レコード内に Action データが追記されていく。各 Action には、当該 Action の時点での Contents が

逐次格納される。これにより、対象ワークフロー案件のデータを、Request ID を指定することにより一括で取得することができる。

それぞれのデータ構造案における Read/Write セットの範囲を、Figure 3-4、及び Figure 3-5 に示す。構成案(a)の場合、トランザクション処理時に生成される Read/Write セットは、Figure 3-4 に示す通り、当該 Action の更新前後のレコードと、生成される次の Action から構成される。当該ワークフロー案件に関するレコードの内、上記以外のレコードについては、Read/Write セットに含まれず、ノード間で照合処理が行われない。このため、例えばある組織が承認リクエストを処理する際に、この組織のノード上で、起案時の Request ID のコンテンツがシステム管理者によって改ざんされていた場合、改ざんされた起案内容に基づいて承認処理がなされ、承認後にシステム管理者がこれを隠ぺいするためにコンテンツを元に戻すというような不正が行われる可能性がある。このように、構成案(a)では、ワークフロー実行中のデータの信頼性に関して課題がある。

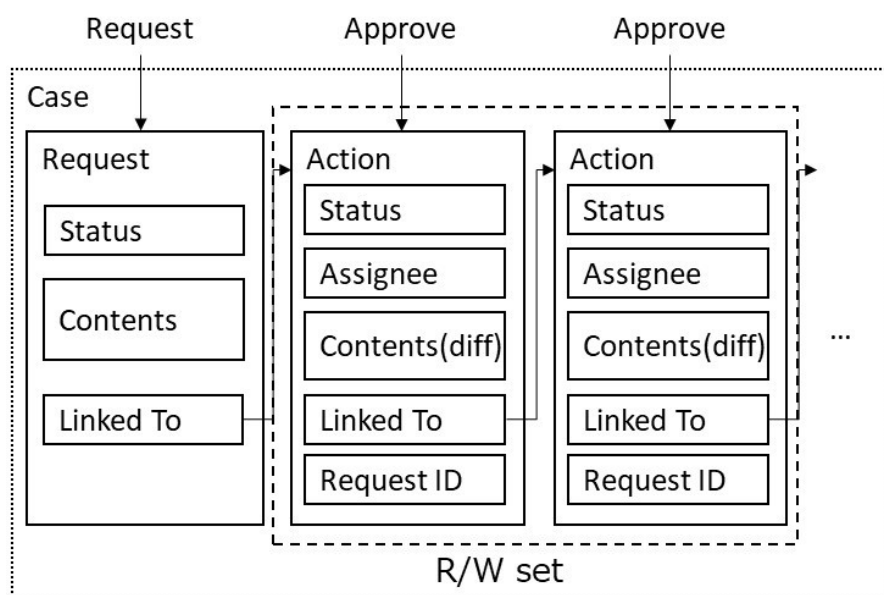


Figure 3-4 Read/Write set in data structure (a)

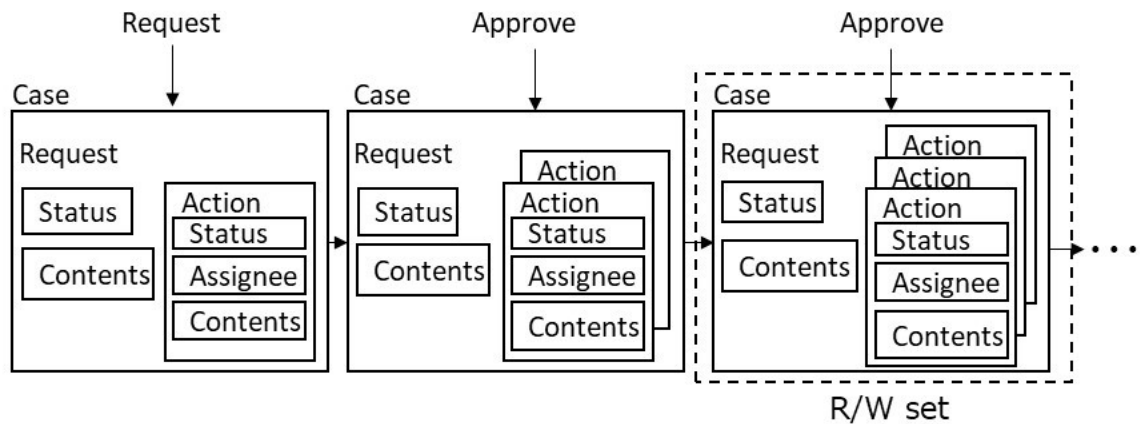


Figure 3-5 Read/Write set in data structure (b)

一方、構成案(b)の場合、トランザクション処理時に生成される Read/Write セットは、Figure 3-5 に示す通り、当該時点での最新の Request レコードの更新前後のデータとなる。Request レコード内には、当該ワークフロー案件のそれまでの処理履歴が全て含まれている。このため、ワークフローの各ステップを処理するたびに、当該ワークフロー案件の過去の処理履歴も含めて、全てのデータをノード間で照合しながら処理することになる。これにより、ワークフローの各処理者が Request や Action の処理をする際に、そのステップでの処理内容だけでなく、どのような Contents に対して処理を行ったのか、またどのような処理履歴を参照した上で処理を行ったかも含めて、全ての参加組織間で逐次データの整合性を保証しながら処理をすることが可能となる。このように、構成案(b)によれば、構成案(a)で課題となっていた、ワークフロー実行中にデータが改ざんされる可能性を排除し、データ改ざん耐性の高いワークフロー実行エンジンを実現することができる。しかし、構成案(b)では、ワークフローのステップが進むたびに Request レコードのサイズが増大し、それに伴って Read/Write セットのサイズも増大する。このため、処理性能への影響やストレージ容量の増大が懸念事項となる。これについては 3.4 節で評価する。

次項では、構成案(b)に基づいた逐次データ保証型(Sequential Data Verification, 以下 SDV)ワークフロー実行エンジンの処理の詳細について述べる。

3.3.2 SDV ワークフロー実行エンジンの詳細

構成案(b)を用いた SDV ワークフロー実行エンジンの処理フローを Figure 3-6 に示す。ここでは、Figure 3-6 の左上にあるワークフロー案件の Case a について、Action-1 として定義された承認処理を割り当てられたユーザが、ワークフロー実行エンジンを通じて承認処理を行うケースを例に、構成案(b)を用いることでどのようにデータの整合性が保証されるかを示す。

まず、ワークフロー実行エンジンは、処理(1)においてユーザ権限を確認したのち、処理(2)において、その時点で当該ユーザに割り当てられているワークフロー案件の一覧を取得する。Action データの Assignee に当該ユーザが指定されており、かつ当該 Action の Status が Open の Action を含む Request レコードを指定して取得することで、当該ユーザに割り当てられている処理待ちの全案件のデータを取得することができる。ユーザは処理対象案件を選択し、承認／却下の処理内容を Status に指定し、必要に応じて Contents を更新した Action レコードを、処理(3)においてワークフロー実行エンジンに送信する。

これを受信したワークフロー実行エンジンは、処理(4)において、State DB に格納した当該案件のワークフロー定義に基づいて、送信された Action レコードのコンテンツを検証する。問題がない場合は、処理(5)において、ワークフロー定義に基づいて次の処理を Action として生成し、Assignee などの情報をセットした上で、処理(6)においてこれらの情報をまとめて Read/Write セットを生成する。Figure 3-6 では、このユーザが、当該ワークフロー案件の起案に対する次の処理 (Action-1) を実行し、次の承認者の処理 (Action-2) を生成する例を示している。

この際、構成案(b)のデータ構造を採用することにより、Read/Write セットには、このユーザによる承認／却下等の意思決定の結果だけでなく、意思決定に用いた参照データや処理プロセスの情報が含まれる。Read/Write セットの内、Read 部分は、Request レコードの更新前のデータであり、この案件の過去の処理履歴として、元々の起案データと、このユーザが処理する前の Action のデータが含まれる。これを処理(7)において、他の参加組織のノードとの間で照合を行うことにより、当該ユーザが Action-1 を処理するにあたり参照した当該案件の起案コンテンツや過去の処理履歴が、ノード間で一致していることを保証できる。

一方、Read/Write セットの内、Write 部分には、当該ユーザによる処理内容である Action-1 の更新部分と、次の処理として生成された Action-2 のデータが含まれる。これを他のノードと照合することにより、Action-1 の処理内容と、生成された Action-2 の情報から、各ノードのワークフロー定義にも改ざんがないことを確認できる。

このように、構成案(b)を用いた SDV ワークフロー実行エンジンによれば、ワークフロー処理者の意思決定の根拠も含めて、Read/Write セットとして他ノードと照合されるため、参照情報の改ざんによる不正も含めて検知可能となり、プロセスが正しく行われていることを保証できる。さらに、Read/Write セットに含まれるこのような情報は、後の監査フェーズでも必要な情報であり、情報収集工数の削減と監査漏れの防止にも役立つ。

処理(7)の照合処理の結果、他ノードと Read/Write セットが一致しない場合、処理(8)においてユーザに Reject を送信する。システム管理者は、不一致部分を元に、どの部分に改ざんがあったかを確認することができる。処理(7)の照合処理が一致した場合には、処理(8)において Write データを他ノードに配信するとともに、処理(9)において、自ノードの State DB と blockchain のデータを更新し、処理(10)において結果をユーザに送信する。

このように、ワークフローのステップ毎に、その時点における当該案件のデータ全体、及びワークフロー定義に基づく次のステップの Action を含む更新データをノード間で逐次確認することにより、処理結果とプロセスの両方のデータの信頼性を、ワークフローの実行時に逐次保証するワークフロー実行エンジンを実現できる。

3.4 SDV ワークフロー実行エンジンの評価

SDV 方式では、ワークフロー実行処理の各ステップにおいて、当該案件に関わる全てのデータをノード間で照合し、ブロックチェーンに追記していくため、ワークフローのステップ数に応じた保存データ量の増大とトランザクション処理性能の低下が問題となる可能性がある。本節では、提案方式によるワークフロー実行エンジンの実装を通じて、データ量と処理時間を評価し、提案方式の実用性を検証する。

評価には、Figure 2-10 のユースケースを用いる。Figure 2-10 において、Organization B、あるいは Organization C が提案内容を却下することで、ワークフロー処理は

Organization A に差し戻され、再提案、承認を繰り返す場合が想定できる。このように、本ユースケースを用いた評価により、ステップ数に応じたトランザクション処理性能と保存データ量の評価を行うことができる。

3.4.1 データサイズの評価

3.3.1 節で述べた通り、State DB のデータ構造には、Figure 3-2 に示した構成案(a)と、Figure 3-3 に示した構成案(b)がある。構成案(a)は、ワークフローの各ステップにおいて生成される差分データのみを Read/Write セットとして生成するため、State DB、及び blockchain のデータ量を最小限にできる。一方、提案方式で用いる構成案(b)によれば、ワークフロー実行時のデータの信頼性は担保されるが、データ量が増大する問題がある。本節では、Figure 2-10 に示す 3 つの組織から承認を得る顧客情報登録のワークフローを前提として、構成案(a)と構成案(b)それぞれについて、システム全体で必要とされるデータ容量を評価する。

Figure 2-10 に示したワークフロー処理で使用する各データのサイズを、Table 3-1 のように設定する。

Table 3-1 Record size (bytes)

	Contents	Header	Total
Request	1024	32	1056
Action	1024	48	1072

Contents は、本ワークフローで登録する顧客情報、及びワークフロー処理時に追加される処理者のコメントなどのデータである。Header は、それぞれ Request、Action レコードの Contents 以外のデータであり、Status や LinkTo の情報の他、更新日時や登録者などの情報を含む。Contents のデータサイズは、ワークフローのユースケースによって変動することが想定されるが、添付ファイルを含まないテキスト情報からなるワークフロー処理の場合は、概ねこの程度のサイズに収まると考えられる。また、構成案(b)については、Request レコードに Action レコードが含まれるが、以下では Request レコードは Action データ以外の部分を表すこととする。

- ・ R : Request レコード全体のサイズ(bytes)
- ・ A : Action レコード全体のサイズ(bytes)

- A_H : Action レコードのヘッダサイズ(bytes)
- A_C : Action レコードのコンテンツサイズ(bytes)

とすると、構成案(a)におけるワークフローのステップ n での Read/Write セットのサイズ RW_{an} は、まずステップ 1 の Request トランザクションで、

$$RW_{a1} = R + A_H \quad (1)$$

となる。ステップ 2 以降は、更新前後の Action レコードのサイズと次のステップの Action レコードの合計となる。更新後の Action レコードのサイズは、各ステップにおけるコンテンツ更新の有無によって変動するが、承認のみでコンテンツの更新を含まない場合、ヘッダ内の Type を承認済みに変更するのみであり、Read/Write セットのサイズは、

$$RW_{an} = A_H \times 3 \quad (2)$$

で一定となる。

一方、構成案(b)の Read/Write セットのサイズ RW_{bn} は、ステップ 1 の Request トランザクションは構成案(a)の場合と同様、

$$RW_{b1} = R + A_H \quad (3)$$

となるが、ステップ 2 以降は、

$$RW_{bn} = (R + A \times (n - 2) + A_H) + (R + A \times (n - 1) + A_H) \quad (4)$$

