情報処理学会論文誌　デジタルプラクティス　Vol.4　No.3　33–47 (Oct. 2023)

ネットワーク構成モデルに基づく ネットワーク機器設定手順自動生成システム

新井 凪[1,a)]　小形 真平[1,b)]　鈴木 彦文[1,c)]　岡野 浩三[1,d)]

受付日 2022年11月22日, 採録日 2023年4月21日

概要：情報ネットワークを構築・構成変更する準備として，ネットワークエンジニアはその仕様に基づきネットワーク機器の設定手順を作成する．しかし，エンジニアは通常，設定手順を手作業で作成するため，設定手順は仕様と乖離しやすい．そこで本稿ではその改善のため，ネットワーク構成やその変更の仕様に基づき，ネットワーク機器の設定コマンドからなる設定手順を自動生成する手法を提案する．本研究では，仕様の厳密性と拡張性を両立しやすいように，オブジェクト指向モデリング言語 UML（Unified Modeling Language）を応用してネットワーク構成をモデル化する記法を実現した．さらに，仕様と整合する設定手順を得られるように，ネットワーク構成モデルに基づき設定手順を自動生成する方法を実現した．評価実験として，広域型キャンパスである信州大学のネットワーク構成を例に，スタティックルーティングによって構成されたネットワークから OSPF（Open Shortest Path First）プロトコルによってダイナミックルーティングで構成されたネットワークに構成変更をした場合を適用事例とし，提案手法を適用した結果，すべての設定手順が期待どおりに得られ，期待どおりの振る舞いをするネットワークが構築されたことを確認した．

キーワード：ネットワーク管理，ネットワーク構成，ネットワーク設計，モデリング，自動生成

A System to Automatically Generate Configuration Instructions for Network Elements from Network Configuration Models

Nagi Arai[1,a)]　Shinpei Ogata[1,b)]　Hikofumi Suzuki[1,c)]　Kozo Okano[1,d)]

Received: November 22, 2022, Accepted: April 21, 2023

Abstract: In preparation for constructing or changing an information network, network engineers create configuration instructions for network elements based on the specifications of the network configuration and its changes. However, engineers usually create configuration instructions manually, and thus configuration instructions often deviate from the specifications. To improve this situation, this paper proposes a method to automatically generate configuration instructions consisting of configuration commands for network elements based on the specification of the network configuration and its changes. In this study, we applied the object-oriented modeling language UML (Unified Modeling Language) to realize a notation for modeling network configurations so that the specification should be both rigorous and easy to extend. Furthermore, to obtain configuration instructions consistent with the specifications, we have implemented a method to automatically generate configuration instructions based on network configuration models. As an evaluation experiment, we applied the proposed method to a configuration change case in which the configuration of multiple network elements is changed, and all configuration instructions were obtained as expected.

Keywords: network management, network configuration, network design, modeling, automatic generation

1. はじめに

情報ネットワークを構築・構成変更する準備として，ネットワークエンジニアはその仕様を吟味した後に，ネットワーク機器を設定するための作業手順書を作成する [1], [2]．作業手順書とは，エンジニアの能力によらないよう

[1] 信州大学
Shinshu University, Nagano 380–8553, Japan
[a)] 21w2002h@shinshu-u.ac.jp
[b)] ogata@cs.shinshu-u.ac.jp
[c)] h-suzuki@shinshu-u.ac.jp
[d)] okano@cs.shinshu-u.ac.jp

設定作業の均質化を図り，また設定内容を後から追跡できるように，ネットワーク機器設定コマンドからなる設定手順（以下，機器設定手順）等を明記した手順書である．エンジニアは通常，ネットワーク構成やその変更の仕様を基に手作業で作業手順書を作成するため，作業手順書が実際の仕様と乖離しやすい問題がある．

その改善に直接的に寄与する従来研究として，ネットワーク構成やその変更の仕様に基づき作業手順書を自動生成するアプローチの研究がある [3], [4]．本アプローチは，仕様と整合する作業手順書を得るために有効である．その一方で，複雑かつ多様な機器設定手順を表す作業手順書を正確に自動生成するには，生成元の仕様に対する厳密性や表現力の向上が課題とされる．仕様の厳密性を高める従来研究としては，図記号を取り入れた準形式的な仕様記述であるモデルを活用するアプローチがある [5]–[10]．しかし，これらの手法は，現在可用な仕様のみを網羅するものであり，拡張性を考慮した仕様の表現方法については言及されていない．