となる。

次に、State DB のサイズについては、構成案(a)の場合、ステップごとに Action レコードが 1 件ずつ追加されていくため、ステップ n における State DB のサイズ S_{an} は、

$$S_{an} = RW_{a1} + (A_H \times (n - 1)) \quad (5)$$

となる。一方、構成案(b)では、Contents を含む Action レコードが Request レコード内に追加・更新されていくため、ステップ n における State DB のサイズ S_{bn} は、

$$S_{bn} = RW_{b1} + (A \times (n - 1)) \quad (6)$$

となり、各トランザクションにおいて、コンテンツサイズ分、構成案(a)よりサイズが大きくなる。

最後に、Blockchain のデータサイズは、トランザクションごとに Read/Write セットが追加されていくため、R/W セットの累積のデータ量となり、構成案(a)の場合のステップ n における Blockchain データサイズ B_{an} 、構成案(b)の場合のステップ n における Blockchain データサイズ B_{bn} は、それぞれ、

$$B_{an} = \sum_{i=1}^n RW_{ai} \quad (7)$$

$$B_{bn} = \sum_{i=1}^n RW_{bi} \quad (8)$$

となる。

これらの数式と、Table 3-1 に示した顧客情報登録のユースケースにおけるレコードサイズに基づいて、ワークフローの各ステップにおけるそれぞれのデータサイズを計算した結果を Table 3-2 に示す。一般に、組織間ワークフローのステップ数は、参加組織数と同程度となることが想定される。このため、Figure 2-10 で示した 3 ステップ (Action-2 まで) のワークフローにおけるデータ量に加え、大規模な組織間ワークフローを想定し、ステップ数が 10 の場合の 10 ステップ目 (Action-9) のデータ量を計算し、ワークフローのステップ数に応じたデータ量を評価した。

Table 3-2 Data size (bytes)

	R/W set		State DB		blockchain		Storage	
	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
Request (Step 1)	1104	1104	1104	1104	1104	1104	2208	2208
Action-1 (Step 2)	144	3280	1152	2176	1248	4384	2400	6560
Action-2 (Step 3)	144	5424	1200	3248	1392	9808	2592	13056
Action-9 (Step 10)	144	20432	1536	10752	2400	107808	3936	118560

構成案(a)のデータ構造では、R/W セットは対象ステップの更新前後のデータのみであるため、ワークフローのステップによらず一定である。このため、R/W セットの累積となる Blockchain データのサイズも線形で増加するのみであり、State DB と Blockchain を合わせたストレージ容量は 10 ステップ目でも約 4kB/件である。月 1000 件のワークフローを処理した場合でも、 $4\text{kB} \times 1000 \times 12 = 48\text{MB/年}$ である。

一方、構成案(b)では、各ステップにて処理されるコンテンツも含めた Action レコードが、Request レコード内に追加・更新されていくため、この更新前後のデータを合わせた R/W セットのサイズがステップごとに増加していく。さらに、R/W セットの累積である Blockchain データもこの累積で増加し、ステップ数のおよそ二乗で増加する。

この結果、Blockchain のデータサイズは、3 ステップ目で 9,908bytes、Blockchain と State DB を合わせた全体のストレージ容量は、13,056bytes/件となり、10 ステップ目で約 119kB/件となる。しかし、月 1000 件のワークフローを処理した場合でも、 $119\text{kB} \times 1000 \times 12 = 1,428\text{MB/年}$ であり、企業システムのデータ容量としては問題ないと言える。ただし、Contents のデータ容量によってはこれがさらに増大する可能性はある。しかし、ワークフローに添付される PDF ファイルや画像ファイルのような大容量データは、IPFS 等の外部ストレージに格納し、ブロックチェーンシステムにはこのファイルへのリンクと改ざん防止のためのハッシュ値のみを格納する方法が一般的である [44]。このような方法によれば、改ざんが困難なブロックチェーン上のハッシュ値を用いて、外部ストレージに格納されたファイルに改ざんがないことを検証することが可能となるとともに、ブロックチェーン上のストレージ容量を抑制することができる。

3.4.2 性能評価

3.4.1 節で評価した通り、SDV 方式では、ワークフローのステップ数に応じて Read/Write セットのデータサイズが増大するため、Figure 3-6 に示した処理フローのうち、特に処理(7)における他ノードとの比較や処理(9)におけるデータの書き込み処理において処理時間が増大し、トランザクション性能が低下する可能性がある。このため、ワークフロー実行、及び監査の処理性能を評価するため、提案方式によるワークフロー実行エンジンを、コンソーシアム型ブロックチェーン基盤である Hyperledger Fabric を用いて実装し、書き込みデータサイズと処理性能の関係を評価した。実測環境の構成を Table 3-3 に示す。クラウド上の 1 インスタンス上に、3 組織のノードからなるブロックチェーンネットワークを構成し、合意形成方式(Endorsement Policy)には、3 ノードの内、2 ノードからの Endorsement を基に合意とみなす方式を採用した。

Figure 3-6 の処理フローにおいて、処理(3)でクライアントアプリケーションがリクエストを送信してから、処理(10)で結果を受信するまでの応答時間を、State DB への書き込みデータ量を変動させて計測した結果を Figure 3-7 に示す。

Table 3-3 Specifications of evaluation environment for SDV

Item		Spec
Hardware	CPU	Xeon 8175M (2.5GHz) x 2 cores
	Memory	8GB
Software	OS	Ubuntu 18.04.5 LTS
	Blockchain	Hyperledger Fabric 2.2.1
	Database	CouchDB 3.1.0
	Language	Node.js (client/chaincode)
Blockchain Network Configuration		3 Organizations on a single instance
Endorsement policy		2 out of 3

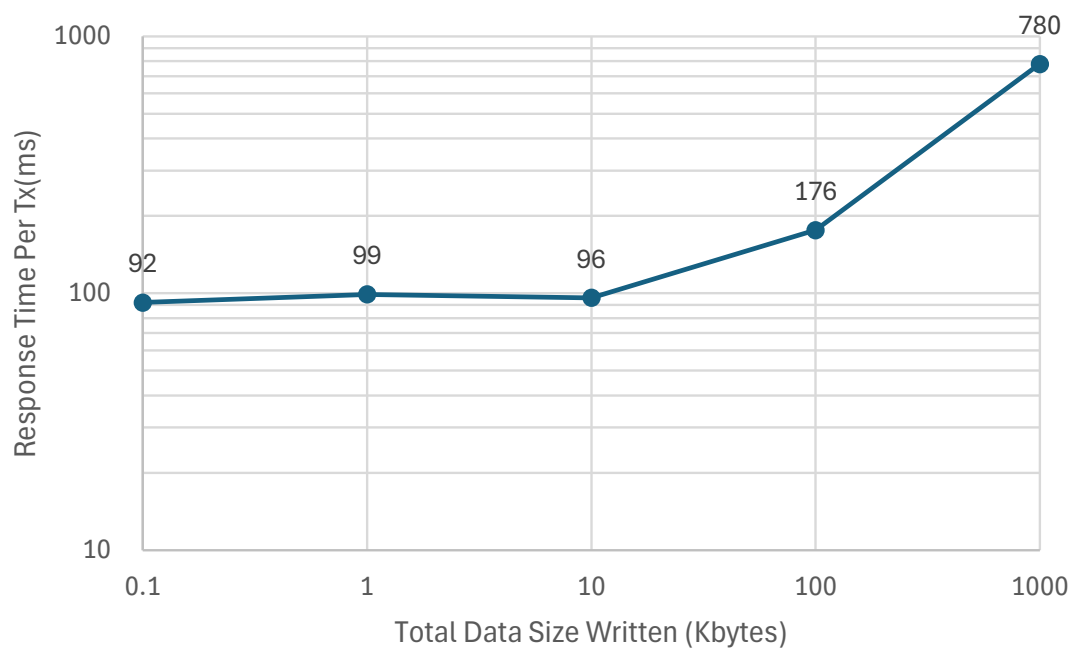


Figure 3-7 Response time by data size

提案方式における書き込みデータは、当該ステップにおいて更新される Request レコードである。構成案(b)では、State DB 上で 1 件の Request データが更新されていくため、このデータサイズは、State DB のサイズと同一であり、Table 3-2 に示す通り、3 ステップ目で、3,248bytes、10 ステップ目で 10,752bytes である。Figure 3-7 に示す通り、書き込みデータサイズが 10kB までは、応答時間はほぼ 100ms で変動がない。このため、参加組織数が増大し、10 ステップ目に到達するような大規模な組織間ワー

クフローにおいても、実用上問題ない処理時間で処理できると言える。一方、書き込みデータがこれ以上のサイズになる可能性として、Request データのうちの Contents のサイズが増大するケースが考えられる。しかし、この場合は、3.4.1 項で述べた通り、Contents に含まれる PDF ファイルなどの大容量ファイルを外部ストレージに格納すれば、ブロックチェーンシステムへの書き込みデータ量を 1MB 程度に抑えることは可能である。したがって、提案方式による処理性能は実用上問題ないと言える。

次に、監査時の処理性能について評価する。監査時には、各ワークフロー案件について、全ての処理データを取得する必要がある。構成案(b)のデータ構造によれば、各ワークフロー案件のデータは Request レコードに全て格納されているため、1 トランザクションで 1 案件のデータを取得できる。このため、ワークフロー実行時のトランザクションと同様の処理時間で監査向けデータの取得を行える。

提案方式によって、月 1000 件のワークフロー案件を処理した場合、年間 12,000 件の取得に要する時間は、 $100\text{ms} * 12,000 = 1,200 \text{ sec (20min)}$ となる。従来、組織間ワークフローに対する監査業務では、対象企業でのデータ取得に加え、相手先企業のデータも個別に取得し、これらを突き合わせる作業が必要であった。しかし、提案方式によれば、組織間で一致が保証されたデータを 20 分で取得可能であり、監査業務を大幅に効率化できると考えられる。

3.5 考察

本章では、業務プロセスの単位であるワークフロー案件の全てのステップのデータが 1 レコードに含まれるデータ構造とし、これを各トランザクション実行時に逐次照合する処理方式として、SDV ワークフロー実行エンジンを提案した。提案方式によれば、業務プロセス単位でデータの一貫性を保証することができる。

一方で、このような設計とした結果、業務プロセスに含まれるステップ数に比例して、ブロックチェーンへの書き込みデータサイズが増大し、処理性能への影響が懸念されるが、評価ユースケースで 10 ステップ目に相当する 10kB 程度の書き込みまでであれば、性能への影響はないことが分かった。一般に、組織間ワークフローでは、参加組織数と同程度のステップ数となることが想定されるため、10 社程度が参加する大規模な組織間ワークフローでも、実用上問題ないと言える。

また、データ容量の増大について、3.4.1 で述べた通り、各ノードでのデータサイズのうち、**blockchain** データの占める割合が大きく、このサイズはステップ数のおよそ二乗に比例する。さらに、**SDV** ワークフロー実行エンジンでは、参加する組織全てが同じデータを保有するため、参加組織数が増大した場合には、組織間ワークフローシステム全体でのデータの冗長性が増大し、システム全体での格納データサイズは、参加組織数が n の場合、各ノードのデータ容量の n 倍になる。上記の通り、ワークフローのステップ数がおよそ参加組織数と同程度となることを考え合わせると、システム全体での格納データサイズの合計は、参加組織数の三乗に比例すると考えられる。ただし、3.4.1 で述べた通り、各組織でのデータ容量は、企業システムのデータ容量として問題ない程度であるといえる。

また、実際には、ワークフロー案件ごとに参加する組織は異なることが考えられる。このため、情報の開示範囲を限定することが課題となる。

3.6 結言

本章では、2章で提案したブロックチェーンを活用した組織間ワークフローシステムのアーキテクチャで動作するワークフロー実行スマートコントラクトの実装方式として、**SDV** ワークフロー実行エンジンを提案した。提案方式を用いることで、ワークフローの実行中に、全ての関連データが組織間で整合性が取れた状態で処理されていることを保証することができる。この方式では、ワークフローのステップ数が増加した際の保存データ量の増大と処理性能の低下が懸案となるが、複数組織間での顧客情報登録ユースケースを想定した実装評価の結果、保存データ量、処理時間ともに実用上問題ないことを確認した。ただし、ワークフローに参加する組織に限定されたシステムが前提であり、不特定多数の組織が参加するシステムを対象とした場合には、参加組織に比例して増大するデータ容量の問題への対応、および情報の開示範囲の限定が課題となる。

第4章

情報開示範囲の限定と冗長性のないデータ構造による組織間ワークフロー管理システム

4.1 緒言

第3章では、ワークフローのステップ間でのデータの一貫性を保証する SDV ワークフロー実行エンジンの実装方式を提案した。しかし、SDV 方式では、すべての参加組織のブロックチェーンノードに同じデータを格納するため、特定の組織間でのワークフローを実行する際にデータの開示範囲を限定することができないという問題があった。また、ノード間でデータが重複するため、参加組織数が増加した場合に、システム全体でのデータ量が増大するという問題もあった。本章では、SDV ワークフロー実行エンジンにおけるこれらの問題を解決する方式として、参加組織が増えた場合のデータ漏洩防止とデータ量増大防止を実現するための方式を提案する。

データの漏洩を防止するために、ブロックチェーン上の特定のトランザクションに関連するデータの開示を特定のノード・グループに限定する機能である、Hyperledger Fabric の Private Data Collection(PDC)を利用する。但し、PDC を利用するだけでは、ワークフローのステップ間でのデータ一貫性を保証することができない。このため、ワークフローの各ステップにおいて、他のノードに保存されたハッシュ値と自ノードのデータを比較することで、データの冗長性を排除しつつステップ間のデータの一貫性を保証する方式を提案する。この際、PDC の構成では、他のノードに保存されたハッシュ値を高速に取得することができない。このため、PDC にキーバリューストアを追加し、処理性能の課題を解決する構成を併せて提案する。上記のような方式によるワークフロー実行エンジンを実装し、処理性能とデータサイズの観点からその実現可能性を評価する。

以下、4.2 節では、SDV 方式の課題として、不特定多数の組織が参加する場合のデー

データの開示範囲の限定方法と、データ量の増大への対応があることを示す。これを解決する方式として、4.3 節では、複数組織間のワークフローの機密性を確保し、冗長性を排除してデータ量を削減する SDVCD (Sequential Data Verification with Controlled Disclosure) ワークフロー実行エンジンを提案する。その後、4.4 節で処理性能、データ量の評価結果と、SDV 方式と SDVCD 方式の比較評価の結果を述べる。

4.2 安全なワークフロー管理システムのためのアプローチ

SDV ワークフロー実行エンジンでは、参加組織すべてがブロックチェーンノードを保有し、それぞれのブロックチェーンデータストアに同じデータを保存する。例えば、Figure 2-10 に示したユースケースでは、Org.A, Org.B, Org.C それぞれがブロックチェーンノードを保有し、それぞれのノードには同じデータが保存される。このシステムを用いて、別のワークフローとして、例えば Org.A と Org.B のみが利用する顧客情報を登録するワークフローを処理する場合、このワークフローには Org.A と Org.B のみが参加し、Org.C は参加しないが、SDV ワークフロー実行エンジンを用いた場合、このワークフローの処理結果は Org.C が保有するノードにも格納される。しかし、このワークフローの情報は、Org.A と Org.B のみが利用する顧客情報を含むため、Org.A と Org.B のみに開示されるべきものであり、Org.C には開示されてはならない。このように、SDV ワークフロー実行エンジンでは、データを特定の組織に秘匿することができないという問題がある。

このような問題に対処するため、Hyperledger Fabric は、特定のトランザクションに関連するデータの開示を特定のノード・グループに限定する方法として、PDC を提供している。PDC を SDV ワークフロー実行エンジンに適用した場合のシステムアーキテクチャを Figure 4-1 に示す。

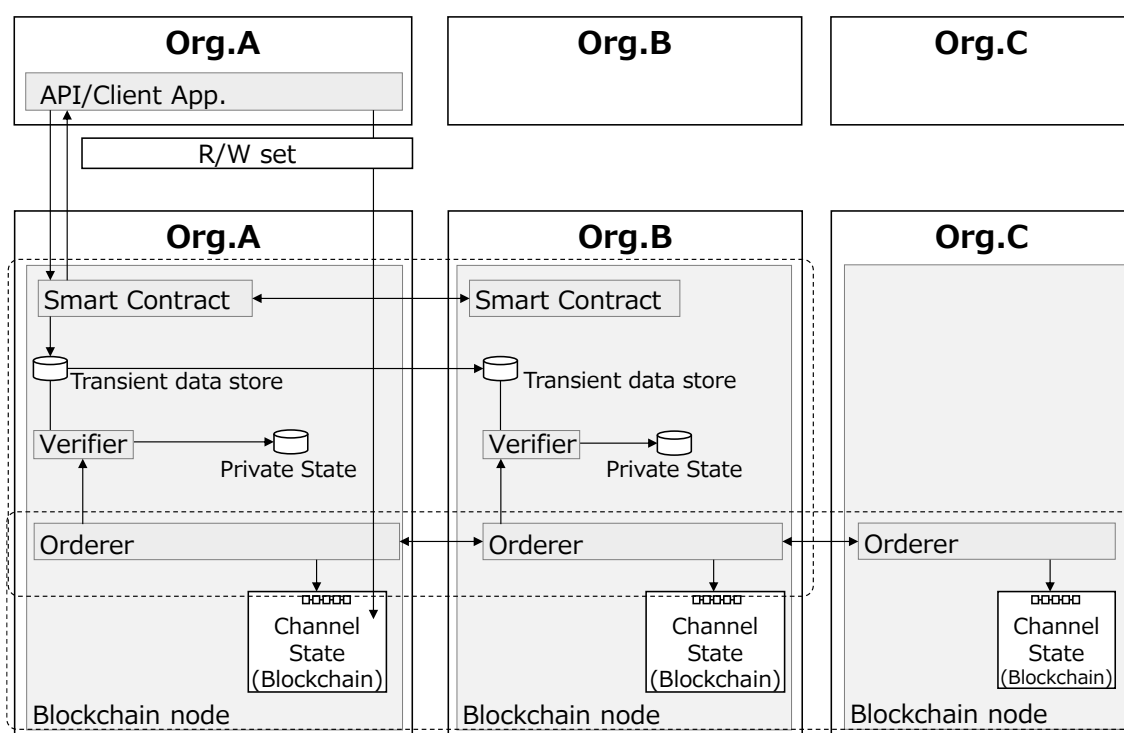


Figure 4-1 Transaction flow using PDC of Hyperledger Fabric

Private State には、ノードがアクセス権を持つトランザクションの生データが格納され、**Channel State** には全トランザクションのハッシュ値が格納される。クライアントからのトランザクションリクエストはアクセス権を持つノードに送信され、スマートコントラクトによって処理されて **R/W** セットが生成される。複数のノード間で **R/W** セットを比較・検証した後、処理結果は **Private State** に格納される。同時に、処理結果のハッシュ値が全ノードに配布され、**Channel State** に格納される。このように、アクセス権を持つノードは改ざんのない生データを共有し、アクセス権を持たないノードにもデータのハッシュ値を配信することで、**Private State** に格納されたデータの正しさを他のノードが検証することができる。

組織間ワークフロー管理システムでは、参照データを含むすべてのステップの関連するワークフローデータが、各ステップで改ざんされていないことを保証する必要がある。この要件は、特定の組織間でのみワークフローが実行される場合にも同様に必要である。したがって、**SDV** ワークフロー実行エンジンで 사용되는データ構造を使用することで、改ざんがないことを検証することが可能である。しかし、**SDV** ワークフロー実行エンジンのデータ構造には冗長性があり、処理ステップに比例してデータ量が増加する。また、ブロックチェーンの特性上、システムを運用する限り、全てのデータを保持する

必要がある。そのため、保管コストが増加し続けるという問題がある。

PDC の場合、Figure 4-1 の処理フローに示すように、Channel State は全ノードにおいて、全てのトランザクションのハッシュ値を格納する。ハッシュ値は、特定のトランザクションに参加していないノードがデータの正しさを検証する目的で格納される。このハッシュ値を利用すれば、参照データを含む関連する全てのワークフローデータを R/W セットに含めて比較する代わりに、他ノードから全てのステップのワークフローデータのハッシュ値を取得し、これを自ノードのデータと比較することによって検証する方法が考えられる。この方法が実現できれば、SDV ワークフロー実行エンジンと同様に各ステップにおけるデータの整合性を保証しつつ、SDV ワークフロー実行エンジンで採用されているデータ構造の冗長性を排除し、ストレージサイズを節約することができる。但し、PDC では、各ノードの Channel State はブロックチェーン形式のデータ構造で保存されるため、上記のような目的でハッシュ値を参照しようとしても、データの検索処理に膨大な時間がかかるという問題がある。また、PDC におけるスマートコントラクトには、当該トランザクション以外のデータを他ノードから取得する機能はないため、これを実現する方式をスマートコントラクトから呼び出し可能とする必要がある。

4.3 節では、上記のような課題を解決し、複数組織間のワークフローの機密性を確保し、冗長性を排除してデータ量を削減する方式として、SDVCD (Sequential Data Verification with Controlled Disclosure) ワークフロー実行エンジンを提案する。

4.3 SDVCD ワークフロー実行エンジン

4.3.1 データモデル

SDVCD ワークフロー実行エンジンでは、ワークフローの現在のステップより前のステップの参照データは Channel State のハッシュ値を用いて検証し、現在のステップのトランザクションデータは R/W セットを他のノードと比較することで検証する。R/W セットには各ステップの要求データのみが含まれる。SDVCD エンジンで 사용되는データ構造を Figure 4-2 に示す。最初のワークフロー起案では、ワークフロー案件 ID、依頼や承認などの各トランザクションのステータス、ドラフトの内容、後続レコードへのリンクを示す Linked To、Action ID からなるレコードを生成する。承認な

どの後続アクションでは、レコードの内容、担当者、起案内容の修正内容である Contents(diff)からなるレコードが生成される。このレコードは R/W セットの単位である。

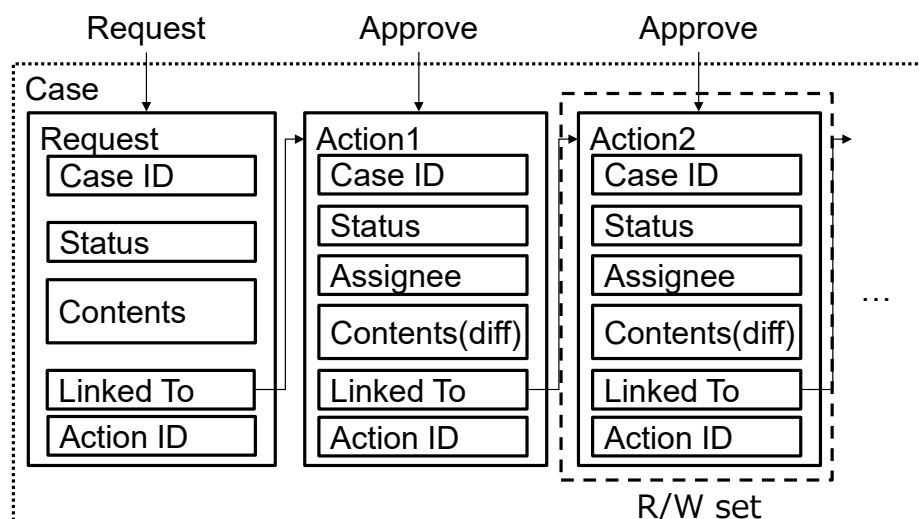


Figure 4-2 Data model for SDVCD workflow execution engine

SDV ワークフローエンジンでは、Figure 3-5 に示したように、R/W セットに当該ワークフロートランザクションの全データを含めるために、各ステップの各レコードにアクションが追加されるデータ構造となっており、冗長になっている。これに対し、SDVCD ワークフローエンジンのデータ構造では、R/W セットに含まれるデータは各ステップのトランザクションのレコードに限定され、それ以前のステップのレコードは Channel State に格納されたハッシュ値と Private State に格納されたデータから生成されたハッシュ値を比較することで検証される。そのため、R/W セット自体には 1 ステップ分のレコードのみを格納すればよく、データの冗長性がなくなる。これにより、保存データ量が削減される。

4.3.2 システムアーキテクチャとトランザクションの流れ

PDC 機能を利用した SDVCD ワークフロー実行エンジンを実現するためには、以下の 2 つの課題を解決する必要がある。

課題 1 ; Channel State はハッシュチェーンによるブロックチェーン形式のデータ構造で保存されるため、トランザクション実行時に Case ID をキーとして高速に検索する方法が求められる。