その他，Ansible や NETCONF を用いたネットワーク構成自動管理についても挙げられる．特に，Ansible に関しては，ネットワークや機器の構成の状態を yaml によってドキュメント化し，仕様をまとめている．また，この仕様から Playbook を用いたネットワーク作業手順書などのドキュメント生成も可能であると考えられる．しかし，Ansible によるネットワークの構成変更は，すでにネットワーク機器に関する文法構造を理解していることが前提であり，どのようなエンジニアでも様々なネットワークの変更に容易に対応することは難しいと考える．またネットワーク構成の管理のたびに，コマンドを考慮した仕様を記述することはエンジニアの負担につながると考える．

さて，仕様の拡張性を高めるとしても，現在利用可能な通信プロトコル等を網羅するのでは一過的である．我々は，通信プロトコル等の進化にエンジニアが維持・管理しやすいように仕様記法の拡張性を高めることが肝要と考える．しかしながら，仕様の厳密性と拡張性の両立や，そのような仕様に基づいて作業手順書を自動生成する観点では未だ確立された方法はない．

そこで本研究では，仕様の厳密性と拡張性の両立を指向するため，オブジェクト指向モデリング言語 UML（Unified Modeling Language）を応用してネットワーク構成をモデル化する記法を新たに実現した．この記法によるモデルは，ネットワーク機器の設定コマンドにおけるパラメタ値レベルの詳細な情報を提供できる．UML とは，グラフィカルな記述言語の一種で，単一のメタモデルで構成されており，オブジェクト指向スタイルを使って構築するシステムの記述や設計に用いられるものである [11], [12]．

一般的に，情報ネットワークを構築・構成変更する際には，様々なエンジニアからなる設計者によりネットワーク構成を設計するが，その役割は，ネットワークニンジニアがネットワークを構築・構成変更などするための詳細な設計を行うことである．この設計とは，ネットワークエンジニアとしては，悩むことなくきわめて直接的に従うことができる指示になっていなければならない．そこで UML に着目し，複雑なネットワーク構成の概念や関係を視覚的に表現させ，そしてクラスやオブジェクトなどの詳細な記述により，より具体的かつ細分化を図ることで，仕様の拡張性や厳密性を高めることを期待し，ネットワーク構成をモデル化する記法を提案している．また本研究ではさらに，2 つのネットワーク構成モデル間の差分に基づき，ネットワーク機器ごとに機器設定手順を自動生成するシステムも新たに実現した．この方法では，ネットワーク構成モデル間差分の詳細な情報に対してネットワーク機器の設定コマンド列を正確に自動生成でき，かつ生成される設定コマンドをエンジニアが拡張変更しやすいように，モデルと設定コマンドの対応関係記述（以下，機器設定コマンドテンプレート）の記法も新たに実現した．これにより，ネットワークシステムを設計すると同時に，ネットワーク機器に発行するべきコマンドを自動で生成することが可能になるため，エンジニアの負担軽減が見込める．

本研究の評価実験として，OSPF プロトコルを用いた，実運用なネットワークにおいて複数のネットワーク機器の設定に変更が生じる構成変更例に提案手法を適用した結果，すべての設定手順が期待どおりに得られ，期待どおりの振る舞いをするネットワークが構築されたことを確認した．

本稿の構成は次のとおりである．2 章で提案手法について説明し，3 章では生成された機器設定手順について説明する．また，4 章で評価について述べ，提案手法の正当性を確認するための評価とその結果についてまとめる．最後に，5 章でまとめと今後の課題を述べる．

2. 提案手法

提案手法を 2 つのパートで説明する．1 つ目はネットワーク構成の文法構造を規定するネットワーク構成メタモデルと，ネットワーク構成のインスタンスを表すネットワーク構成モデルについてである．2 つ目は 2 つのネットワーク構成モデル間の差分と機器設定コマンドテンプレートに基づく機器設定手順の自動生成方法についてである．

2.1 ネットワーク構成のメタモデルとモデル

ネットワーク構成情報とは，結線等のネットワーク機器の構成や，各機器への設定内容をまとめたものである．実運用なネットワークであるほど，その構築や構成変更の失敗を防ぐためにはネットワーク構成情報を明確に記述し，エンジニア間で吟味することが重要となる．設計段階でネットワーク構成情報を明確化するには，ネットワーク機

器に対する設定コマンドのパラメタ値といった構築・構成変更時に必須なデータをそのままの粒度で表すことが重要であり，かつ，それらデータを整理しやすい記法も重要である．

従来の典型的なネットワーク構成図 [2] では，ネットワーク機器などのネットワーク要素のアイコンを線で繋ぐことでネットワーク構成を分かりやすく図示するものであるが，各機器の設定を記述する空間は取りにくく曖昧になりやすい．加えて，各機器の設定詳細を任意で記述するにしても自由記述にならざるを得ないため，書き手の能力に大きく依存して質が保証しにくい．

情報の充実化と記述の形式化を図った方法として，吉澤ら [4] のシステムでは，ネットワーク構成図を図示でき，かつ各ネットワーク機器の設定詳細を入力できるダイアログも表示できる．そのため，典型的なネットワーク構成図に比べてネットワーク構成を厳密かつ詳細に管理できる．しかし，ネットワーク機器の設定項目を拡張変更する場合にはソフトウェアの改変が必要であり，そのような改変作業は一般に高コストである．本システムのデータ構造を明文化した仕様を活用できるならば，設定項目の拡張容易性を高めることに向くが，仕様記述に基づく作業手順書生成のアプローチは示されていない．

そこで本研究では，ネットワーク構成やその変更の仕様について厳密性と拡張性の両立を指向するため，オブジェクト指向モデリング言語 UML を応用したネットワーク構成のモデル化記法を実現した．提案記法の主な登場概念として，ネットワーク構成の仕様項目を構造的に規定するネットワーク構成メタモデル（以下，単にメタモデルとも呼称）と，具体的なネットワーク構成を表すネットワーク構成モデル（以下，単にモデルとも呼称）がある．たとえば，メタモデルでは“port”等の仕様項目を規定し，モデルでは“port”に対して“2”等の値を表現する．以下の項で，メタモデルとモデルについて詳解する．

2.1.1　ネットワーク構成メタモデル

図 1 に，本稿が係わる部分のメタモデルを示す．メタモデルは UML クラス図の記法をベースに表される．主な構成概念を図中の番号（1）〜（6）に対応させ，次にまとめる．

(1) `ipAddress` や `port` 等の互いに関連が強い仕様項目をまとめた**仕様項目グループ**（例：`EthernetSetting` をラベルとする大きな四角形）がある．ここで `EthernetSetting` はグループ名である．

(2) 仕様化すべき項目を表す**仕様項目**（例：`port : int`）がある．ここで `port` は項目名であり，`int` は型である．

(3) 仕様項目グループ間の関係性を規定する**関係**（例：`OspfSetting` と `Config` の間の線）がある．

(4) 関係の端には多重度（例：*）と呼ぶ記述があり，* は“0 個以上”を意味する．たとえば `OspfSetting` が 1 個存在したときに関係する `OspfVirtualLink` が 0 個以上存在することを表す．なお，`0..1` は“0 個または 1 個”を意味する．