課題 2 ; スマートコントラクトに、現在のステップ以外の全ステップのデータを検証するプロセスを追加する必要がある。

課題 1 に対しては、ブロックチェーン形式のデータ構造によるデータベースに加えて、KVS (Key Value Store) 型のデータベースを追加する。これにより、キーに基づく高速なデータ検索・取得を可能とする。SDVCD 方式によるワークフロー実行エンジンのシステムアーキテクチャを Figure 4-3 に示す。SDVCD 方式では、各トランザクションの処理時に、ブロックチェーン形式のデータと KVS 形式のデータを同時に書き込む。Table 4-1 に KVS のデータ構造を示す。各レコードは、キーとなる Case ID, 比較単位となる Action ID, レコードのハッシュ値から構成される。SDVCD 方式による実行エンジンでは、ワークフロー案件の全レコードを Case ID で検索し、Action ID 毎にハッシュ値で比較する。

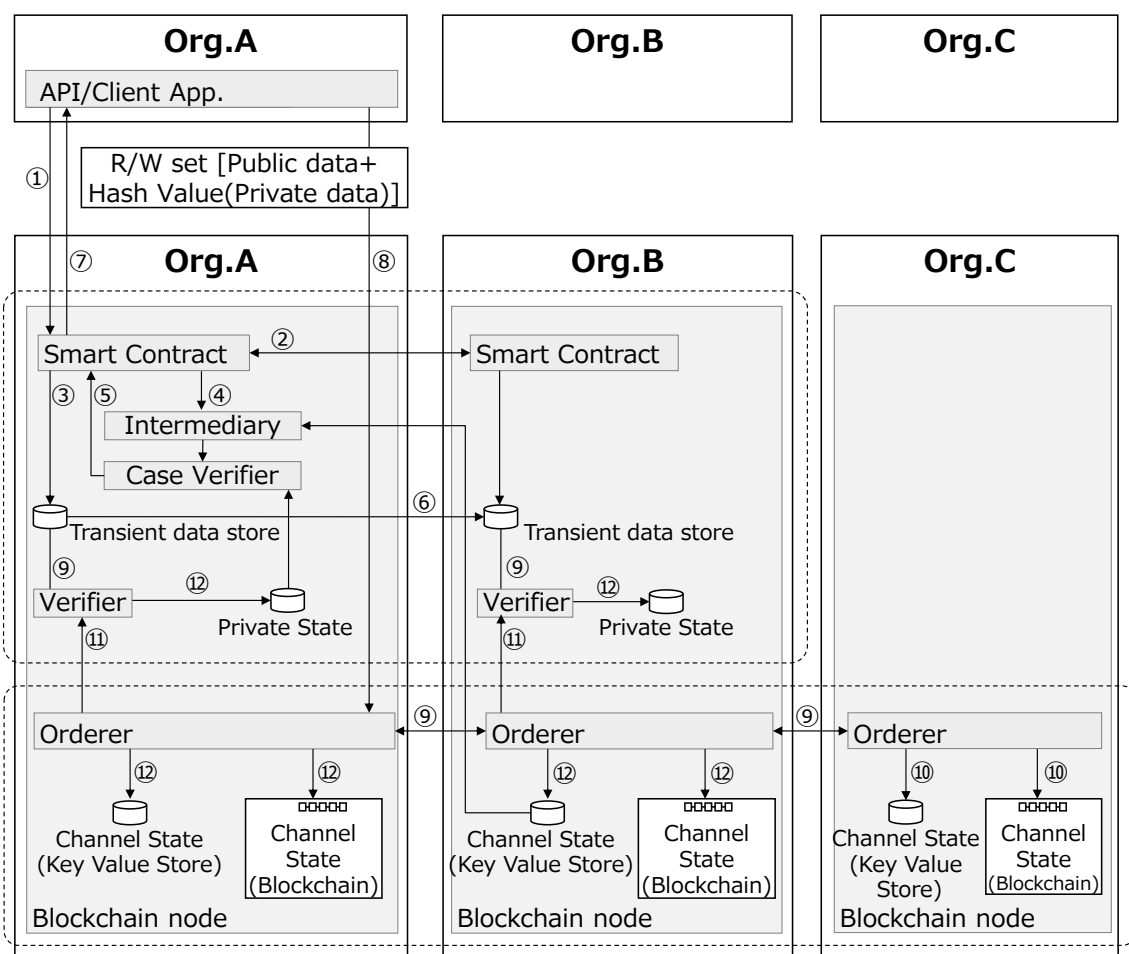


Figure 4-3 Transaction flow in proposed architecture

Table 4-1 Data model of channel state in key value store

Case ID	Action ID	Hash Value
0011	002	0f4b...
0011	003	bc06...
...	...	...

次に、課題 2 に対しては、他ノードの **KVS Channel State** に格納されたハッシュ値とローカルの **Private State** に格納されたデータのハッシュ値を比較することにより、ワークフロー案件全体のデータの整合性を確認するため、他ノードのハッシュ値を取得する **Intermediary** 機能と、ハッシュ値同士を比較する **Case Verifier** 機能を追加し、これらを **Smart Contract** から呼び出す。これにより、**R/W** セットに当該ワークフロートランザクションの全データを含めず、ワークフローの全ステップのデータの整合性を確認することが可能となる。

これらの機能により、複数組織間のワークフローにおけるデータの機密性を確保するとともに、冗長性を排除してデータ量を削減する、組織間ワークフロー実行エンジンが実現できる。

提案する **SDVCD** ワークフロー実行エンジンの処理フローを、**Figure 4-3** を用いて説明する。ステップ 1 で、クライアントがノードにワークフロー処理要求や承認などのトランザクションを送信すると、ステップ 2 で、トランザクションデータが、このワークフローに参加するアクセス権限を持つノードに転送される。トランザクションを受信した各ノードではスマートコントラクトが実行され、ステップ 3 で、このトランザクションで読み込まれたデータと処理結果として格納されるデータを組み合わせた実行結果である **R/W** セットを生成し、**Transient Data Store** に格納する。ステップ 4 で、**Intermediary** は、他ノードの **KVS** 型 **Channel State** に格納されているワークフロー案件の全レコードのハッシュ値を取得し、**Case Verifier** に渡す。ステップ 5 で、**Case Verifier** は、受け取ったハッシュ値と **Private State** から生成されたハッシュ値とを比較検証する。**Case Verifier** を追加したことにより、**SDV** 方式のように **R/W** セットにワークフロー案件の全データを含めて比較するのではなく、ワークフロー実行エンジンである **Smart Contract** の処理の中でワークフロー案件の他のステップのデータを比較検証することができる。これにより、**R/W** セットのデータ量を削減することができる。こ

の際、元々の PDC の構成では、他ノードのデータはブロックチェーン形式で格納されているため、検索性能が出ないという問題があった。これに対し、SDVCD 方式では、同じデータを KVS 型の Channel State として保持する構成としたことにより、格納データを高速に取得することが可能となる。

次に、ステップ 6 では、R/W セットが、このトランザクションのアクセス権を持つノードの Transient Data Store に配布される。ステップ 7 では、R/W セットのハッシュ値が Transaction Proposal としてクライアントに送信される。ステップ 8 で、クライアントは複数のノードからのトランザクション提案を比較し、それらが一致した場合、クライアントはそのトランザクション提案を Orderer に送信する。ステップ 9 で、Orderer はトランザクションの格納順序を決定し、ブロックデータを生成して全ノードに送信する。アクセス権を持たないノードは、ステップ 10 でこのデータをブロックチェーン型 Channel State に書き込み、同じデータを KVS 型 Channel State にも書き込む。ステップ 11 では、まずアクセス権を持つノードが、ステップ 6 で既に配信された Transient Data のハッシュ値を取得する。そして、KVS Channel State のハッシュ値と一致するかどうかを検証する。一致した場合、ノードはステップ 12 で Transient Data を Private State に転送し、ブロックチェーン型 Channel State と KVS 型 Channel State にそれぞれ書き込む。

SDVCD 方式では、ステップ 4 とステップ 5 において、Intermediary 機能と連携した Case Verifier 機能により、ワークフロー案件の全レコードのデータを他ノードに格納されたデータと比較する処理を追加した。この処理により、SDV 方式のように、ワークフロー案件の全レコードのデータを R/W セットに含めることなく、各ステップでワークフロー案件の全てのレコードが変更されていないことを確認することができる。このため、R/W セットには必要最小限のデータ、つまり当該ステップの読み込み、書き込みデータのみを格納すればよいため、Channel State とブロックチェーンへの書き込みデータ量も削減することができる。

4.4 SDVCD ワークフロー実行システムの評価

4.4.1 評価条件

提案する SDVCD ワークフロー実行エンジンを実装し、データ量と性能の観点から

評価した。SDVDC 方式の目的は、組織間ワークフロー管理システムに複数の組織が参加している場合に、その中の一部の組織のみが参加するワークフローを処理する際、データの開示範囲を参加組織に限定しつつ、データの冗長性を排除してシステム全体のデータ量を削減することにある。各組織のノードに格納されるデータ量は、特定のワークフローに参加するかどうかによって決定される。したがって、システム全体のデータ量は、特定のワークフローに参加する組織の数と参加しない組織の数によって決定される。処理性能は、ブロックチェーン基盤の合意形成アルゴリズムとストレージへの書き込みデータ量、ブロックチェーンネットワークの構成によって決定される。合意形成には、特定のワークフローへの参加有無によらず、全ノードが参加可能である。また、合意形成アルゴリズムには、一定数のノードからの回答のうち、過半数が一致することで合意とみなすアルゴリズムを用いることにより、ブロックチェーンネットワークへの参加組織数によらず、一定の処理性能を実現できる。

上記の条件を踏まえ、特定のワークフローに参加する組織と参加しない組織が混在するユースケースとして、Figure 4-4 に示すユースケースを実装し、提案する SDVCD ワークフロー実行エンジン(b)と、SDV ワークフロー実行エンジン (a)を比較、評価した。

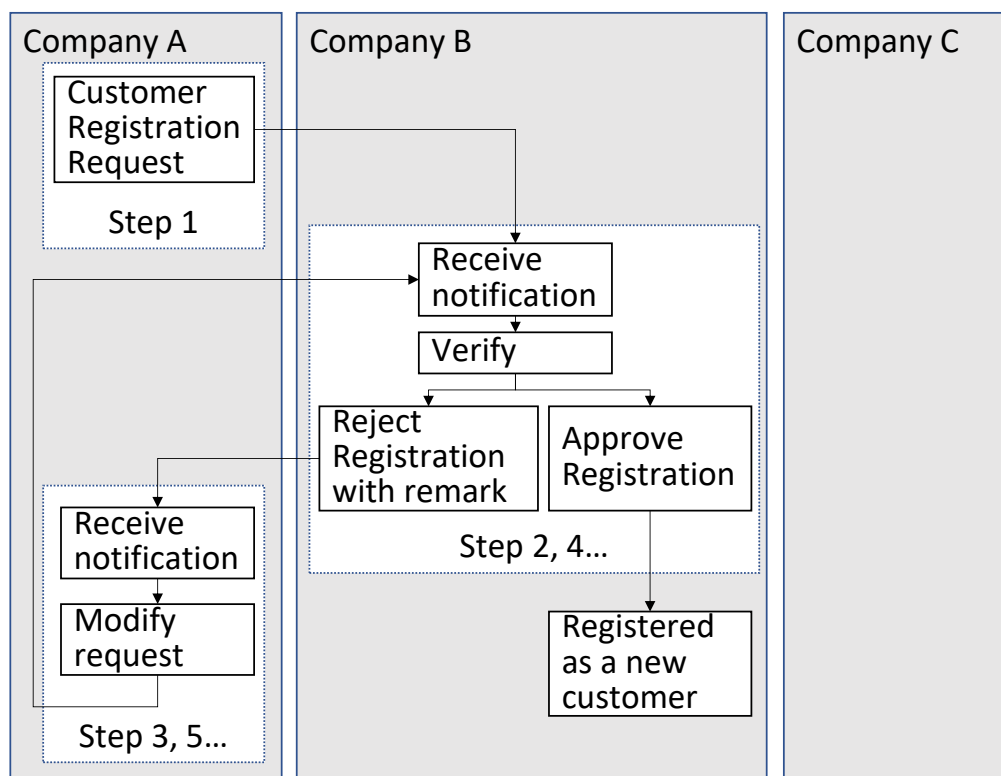


Figure 4-4 Evaluated use case of cross organizational workflow

Figure 4-4 に示すユースケースは、3 組織のうち 2 組織が参加し、2 組織が連携してサービスを提供するシステムへのユーザ登録を承認するビジネスプロセスである。このユースケースでは、ステップ 1 で参加組織の 1 つである組織 A が顧客登録リクエストを送信し、ステップ 2 で組織 B がリクエストの内容を確認し、承認または却下する。リクエストが承認されれば、ユーザは顧客として登録され、ワークフローは完了する。却下された場合は組織 A に戻り、修正後、組織 A は再度組織 B にリクエストを送り、承認を得る。組織 C は、組織間ワークフロー管理システムに参加している組織であるが、本サービスおよびワークフローには参加していない。このユースケースにより、組織間ワークフローシステム内に参加している組織のうち、特定のワークフローに参加する組織と参加しない組織におけるデータ量、および組織間ワークフローシステムへの参加組織数が増加した場合のシステム全体のデータ容量を評価する。Table 4-2 に評価環境を示す。ブロックチェーンネットワークは 3 つのノードで構成され、3 つのノードのうち 2 つのノードの合意によってコンセンサスが成立する **Endorsement policy** を採用する。組織間ワークフローシステムへの参加組織数が増大した場合でも、同様の **Endorsement policy** を適用すれば、2 つのノードでの合意によってコンセンサスが成立するため、同等の性能を実現できる。

Table 4-2 Specifications of evaluation environment for SDVCD

Item	Spec
CPU	Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz
Memory	16GB
OS	ubuntu24.04, Linux Kernel 5.15.0-1053-aws
Blockchain Platform	Hyperledger Fabric v2.5.9
Database	CouchDB
Language	Golang (Smart Contract) typescript (Client program)
Number of nodes	3
Endorsement policy	2 out of 3

4.4.2 データ量評価

顧客登録のユースケースにおける 1 リクエストあたりのデータ量は、3.4.1 項で示したとおり約 1kB である。このサイズに基づき、各データストアのサイズを Figure 4-5 に、ワークフロー参加組織のノードと不参加組織のノードの総ストレージサイズを Figure 4-6 に示す。

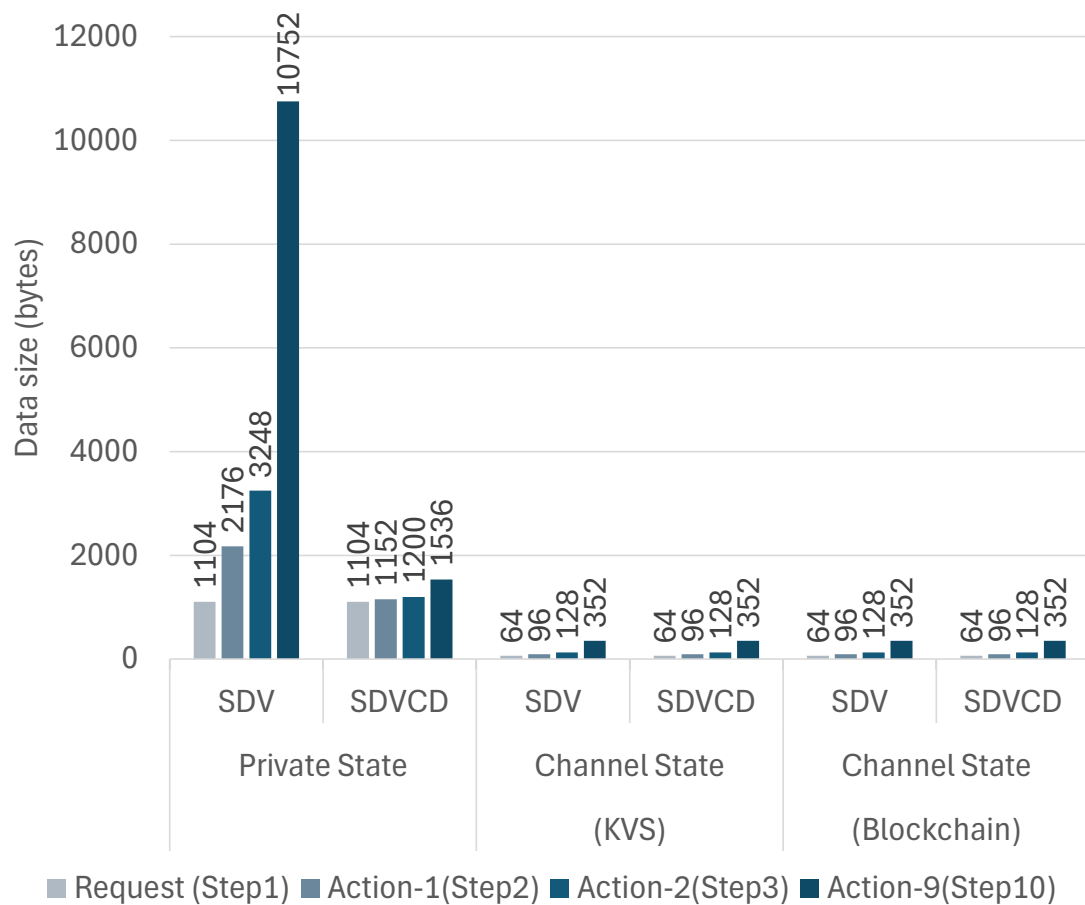


Figure 4-5 Data size of each data store

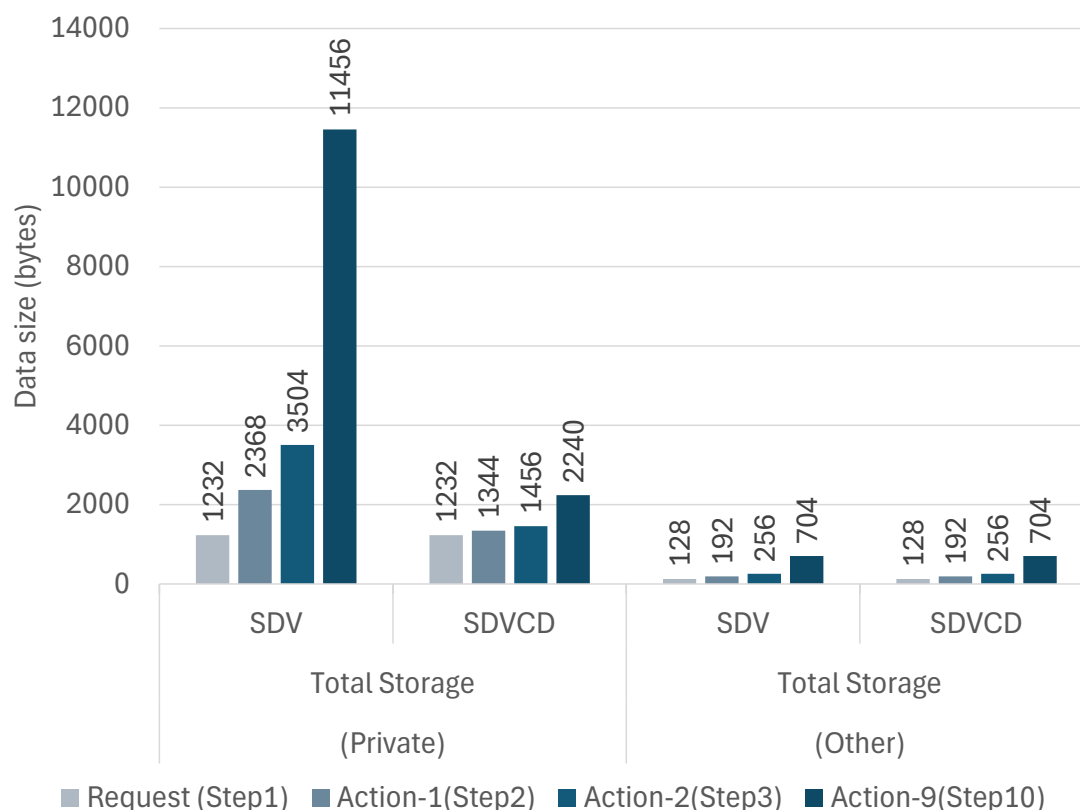


Figure 4-6 Total storage size

SDV 方式では、全てのワークフローデータが 1 レコードに格納されるため、ステップ数に応じて **Private State** のデータサイズが大きく増加するが、1 ステップのデータしか含まれない **SDVCD** 方式ではデータサイズの増加は小さい。また、PDC を利用することにより、**KVS 型 Channel State**、**Blockchain 型 Channel State** ともにハッシュ値が格納されるため、データ量が大幅に削減される。**SDVCD** 方式では、10 ステップのワークフローの場合、SDV 方式と比較して、特にワークフロー参加ノードにおけるストレージサイズ(**Total Storage (Private)**)を 20%程度に削減できる。

4.4.3 性能評価

クライアントアプリケーションがリクエストを送信してから処理結果を受信するまでの応答時間を、リクエストデータサイズを 10 段階に変化させて測定した結果を **Figure 4-7** に示す。横軸は 1 リクエストあたりのデータサイズ、縦軸は応答時間であ

る。

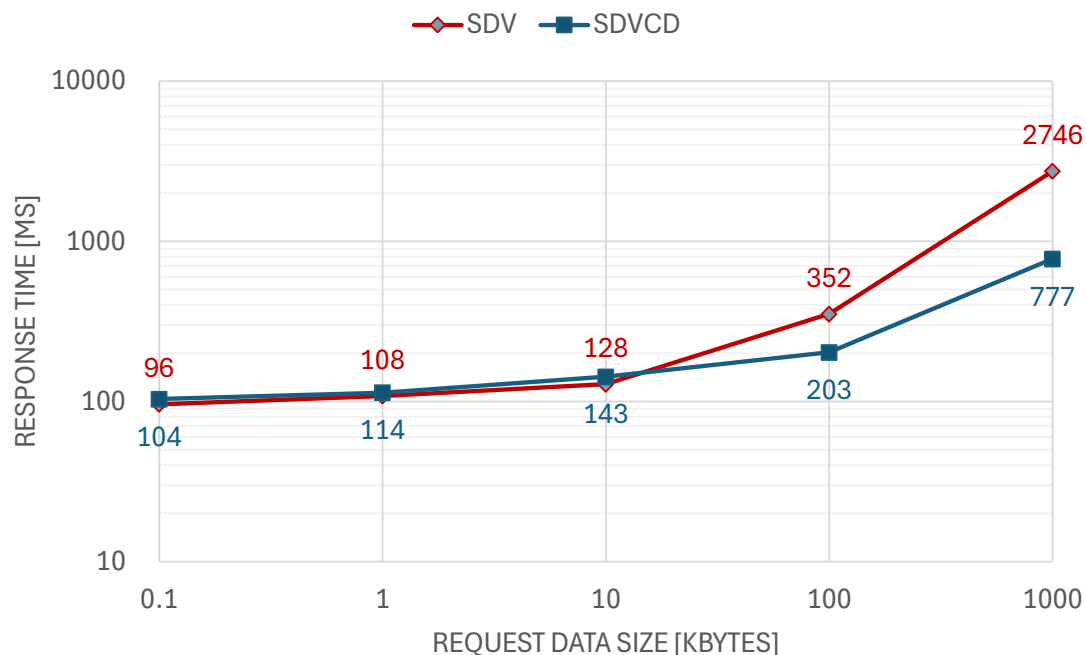


Figure 4-7 Response time by data size

リクエストデータは、**Private State** に書き込まれるデータであり、Figure 4-5 に示す通り、10 ステップ目のデータサイズは、SDV 方式で約 10kB、SDVCD 方式で約 1.5kB である。3.5 節で述べた通り、一般に組織間ワークフローでは、参加組織数と同程度のステップ数となることが想定されるが、10 組織が参加するような大規模な組織間ワークフローでも、SDV 方式、SDVCD 方式のいずれも 100ms 程度で処理可能であり、実用上問題ないことがわかる。リクエストに添付ファイルを含むなど、リクエストデータのサイズが 1MB になった場合でも、SDV 方式で 2746ms、SDVCD 方式で 777ms と、人間が操作する業務システムとしては許容できる応答時間である。データサイズが 1MB を超える場合は、生データを **Private State** ではなく外部ストレージに格納し、ハッシュ値のみを **Private State** に格納する方式が採用できる。但し、この場合、外部ストレージに格納されたデータに改ざんがないことは **Private State** のハッシュ値を用いて確認できるが、外部ストレージに格納されたデータの改ざんを防ぐ方法や、データを改ざんされた場合のデータの復元方法などは個別に設計する必要がある。