(5) 白三角付きの関係は**汎化**と呼び，三角側の性質を無印側が引き継ぐことを意味する．この意味論により，た

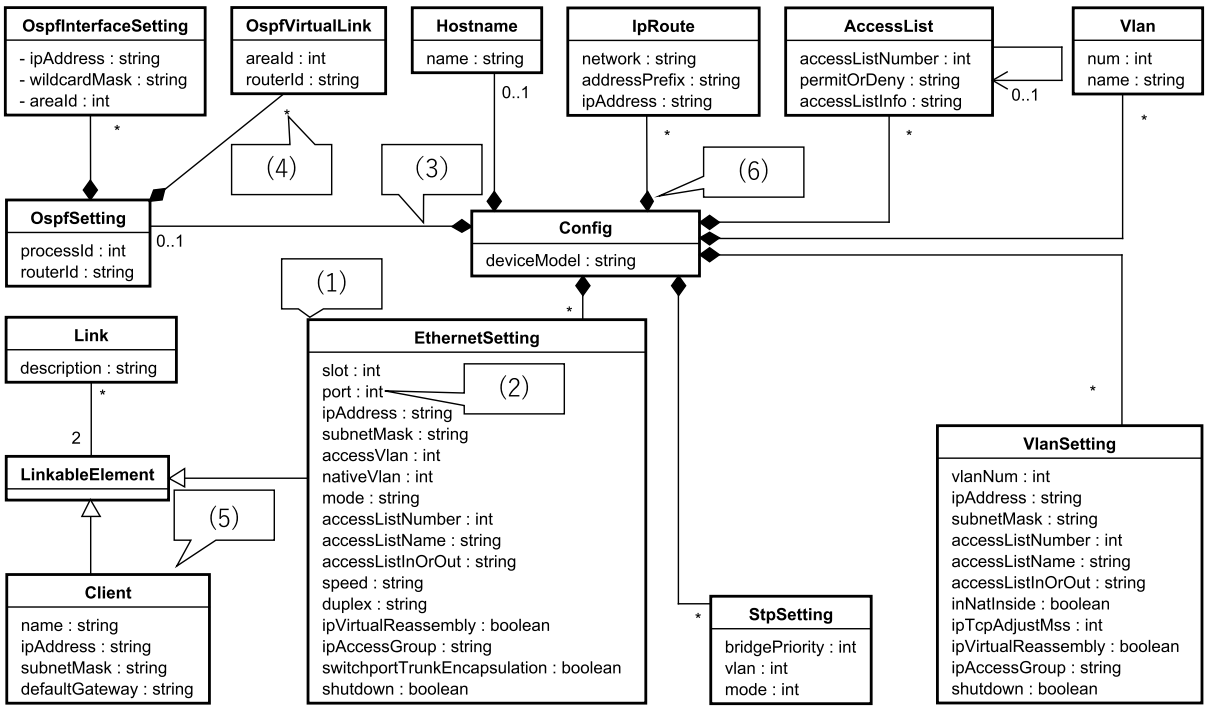

図 1　ネットワーク構成メタモデル

Fig. 1　Network configuration metamodel

とえば Client は Link と関係があるとみなせる．

(6) 黒菱形付きの関係は**コンポジション**と呼び，菱形側が無印側を専有することを意味する．この意味論により，たとえば 1 個の OspfSetting が，ある複数個の OspfInterfaceSetting を共有するような関係は持てないとみなせる．

したがって，本研究におけるメタモデルは，物理設計観点の仕様項目群（例：Link や LinkableElement）と，論理設計観点の仕様項目群（例：EthernetSetting や OspfSetting）を組み合わせて定義されている．このようにメタモデルでは異なる観点の仕様項目の構造を 1 つの記法で一貫的に整理できる．

また，本稿における論理設計観点の仕様項目のほとんどは Cisco 製ネットワーク機器のコマンド体系を基に定義されている．そのため，図 1 に示される仕様項目は多くないが，当該体系に一貫した拡張は容易である．一方で，本稿のメタモデルではマルチベンダ対応を指向していないが，Cisco 製機器のコマンド体系と整合性が高い他ベンダの機器については対応は比較的容易と考えられる．

2.1.2　ネットワーク構成モデル

図 2 にモデルを例示する．モデルは UML オブジェクト図の記法をベースに表されている．モデルは図 1 のメタモデルに準拠している．この準拠とは次の (1)〜(3) のことを意味する．また，図中に登場する (1)〜(3) と対応している．

(1) メタモデルの**仕様項目グループ**を型とした値（以下，**グループ値**；例：Cf1 : Config をラベルとする大きな四角形）を作成している．ここで Cf1 : Config を**識別子**と呼び，Cf1 をグループ値名と呼ぶ．また，Config は仕様項目グループ名である．