SDV 方式と SDVCD 方式では、データサイズが 10kB を超える場合の応答時間に差

がある。Figure 4-8 にリクエストデータサイズが 1MB の場合の処理時間の内訳を示す。

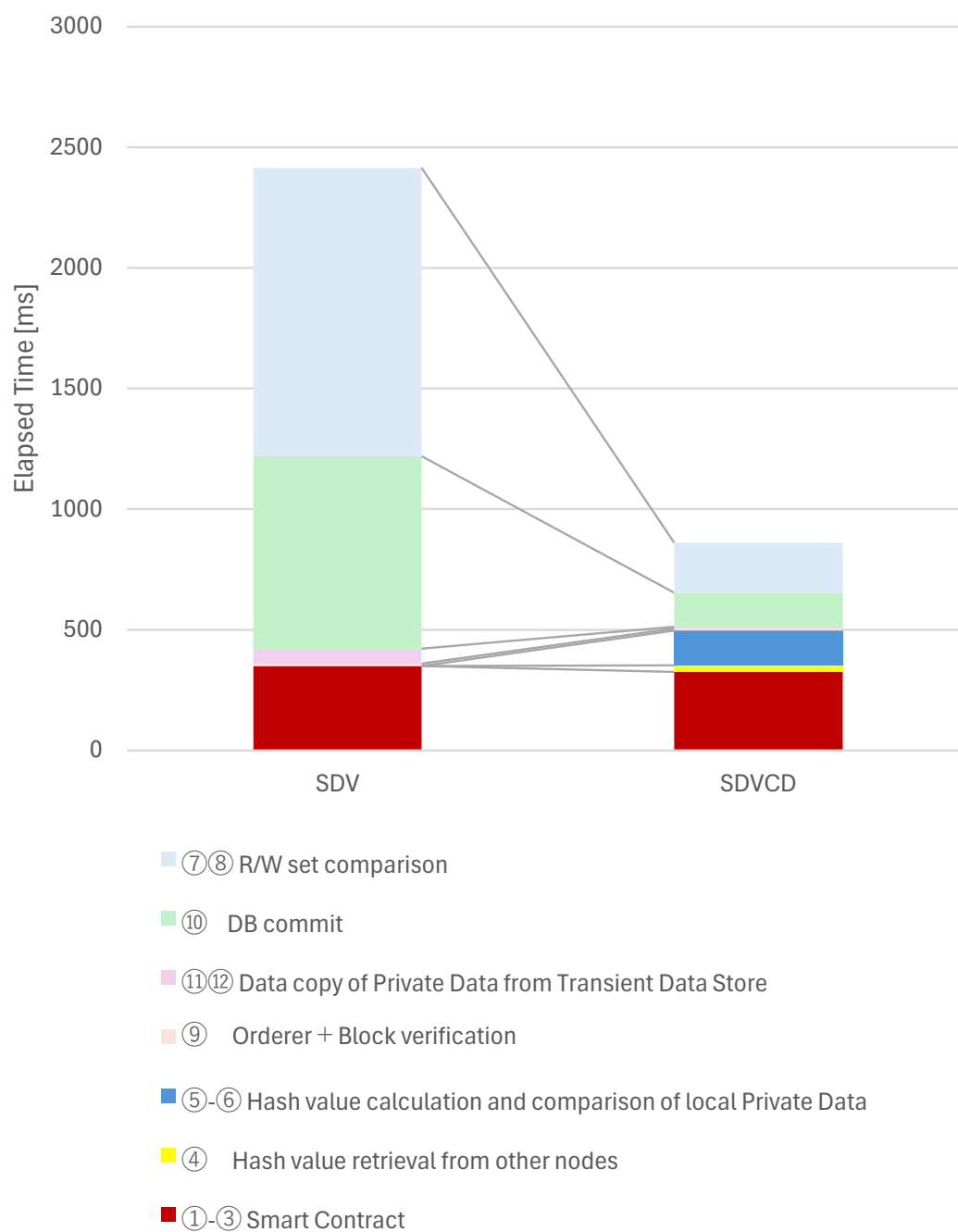


Figure 4-8 Breakdown of processing time

SDV 方式では、ワークフローデータをすべて R/W セットに含めるため、Private State

に書き込まれるデータサイズがリクエストデータサイズに対して大きくなる。このため、Figure 4-8 に示す通り、ステップ 7, 8 における R/W セットの比較や、ステップ 10 におけるステート DB へのコミット、ステップ 11, 12 における Private State へのデータコピーの処理時間が SDVCD 方式よりも大きくなる。一方、SDVCD 方式では、4.3.2 で説明したように、全てのワークフローデータを R/W セットに含めてデータベースに格納するのではなく、他のノードからハッシュ値を取得し、スマートコントラクト内で全ステップのデータを比較検証する。この処理はストレージにデータを書き込まないため、処理時間は比較的短く、応答時間は SDV 方式よりも短い。但し、SDVCD 方式では、ステップ 4 における Intermediary による他ノードからのデータ取得や、ステップ 5, 6 における Case Validator によるハッシュ値の比較処理が追加になるが、これらの処理時間は SDV 方式における R/W セットの比較、Channel State へのコミット、Private State へのデータコピーの処理時間の増加分に対して十分に小さい。これらの理由により、SDVCD 方式による処理時間は、SDV 方式と比較して大幅に短くなっている。

4.4.4 SDV 方式と SDVCD 方式の比較評価

SDVCD 方式によれば、不特定多数の組織が参加する組織間ワークフローシステムにおいて、特定の複数組織のみが参加するワークフローデータへのアクセスを参加組織のみに限定できることを確認した。同様の機能を SDV 方式で実現するために、組織間ワークフローの案件毎に参加組織のみからなるブロックチェーンネットワークを形成し、SDV 方式による実行エンジンによってワークフローを実行することも可能である。ただし、参加組織の構成が変わる場合には、各ワークフロー案件の参加組織からなるブロックチェーンネットワークをその都度構成する必要があり、運用コストが大きい。

また、Figure 4-6 に示した通り、SDV 方式ではデータモデルの特徴により、ワークフローのステップ数が増えれば増えるほど、ストレージ容量が著しく増大するという問題がある。例えば、1 ステップ目では、SDV 方式、SDVCD 方式ともに 1 ワークフローあたりの総ストレージ量は 1232bytes であるのに対し、10 ステップ目では SDV 方式では 11456bytes となり、SDVCD 方式の約 5 倍のストレージ量が必要となる。一方で、SDVCD 方式では、特定のワークフローに参加しない組織においても、ハッシュ値を格

納するために一定のストレージを消費する。これに対し、SDV 方式を用いて、組織間ワークフローの案件毎に参加組織のみからなるブロックチェーンネットワークを形成した場合、自組織が参加するワークフローに関する情報のみを格納するだけである。

このように、ワークフロー案件毎にどの程度参加組織の構成が異なるかという点と、それぞれのワークフローのステップ数がどの程度かという点によって、SDV 方式とSDVCD 方式のいずれを利用すべきかを判断する必要がある。この際の評価観点としては、ストレージ容量、処理性能、運用性、が考えられるが、Figure 4-7 に示した通り、処理性能はデータサイズが 100kB 程度までは両方式で大差がないため、ここではストレージ容量と運用性について評価する。

前述の通り、ワークフロー案件のステップ数と、各ワークフロー案件の構成組織の複雑さによって特性が変化するため、A, B, C の 3 つの組織において 10 案件のワークフローを処理する場合において、以下の 4 つのシナリオで評価を行った。

- (1) 組織 A-B 間の 2 ステップの WF を 10 案件処理する場合（シナリオ 1）
- (2) 組織 A-B 間の 10 ステップの WF を 10 案件処理する場合（シナリオ 2）
- (3) 組織 A-B 間の 2 ステップの WF を 3 案件、組織 A-C 間の 2 ステップの WF を 3 案件、組織 B-C 間の 2 ステップの WF を 4 案件処理する場合（シナリオ 3）
- (4) 組織 A-B 間の 10 ステップの WF を 3 案件、組織 A-C 間の 10 ステップの WF を 3 案件、組織 B-C 間の 10 ステップの WF を 4 案件処理する場合（シナリオ 4）

それぞれのシナリオにおける各組織、及び 3 組織合計でのデータサイズの比較結果を Table 4-3, Table 4-4 に示す。

Table 4-3 Comparison of data size between SDV and SDVCD (1) (bytes)

	Scenario 1		Scenario 2	
	SDV	SDVCD	SDV	SDVCD
Organization A	65600	13440	1185600	13440
Organization B	65600	13440	1185600	13440
Organization C	0	1920	0	1920
Total	131200	28800	2371200	28800
SDVCD/SDV(%)		21.95%		1.21%

Table 4-4 Comparison of data size between SDV and SDVCD (2) (bytes)

	Scenario 3		Scenario 4	
	SDV	SDVCD	SDV	SDVCD
Organization A	39360	8832	711360	16256
Organization B	45920	9984	829920	17792
Organization C	39360	8832	711360	16256
Total	124640	27648	2252640	50304
SDVCD/SDV(%)		22.18%		2.23%

いずれのシナリオでも、SDVCD 方式は SDV 方式と比較して、3 組織合計でのストレージ容量が最大でも 20%程度に削減できることが分かった。特に、シナリオ 2、4 の

ようなステップ数の多いワークフローを多数処理する場合には、SDVCD 方式は SDV 方式に対して 1~2%程度のストレージ容量に削減できる。

さらに、シナリオ 3、4 のように、複数の組織の組み合わせによるワークフローを処理する場合、SDV 方式では組み合わせの数のブロックチェーンネットワークを構成する必要がある。すなわち、各組織は、参加する組み合わせの数のブロックチェーンノードを運用する必要がある。これに対し、SDVCD 方式では、ワークフロー案件毎にワークフロー定義を変更することにより、参加組織のみにデータを格納する構成で運用することができる。一方、シナリオ 1、2 のように、特定の組織の組み合わせにワークフロー案件が集中する場合、SDVCD 方式ではワークフロー案件に参加しない組織もブロックチェーンノードを運用し、かつ一定のデータ（ハッシュ値）を格納し続ける必要がある。このため、ワークフロー処理を頻繁には利用しない組織のコスト負担を考慮する必要がある。

以上をまとめると、特定の組織間でステップ数の少ないワークフローを固定的に処理する場合には、SDV 方式の方がストレージ容量とノード運用コスト、運用性の観点から有利であると言える。一方、ステップ数の多いワークフローを多数処理する場合や、ワークフロー案件ごとに参加組織が異なるような動的な定義による運用を行う場合には、SDVCD 方式の方がストレージ容量、運用性の観点から有利であると言える。

4.5 考察

評価の結果、SDVCD 方式によれば、情報の開示範囲をワークフロー参加者に限定することができるとともに、SDV 方式と同等の処理性能を実現し、かつデータ量を 20%に削減できることを確認した。処理性能評価では、3 組織が参加する組織間ワークフロー管理システムにおいて、2 組織が参加するワークフロー処理を例に評価したが、ブロックチェーンネットワークにおける処理性能は、参加ノード数ではなく **Endorsement policy** に依存するため、同じ **Endorsement policy** を採用する限り、参加組織数が増大した場合も同等の性能を実現できると考えられる。但し、複数のノードからの回答により合意形成を行うため、ブロックチェーンネットワークの構成には影響を受ける。また、SDV 方式が固定的かつステップ数の少ないワークフローに適する一方、動的な組織の組み合わせによるワークフローやステップ数の多いワークフローにおいては、SDVCD 方式が適することを明らかにした。但し、SDVCD 方式では、Figure 4-6 に示した通り、

ワークフローに参加していない組織も、10 ステップのワークフロー1 件あたり 704 bytes のストレージを消費する。また、データの送受信や他ノードからのハッシュ値取得要求への応答により、CPU リソースも消費される。従って、ワークフロー処理の利用が特定の組織に偏在している場合には、その利用状況に応じて、IT リソースのコストをネットワーク参加組織間で分担する工夫が必要となる。

4.6 結言

本章では、ワークフローに関係する組織のみに情報開示範囲を限定しつつ、最小限のデータサイズで改ざん防止を保証する組織間ワークフロー実行エンジンを提案した。提案エンジンを顧客登録ワークフローのユースケースに適用した結果、SDV ワークフロー実行エンジンと同等の性能を実現し、データサイズを 20%削減できることを確認した。また、SDV 方式が固定的かつステップ数の少ないワークフローに適する一方、動的な組織の組み合わせによるワークフローやステップ数の多いワークフローにおいては、SDVCD 方式が適することを明らかにした。このような組織間ワークフロー実行エンジンを適用する具体的なケースとして、例えば金融や製造業などの業界ごとにワークフロー管理システムを運営し、業界内の特定の複数企業間で契約等のワークフロー処理を行うようなケースが考えられる。SDVCD 方式によれば、このような場合でも、ワークフロー毎に関係する企業だけに情報の開示を限定することができる。ただし、SDVCD 方式においては、全てのノードにワークフローデータのハッシュ値が格納されるため、ワークフローの利用量に関わらず、参加組織は最低限の IT リソースを負担する必要がある。従って、利用量が特定の組織に偏在している場合は、コスト分担を考慮する必要がある。

第5章

結論

5.1 本研究のまとめ

本論文では、組織間ワークフローを安全に実現するためのワークフロー管理システムについて検討し、ブロックチェーンを用いた高信頼な組織間ワークフロー管理システムのアーキテクチャ、データの改ざんを防止する組織間ワークフロー実行エンジンの処理方式、及び情報開示範囲の限定と冗長性を排除したデータ構造による組織間ワークフロー管理システムのアーキテクチャと処理方式を提案した。これらの研究成果を、以下の4章に分けて述べた。

第1章では、組織間のワークフローが依然として紙の書類や電子メールといった個別の通信手段を用いて処理されていることにより、多大な時間と労働コストを要している現状の問題を説明し、これを情報システム化する必要性を示した。その際に、第1の課題として、ワークフローの処理やデータを、全ての参加組織がワークフローのライフサイクル全体にわたって信頼可能とするためには、特定の組織やユーザに管理を依存する単一信頼点(SPoT)を排除したシステムアーキテクチャの実現が課題であることを示した。また、第2の課題として、そのようなシステムアーキテクチャにおいて、特にワークフローの実行フェーズにおけるデータとプロセスの改ざんを防止するような組織間ワークフローの実行エンジンの実現が必要であることを示した。さらに、第3の課題として、参加組織が拡大した際に、参加組織間で情報開示範囲を限定しつつ、データ量や処理性能を担保するための対応が必要であることを示した。

第2章では、組織間での利用を想定したコンソーシアム型ブロックチェーンを用い、組織間ワークフロー管理におけるライフサイクル全体にわたって SPoT を排除するシステムアーキテクチャを提案した。提案システムアーキテクチャでは、信頼できる組織間のセキュアなデータストアとしてブロックチェーンを利用するとともに、ワークフローの定義、実行、および監査をすべてブロックチェーン上のスマートコントラクトとして実装し、これらを連動させることにより、ワークフローのライフサイクル全体にわたって SPoT を排除し、参加組織の合意なしには処理プロセスやデータを改ざんできない

構成とした。ワークフローのライフサイクルにおける攻撃シナリオに基づいて、ワークフロー関連データの改ざん耐性を評価した結果、提案アーキテクチャによれば、組織間ワークフローのライフサイクル全体にわたって、データの改ざんを防止できることを確認した。

第3章では、組織間ワークフロー管理システムのアーキテクチャで動作するワークフロー実行スマートコントラクトの実装方式として、データの改ざんがないことをワークフローの各ステップで逐次確認する、SDV ワークフロー実行エンジンの実装方式を提案した。提案方式によれば、ワークフロー実行のステップ間でのデータの改ざんを防止できるため、ワークフロー案件全体のデータの整合性を保証することができる。ただし、提案方式では、ワークフローのステップ数が増加した際の保存データ量の増大と処理性能の低下が懸案となる。提案方式を用いて、複数組織間で顧客情報を共有する場合の顧客情報登録ワークフローを実装し、評価した結果、保存データ量、処理時間ともに実用上問題ないことを確認した。

第4章では、複数組織が参加する組織間ワークフロー管理システムにおいて、ワークフローに関係する組織のみに情報開示範囲を限定しつつ、最小限のデータ量でワークフローデータの整合性を保証し、参加者間のデータ漏洩を抑制する SDVCD 方式によるワークフロー実行エンジンを提案した。提案方式を実装評価した結果、SDV 方式に対してデータ量を20%以上削減できるとともに、SDV 方式と同等の処理性能を実現できることを確認した。また、SDV 方式が固定的かつステップ数の少ないワークフローに適する一方、動的な組織の組み合わせによるワークフローやステップ数の多いワークフローにおいては、データ量や運用性の観点で SDVCD 方式が適することを示した。

5.2 今後の課題

最後に、本研究における今後の課題について述べる。

本論文では、ブロックチェーン基盤を活用し、不特定多数の組織が参加する組織間ワークフローを、定義、実行、監査にわたるすべてのフェーズで特定の集中管理者に依存せずに実現する、信頼できる組織間ワークフロー管理システムを提案した。ただし、提案システムの実運用に向けては、以下のような運用面の課題を解決する必要がある。

(1) 分散型のシステム運用方法の実現

提案システムはブロックチェーン基盤を利用しているため、運用時にはどのような合意形成アルゴリズムを用いるか、合意形成に参加する組織ならびにノードをどのように定めるか、などの運用ルールを定め、参加組織の合意に基づいて運用する必要がある。このような運用ルールの策定やルールに基づくシステム運用を、ワークフロー処理と同様に SPoT を排除した構成で実現する方法が残された課題である。

(2)新規参加や離脱時の運用方法の実現

ブロックチェーンを活用したシステムでは、そのデータ構造の性質上、データの一部を削除することはできず、データは参加ノード全てに蓄積され続けることになる。本システムのブロックチェーンネットワークに新たに組織が参加する際には、過去のデータも含めたブロックチェーンデータ全てを新たに参加する組織のノードにコピーする必要がある。一方で、本システムからある組織が離脱する際には、当該組織が関与したワークフローのデータを他ノードから削除することはできない。このように、組織が新たに参加する際の効率的なノード立ち上げ方法、および組織が離脱する際に、機密情報をブロックチェーンネットワークから削除する、あるいは見えなくする運用方法の実現が残された課題である。

参考文献

- [1] R. Miles and K. Hamilton, Learning UML 2.0, O'Reilly Media, Inc., 2006.
- [2] H. E. Eriksson and M. Penker, "Business Modeling with UML," 2000. [Online].
- [3] B. Silver and B. Richard, BPMN Method and Style: Second Edition, with BPMN Implementer's Guide, Cody-Casidy Press, 2009.
- [4] D. Krafzig, K. Banke and D. Slama, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall Professional., 2005.
- [5] R. Acharya, "EAI: A Business Perspective," *eAI Journal*, pp. 37-44, 2003.
- [6] J. Lee, K. Siau and S. Hong, "Enterprise Integration with ERP and EAI," *Communications of the ACM*, vol. 46, no. 2, pp. 54-60, 2003.
- [7] 小林隆, ビジネスプロセスのモデリングと設計, コロナ社, 2005.
- [8] M. Havey, Essential Business Process Modeling, O'Reilly Media, 2005.
- [9] R. K. Ko, S. S. Lee and E. Wah Lee, "Business Process Management (BPM) Standards: A Survey," *Business Process Management Journal*, vol. 15, no. 5, pp. 744-791, 2009.
- [10] H. A. Reijers, "Implementing BPM Systems: The Role of Process Orientation," *Business Process Management Journal*, vol. 12, no. 4, pp. 389-409, 2006.
- [11] R. M. Shapiro, B. Silver, L. Fischer, S. A. White, M. Z. Muehlen, N. Palmer, D. Gagne , T. Allweyer, BPMN 2.0 Handbook, Future Strategies Incorporated, 2010.
- [12] W. M. Van Der Aalst and K. M. Van Hee, Workflow Management: Models, Methods, and Systems, MIT press, 2004.
- [13] 速水治夫, 渋谷亮一, 鈴木登雄, 生駒順一, 寺下陽介, 植野直樹, 金子聡, 林潔, "ここまで来たワークフロー管理システム (3) ワークフロー製品の実際," *情報処理*, vol. 40, no. 5, pp. 507-513, 1999.
- [14] H. A. Reijers, I. Vanderfeesten and W. M. van der Aalst, "The Effectiveness of Workflow Management Systems: A Longitudinal Study," *International Journal of Information Management*, vol. 36, no. 1, pp. 126-141, 2016.
- [15] 電気学会ワークフロー調査専門委員会(編), ワークフローの実際, 日科技連出版, 1999.
- [16] A. E. Stohr and J. L. Zhao, "Workflow Automation: Overview and Research Issues," *Information Systems Frontiers*, vol. 3, no. 3, pp. 281-296, 2001.

- [17] G. Alonso, D. Agrawal, A. El Abbadi and C. Mohan, "Functionality and limitations of Current Workflow Management Systems," *IEEE expert*, vol. 12, no. 5, pp. 105-111, 1997.
- [18] S. Vinoski, "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," *IEEE Communications magazine*, vol. 35, no. 2, pp. 46-55, 1997.
- [19] S. Abiteboul, P. Buneman and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann, 2000.
- [20] 速水治夫, 勝間田仁, 世古将洋, 提箸公代, 渋谷亮一, 石丸知之, 大南正人, 岡田謙一, "商用ワークフロー管理システムと連動するインターワークフロー支援システム," *情報処理学会論文誌*, vol. 41, no. 10, pp. 2708-2718, 2000.
- [21] 森田昌宏, 向垣内岳弥, 山下武史, 速水治夫, "インターワークフロー支援: 組織間連携ワークフロープロセスの構築と分散型運用管理の支援機構," *情報処理学会論文誌*, vol. 38, no. 11, pp. 2298-2308, 1997.
- [22] A. Kumar and J.L. Zhao, "Workflow Support for Electronic Commerce Applications," *Decision Support Systems*, vol. 32, no. 3, pp. 265-278, 2002.
- [23] 速水治夫, 岡田謙一, "ワークフローからインターワークフローへ企業間電子商取引の基盤を目指して," *情報処理学会論文誌*, vol. 42, no. 3, pp. 432-447, 2001.
- [24] 小池賢一, 川口正高, "企業内ワークフローシステムのインターワークフローシステムへの拡張に関する検討," in *情報処理学会第66回全国大会*, 2004.
- [25] D. Hollingsworth, U. Hampshire, "Workflow Management Coalition: The Workflow Reference Model," *Document Number TC00-1003, 19(16), 224*, 1995.
- [26] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>.
- [27] D. Ciccio, A. Cecconi, J. Mendling, D. Felix, D. Haas, D. Lilek, F. Riel, A. Rumpl and P. Uhlig, "Blockchain-Based Traceability of Inter-organisational Business Processes," *Business Modeling and Software Design*, pp. 56-68, 2018.
- [28] 金子雄介, 田村浩気, 河合伸浩, 田中俊太郎, 岡知博, "貿易実務のブロックチェーン利用, 実践と課題," *情報処理学会デジタルプラクティス*, vol. 10, no. 3, pp. 492-505, 2019.
- [29] P. Helo and Y. Hao, "Blockchains in operations and supply chains : a model and reference implementation," *Computers and Industrial Engineering*, vol. 136, pp. 242-251, 2019.
- [30] I. G. M. T. Pradana, T. Djatna and I. Hermadi, "Blockchain Modeling for Traceability Information System in Supply Chain of Coffee Agroindustry," *2020*

International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 217-224, 2020.

- [31] M. Das, X. Tao, Y. Liu and J. C. P. Cheng, "A blockchain-based integrated document management framework for construction applications," *Automation in Construction*, vol. 133, p. Article 104001, 2022.
- [32] E. W. Kim, M. S. Park, K. Kim and K. J. Kim, "Blockchain-Based Automatic Tracking and Extracting Construction Document for Claim and Dispute Support," *KSCCE Journal of Civil Engineering*, vol. 9, no. 3707-3724, p. 26, 2022.
- [33] G. Hileman and M. Rauchs, "Global Blockchain Benchmarking Study," 2017. [Online]. Available: <https://ssrn.com/abstract=3040224>.
- [34] World Economic Forum, "Trade Tech - A New Age for Trade and Supply Chain Finance," 2018. [Online]. Available: http://www3.weforum.org/docs/White_Paper_Trade_Tech_report_2018.pdf.
- [35] World Trade Organization, "can Blockchain Revolutionize International Trade?," 2018. [Online]. Available: https://www.wto.org/English/res_e/booksp_e/blockchainrev18_e.pdf.
- [36] S. Namasudra, G. C. Deka, P. Johri, M. Hosseinpour and A. H. Gandomi, "The Revolution of Blockchain: State-of-the-Art and Research Challenges," *Archives of Computational Methods in Engineering*, vol. 28, pp. 1497-1515, 2021.
- [37] I. Abrar and J. A. Sheikh, "Current trends of blockchain technology: architecture, applications, challenges, and opportunities," *Discover Internet of Things*, vol. 4, no. 1, p. 7, 2024.
- [38] H. Michel and R. Garrick, "2017 Global Blockchain Benchmarking Study," 2017. [Online]. Available: <https://ssrn.com/abstract>.
- [39] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.," 2014. [Online]. Available: https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf.
- [40] I. S. Rao, M. L. M. Kiah, M. M. Hameed and Z. A. Memon, "Scalability of blockchain: a comprehensive review and future research direction," *Cluster Computing*, vol. 27, no. 5, pp. 5547-5570, 2024.
- [41] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han and F. Y. Wang, "Blockchain-enabled smart contracts: architecture, applications, and future trends," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266-2277, 2019.
- [42] Z. Wan, X. Xia, D. Lo, J. Chen, X. Luo and X. Yang, "Smart contract security: A

- practitioners' perspective," *Proceedings of the 2021 IEEE/ACM 43rd international conference on software engineering (ICSE)*, pp. 1410-1422, 2021.
- [43] N. Narula, W. Vasquez and M. Virza, "zkLedger: Privacy-Preserving auditing for distributed ledgers.," *Proceedings of the 15th USENIX symposium on networked systems design and implementation*, pp. 65-80, 2018.
- [44] E. Politou, F. Casino, E. Alepis and C. Patsakis, "Blockchain Mutability: Challenges and Proposed Solutions," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1972-1986, 2019.
- [45] A. Lee, "Single Point of Trust: Alternative Uses of the Blockchain," 2019. [Online]. Available: <https://www.cs.tufts.edu/comp/116/archive/fall2016/alee.pdf>.
- [46] Y. Cen, H. Wang and X. Li, "Improving Business Process Interoperability by Shared Ledgers," *Proceedings of the 6th International Conference on Informatics, Environment, Energy and Applications*, pp. 89-93, 2017.
- [47] P. Alves, R. Paskin, I. Frajho, Y. Miranda, J. Jardim, J. Cardoso and E. Tress, "Exploring Blockchain Technology to Improve Multi-party Relationship in Business Process Management Systems," *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020)*, vol. 2, pp. 817-825, 2020.
- [48] T. Sato, T. Shimosawa and Y. Himura, "OpsSC: Decentralized Blockchain Network Operation Workflow for Hyperledger Fabric," *Proceedings of the 2021 IEEE International Conference on Blockchain*, pp. 287-294, 2021.
- [49] J. Mendling, "Towards Blockchain Support for Business Processes," *Proceedings of the 8th International Symposium of Business Modeling and Software Design*, pp. 243-248, 2018.
- [50] O. López-Pintado, M. Dumas, L. García-Bañuelo and I. Weber, "Caterpillar: A Blockchain-Based Business Process Management System," 2017. [Online]. Available: https://www.researchgate.net/publication/329587092_Caterpillar_A_Blockchain-Based_Business_Process_Management_System.
- [51] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev and J. Mendling, "Using Blockchain to Enable Untrusted Business Process Monitoring and Execution," The University of New South Wales, Technical Report: UNSW-CSE-TR-201609, 2016.
- [52] G. Fridgen, S. Radszuwill, N. Urbach and L. Utz, "Cross-Organizational Workflow Management Using Blockchain Technology – Towards Applicability, Auditability, and Automation," *Proceedings of the 51st Hawaii International Conference on System Sciences*, pp. 1147-1156, 2018.