(2) グループ値ごとに**仕様項目**に値（以下，**項目値**；例：name = campus1 における campus1）を与えている．

(3) 仕様項目グループ間の**関係**を満たすようにグループ値間の線（以下，**関係値**）を記述している．

なお，項目値が空値の仕様項目は，たとえば name = campus1 の = 以降の記述が存在しない．ここで，等号（=）は UML オブジェクト図の意味論に基づき，左辺の仕様項目に右辺の項目値が与えられた状態を示す．

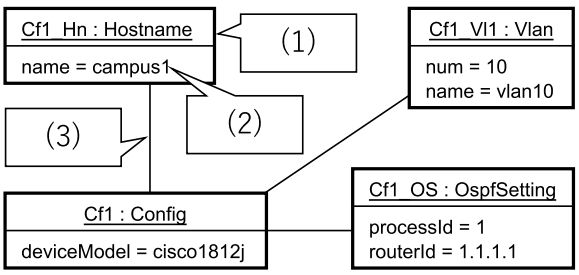

図 2　ネットワーク構成モデル
Fig. 2　Network configuration model

2.2　機器設定手順の生成手法

図 3 に，機器設定手順の生成手法の概要を示す．本手法では，2 つのモデル間の差分に基づき機器設定手順を生成する．この 2 つのモデルは，変更が必要な現状のネットワーク構成を表すモデル（AsIs モデル）と，変更後にあるべきネットワーク構成を表すモデル（ToBe モデル）の関係にあることを前提とする．以下は，本手法のフローである．

1. **手法利用者の入力**　本手法では，後述する AsIs モデル（図 6）と ToBe モデル（図 7），機器設定コマンドテンプレート（図 8）の 3 つを手法利用者の入力とし，機器設定手順を最終出力とする．ここでの機器設定手順とは，AsIs を ToBe に変えるために必要となる，ネットワーク機器ごとに得られる機器設定コマンド列の集合である．モデル間差分に基づき生成する理由の 1 つとして，適用後のネットワークに不具合が生じた場合，切り戻しを容易に行うことができる利点がある．単に，AsIs モデルと ToBe モデルから機器設定手順を生成することも可能ではあるが，元のネットワーク構成からの差分を参照できないため，切り戻しが難しいと考えられる．加えて，ACL（Access Control List）が適用されているインタフェースの変更やリモートからの作業を考慮すると，必要箇所だけの機器設定手順

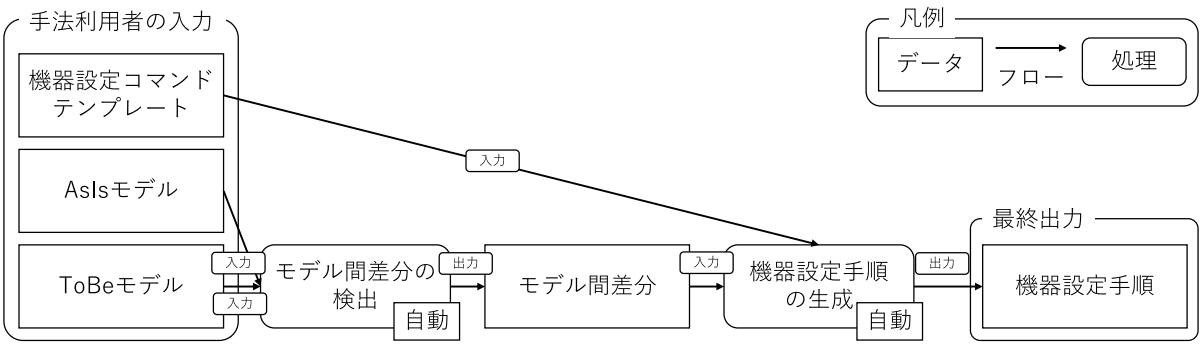

図 3　機器設定手順の生成手法の概要
Fig. 3　Overview of the proposed method to generate configuration commands for network elements