- [53] O. López-Pintado, L. García-Bañuelos, M. Dumas, I. Weber and O. López-Pintado, "Caterpillar: A Business Process Execution Engine on the Ethereum Blockchain," *Software: Practice and Experience*, vol. 49, no. 7, pp. 1162-1193, 2019.
- [54] I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev and J. Mendling, "Untrusted Business Process Monitoring and Execution Using Blockchain," *Business Process Management: 14th International Conference, BPM 2016*, vol. Proceedings 14, pp. 329-347, 2016.
- [55] B. Carminati, E. Ferrari and C. Rondanini, "Blockchain as a Platform for Secure Inter-Organizational Business Processes," . In *2018 IEEE 4th Proceedings of the International Conference on Collaboration and Internet Computing*, pp. 122-129, 2018.
- [56] A. Kosba, A. Miller, E. Shi, Z. Wen and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," *Proceedings of the 2016 IEEE symposium on security and privacy*, pp. 839-858, 2016.
- [57] T. Espel, L. Katz and G. Robin, "Proposal for Protocol on a Quorum Blockchain with Zero Knowledge," Cryptology ePrint Archive, 2017.
- [58] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson and D. Song, "Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts," *Proceedings of 2019 IEEE European Symposium on Security and Privacy*, pp. 185-200, 2019.
- [59] G. Zyskind, O. Nathan and A. Pentland, "Enigma: Decentralized Computation Platform with Guaranteed Privacy," *arXiv Preprint arXiv:1506.03471*, 2015.
- [60] Q. Zheng, Y. Li, P. Chen and X. Dong, "An Innovative IPFS-Based Storage Model for Blockchain," *Proceedings of 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 704-708, 2018.
- [61] A. Palai, M. Vora and A. Shah, "Empowering Light Nodes in Blockchains with Block Summarization," *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1-5, 2018.
- [62] H. Nagano, T. Shimosawa, A. Shimamura and N. Komoda, "Reliable Architecture of Cross Organizational Workflow Management System on Blockchain," *IADIS International Journal on Computer Science and Information Systems*, vol. 15, no. 2, pp. 29-43, 2021.
- [63] H. Nagano, T. Shimosawa, A. Shimamura and N. Komoda, "Blockchain based Cross Organizational Workflow Management System," *Proceedings of the 17th International Conference on Applied Computing 2020*, pp. 97-104, 2020.

- [64] 長野裕史, 下沢拓, 島村敦司, 薦田憲久, "ブロックチェーンを活用した企業間ワークフロー管理システムの実行エンジン," *情報処理学会論文誌デジタルプラクティス (TDP)*, vol. 3, no. 2, pp. 1-11, 2022.
- [65] 長野裕史, 下沢拓, 島村敦司, 薦田憲久, "ブロックチェーンを活用した企業間ワークフローの実装," *電気学会 第83回情報システム研究会, IS-20-053*, pp. 91-95, 2020.
- [66] H. Nagano, K. Fukuchi, N. Komoda and T. Ohkawa, "Cross Organizational Workflow Management System with Controlled Disclosure and No-Redundant Data on Blockchain," *Proceedings of the 21st International Conference on Applied Computing 2024*, pp. 13-21, 2024.
- [67] 長野裕史, 福地開帆, 大川剛直, 薦田憲久, "大量データを処理する企業間ビジネスプロセスの高信頼な処理アーキテクチャ," *電気学会 第94回情報システム研究会, IS-23-043*, pp. 43-47, 2023.
- [68] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5-51, 2003.

研究業績

本研究に関する発表論文

A. 学術論文誌論文

1. H. Nagano, T. Shimosawa, A. Shimamura, and N. Komoda, “Reliable Architecture of Cross Organizational Workflow Management System on Blockchain,” *IADIS International Journal on Computer Science and Information Systems*, vol. 15, no. 2, pp. 29-43, 2021.
2. 長野裕史, 下沢拓, 島村敦司, 薦田憲久, “ブロックチェーンを活用した企業間ワークフロー管理システムの実行エンジン,” *情報処理学会論文誌デジタルプラクティス (TDP)*, vol. 3, no. 2, pp. 1-11, 2022.

B. 国際会議

1. H. Nagano, T. Shimosawa, A. Shimamura and N. Komoda. “Blockchain based Cross Organizational Workflow Management System,” *Proceedings of the 17th International Conference on Applied Computing 2020*, pp. 97-104, 2020.
2. H. Nagano, K. Fukuchi, N. Komoda and T. Ohkawa, “Cross Organizational Workflow Management System with Controlled Disclosure and No-Redundant Data on Blockchain,” *Proceedings of the 21st International Conference on Applied Computing 2024*, pp. 13-21, 2024.

C. 学会講演

1. 長野裕史, 下沢拓, 島村敦司, 薦田憲久, “ブロックチェーンを活用した企業間ワークフローの実装,” *電気学会 第83回情報システム研究会*, IS-20-053, pp. 91-95, 2020.

2. 長野裕史, 福地開帆, 大川剛直, 薦田憲久, "大量データを処理する企業間ビジネスプロセスの高信頼な処理アーキテクチャ," *電気学会 第 94 回情報システム研究会*, IS-23-043, pp. 43-47, 2023.

その他の発表論文

C. 学会講演

1. 長野裕史, 江口俊宏, 浜田ちぐさ, 佐々木敏郎, "IC カードを用いた旅客情報サービスシステムの構想: 座席確定を猶予する列車予約方式の評価," *情報処理学会第 59 回全国大会講演論文集 コンピュータと人間社会*, 5W-8, pp. 4-205 - 4-206, 1999.
2. 浜田ちぐさ, 江口俊宏, 長野裕史, 佐々木敏郎, "IC カードを用いた旅客情報サービスシステムの構想: 個人の行動履歴を利用した旅行計画支援システムの提案," *情報処理学会第 59 回全国大会講演論文集 コンピュータと人間社会*, 5W-9, pp. 4-207 - 4-208, 1999.

D. その他

1. 長野裕史, "Voice from the Business Frontier: 日立がめざす金融× デジタルソリューション (特集 日本の銀行業におけるデジタルトランスフォーメーションの未来)," *日立総研*, vol.15, no.1, pp. 42-45, 2020.
2. 長野裕史, 原有希, 大島訓, 西田一平, 長稔也, "ブロックチェーンを活用した金融サービス・ビジネスモデルの創出 (特集 デジタルが導く金融イノベーション: FinTech & Beyond)---(先端デジタル技術が導く金融イノベーションの姿)," *日立評論*, vol.98, no.9, pp. 558-562, 2016.
3. 原有希, 酒田大樹, 長野裕史, 赤司卓也, "顧客協創を通じた金融ソリューションコンセプトの創出 (特集 デジタルが導く金融イノベーション: FinTech & Beyond)---(顧客協創とデジタル活用による金融ビジネスの効率化と創出)," *日立評論*, vol.98, no.9, pp. 575-578, 2016.

本論文の構成

本論文の第 2 章は，以下の論文からなる。

- H. Nagano, T. Shimosawa, A. Shimamura and N. Komoda, "Blockchain based Cross Organizational Workflow Management System," *Proceedings of the 17th International Conference on Applied Computing 2020*, pp. 97-104, 2020.
- H. Nagano, T. Shimosawa, A. Shimamura and N. Komoda, "Reliable Architecture of Cross Organizational Workflow Management System on Blockchain," *IADIS International Journal on Computer Science and Information Systems*, vol. 15, no. 2, pp. 29-43, 2021.

本論文の第 3 章は，以下の論文からなる。

- 長野裕史，下沢拓，島村敦司，薦田憲久，"ブロックチェーンを活用した企業間ワークフロー管理システムの実行エンジン," *情報処理学会論文誌デジタルプラクティス (TDP)*, vol. 3, no. 2, pp. 1-11, 2022.
- 長野裕史，下沢拓，島村敦司，薦田憲久，"ブロックチェーンを活用した企業間ワークフローの実装," *電気学会 第 83 回情報システム研究会*, IS-20-053, pp. 91-95, 2020.

本論文の第 4 章は，以下の論文からなる。

- H. Nagano, K. Fukuchi, N. Komoda and T. Ohkawa, "Cross Organizational Workflow Management System with Controlled Disclosure and No-Redundant Data on Blockchain," *Proceedings of the 21st International Conference on Applied Computing 2024*, pp. 13-21, 2024.
- 長野裕史，福地開帆，大川剛直，薦田憲久，"大量データを処理する企業間ビジネスプロセスの高信頼な処理アーキテクチャ," *電気学会 第 94 回情報システム研究会*, IS-23-043, pp. 43-47, 2023.

謝 辞

本研究の全過程を通じて、終始懇切丁寧なるご指導とご鞭撻、格別のご配慮を賜りました神戸大学大学院システム情報学研究科システム情報学専攻 大川剛直 教授，ならびに，大阪大学大学院情報科学研究科マルチメディア工学専攻 薦田憲久 招へい教授（大阪大学名誉教授）に深く感謝申し上げます。

本研究をまとめるにあたり，貴重なお時間を割いて頂き，丁寧なるご教示を賜りました神戸大学大学院システム情報学研究科システム情報学専攻 鳩野逸生 教授，太田能教授に深く感謝申し上げます。

本研究の機会と大学院博士課程後期課程に進学する機会を与えて頂くとともに，研究を進めるにあたりご配慮を賜りました（株）日立製作所 戦略 SIB ビジネスユニット Strategy Office 副長 加藤博光 博士，研究開発グループ Digital Innovation R&D システムイノベーションセンタ センタ長 谷繁幸 博士，Digital Innovation R&D Managing Director 付 佐藤暁子氏に心より御礼申し上げます。また，筆者の大学院博士課程後期課程在学中に多大なるご配慮とご支援を頂いた（株）日立製作所 研究開発グループ Digital Innovation R&D システムイノベーションセンタ デジタルエコノミー&コミュニティ研究部の関係各位に厚く御礼申し上げます。

第2章，第3章の研究に関して，共同研究者として様々なご討論やご助言を頂きました（株）日立製作所 研究開発グループ Digital Innovation R&D システムイノベーションセンタ デジタルエコノミー&コミュニティ研究部 リーダ主任研究員 下沢拓博士，主任研究員 島村敦司氏に厚く御礼申し上げます。また第4章の研究に関して，共同研究者として様々なご討論や実験環境構築にご支援をいただいた，（株）日立製作所 研究開発グループ Digital Innovation R&D システムイノベーションセンタ デジタルエコノミー&コミュニティ研究部 研究員 福地開帆氏に深く感謝申し上げます。

最後に，学位取得に向け応援してくれた家族に感謝致します。

神戸大学博士論文

「ブロックチェーンを用いた組織間ワークフロー管理システムに関する研究」 全 90
頁

提 出 日 2025 年 7 月 17 日

本博士論文が神戸大学機関リポジトリ **Kernel** にて掲載される場合、
掲載登録日（公開日）はリポジトリの該当ページ上に掲載されます。

© 長野 裕史

本論文の内容の一部あるいは全部を無断で複製・転載・翻訳することを禁じます。