を生成することが望ましい．

2. **モデル間差分の検出**　本ステップでは，AsIs と ToBe のモデル間差分を検出する．モデル間差分とは端的に，設定解除が必要な設定内容と，新規設定が必要な設定内容を併合した情報である．モデル間差分を得ることで，AsIs から ToBe に変えるために必要な機器設定コマンドを特定できる．モデル間差分の検出手順については，2.2.1 項で詳述する．
3. **機器設定手順の生成**　本ステップでは，モデル間差分に基づいて機器設定コマンドテンプレートを具体化することで，機器設定手順を生成し，最終出力とする．機器設定手順の生成手順については 2.2.2 項で詳述する．

本手法の説明事例として，**図 4** のネットワーク構成を**図 5** に変更するケースを扱う．また，紙面の都合上，AsIs モデルと ToBe モデルに関しては一部のみの掲載とする．本例は，信州大学のネットワーク [13] を模倣したものを取り上げており，スタティックルーティングで構成されたネットワークから，OSPF プロトコルを用いたダイナミックルーティングで構成されるネットワークの設定を行うものである．図 4，図 5 のそれぞれに対応するモデルを**図 6**，**図 7** に示す．

2.2.1　モデル間差分の検出

モデル間差分とは，AsIs と ToBe のモデル間における項目値の差異を指す．項目値に差異があったとき，AsIs モデルの項目値はネットワーク機器から解除が必要であり，ToBe モデルの項目値はネットワーク機器への新規設定が必要となる．本ステップの目的は，設定解除すべき項目値に `unset` ラベルを与え，新規設定すべき項目値に `set` ラベルを与えることである．

次にモデル間差分の検出アルゴリズムを説明する．前提として空値（使用しなかった項目）の項目値は必ずラベル無しとする．そのため，次から説明する手順でのラベリング処理は，空値でない項目値に対してのみ行う．まず，AsIs と ToBe のモデル間で識別子（例：`Cf1_Hn` や `Cf1_Vl1`）が同じグループ値をペアとする．グループ値のペアが成立した場合，ペア内の同名な仕様項目間で項目値に差異があれば，AsIs 側の項目値に `unset` を与え，ToBe 側の項目値に `set` を与える．もし AsIs と ToBe でモデルが異なるなどして同名の仕様項目がない場合，AsIs 側のみにある仕

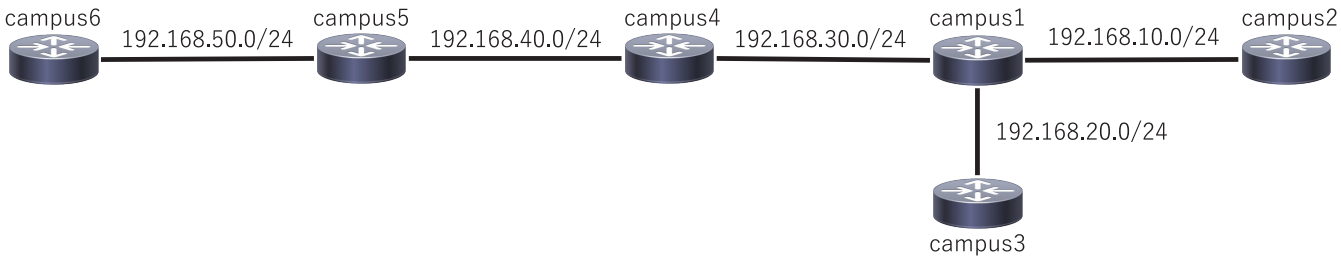

図 4　AsIs ネットワーク構成図
Fig. 4　AsIs network diagram

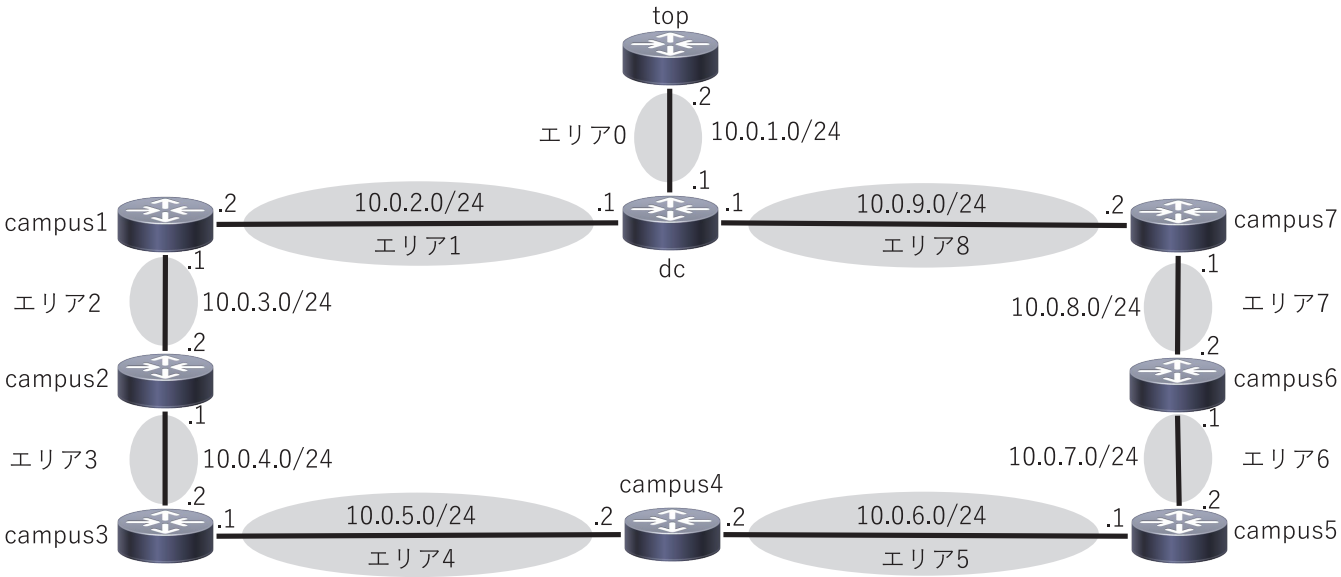

図 5　ToBe ネットワーク構成図
Fig. 5　ToBe network diagram

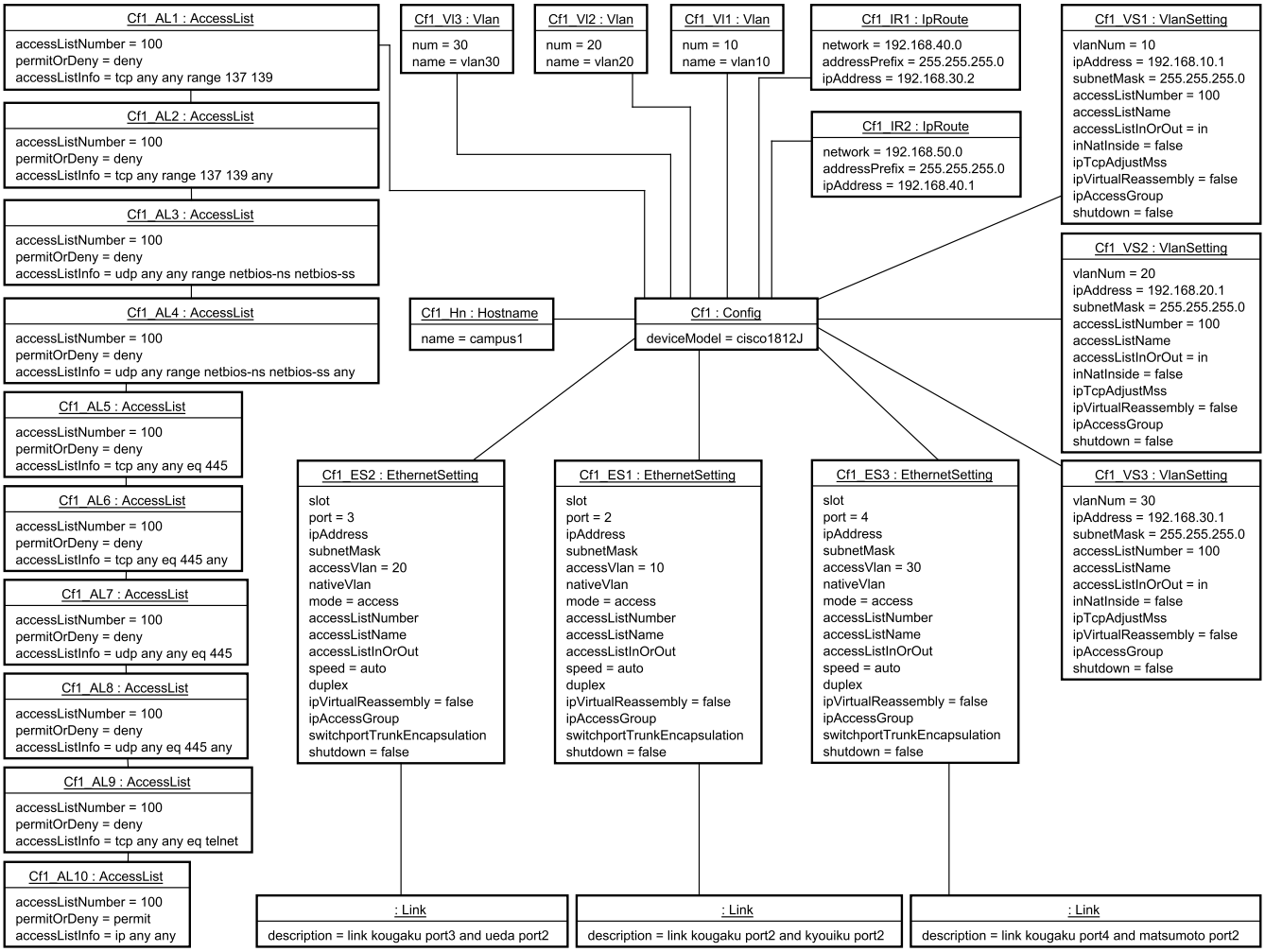

図 6　AsIs ネットワーク構成モデル（図 4 `campus1` に該当）
Fig. 6　AsIs network configuration model (campus1 in Fig. 4)

様項目の項目値に `unset` を与え，ToBe 側のみにある仕様項目の項目値に `set` を与える．一方，グループ値のペアが成立しなかった場合，そのグループ値が有するすべての項目値に対し，項目値が AsIs 側にあれば `unset` を与え，ToBe 側にあれば `set` を与える．このようにして，新規設定または更新する場合と設定解除をする場合にラベルを付与することで，生成するべきコマンドの情報を得ることができる．また，厳密な識別子の書き方については特に定めていない．

図 6 と図 7 に示されるように，ネットワーク機器で設定不要な項目値（例：結線情報を表す `Link` 中の `description` の項目値）にラベルが与えられている．本手法で登場する `Link` や `Client` とは，ネットワーク機器やクライアントとの接続を表す仕様項目グループであり，機器設定手順を生成する際のコマンドは登場しない．そのため，変更前後においてラベルが付与されていたとしても無視するものとする．このような項目値は，機器設定手順の生成アルゴリズムや機器設定コマンドテンプレートにより最終出力から除外するものとする．

2.2.2　機器設定手順の生成

前ステップで得られたモデル間差分をネットワーク機器の設定に反映するには，各項目値のラベルに応じた機器設定コマンドが必要となる．

図 8 に機器設定コマンドテンプレートを例示し，**表 1** で構成要素を説明する．ネットワーク機種が異なる場合，設定コマンドの体系も異なることがある．これを踏まえて，ネットワーク機種ごとに機器設定コマンドテンプレートを作成できるように，`Config` の `deviceModel` の値と，（図 8 には表れないが）機器設定コマンドテンプレートの名前を同名にして対応づける．また，図 8 の `Spec. Item Group` に記載がない仕様項目グループ（例：`Link` や `Client`）は，機器設定手順の生成には利用しない．

その他，ACL（`AccessList`）など，順序の制約が強いものに関しては機器設定コマンドテンプレート等を用いた順序の制約は与えず，モデル内で登場する関連に制約を与える．たとえば，図 1 で登場する `AccessList` では，0 から 1 の多重度を持つ自己参照を行うことで，`AccessList`

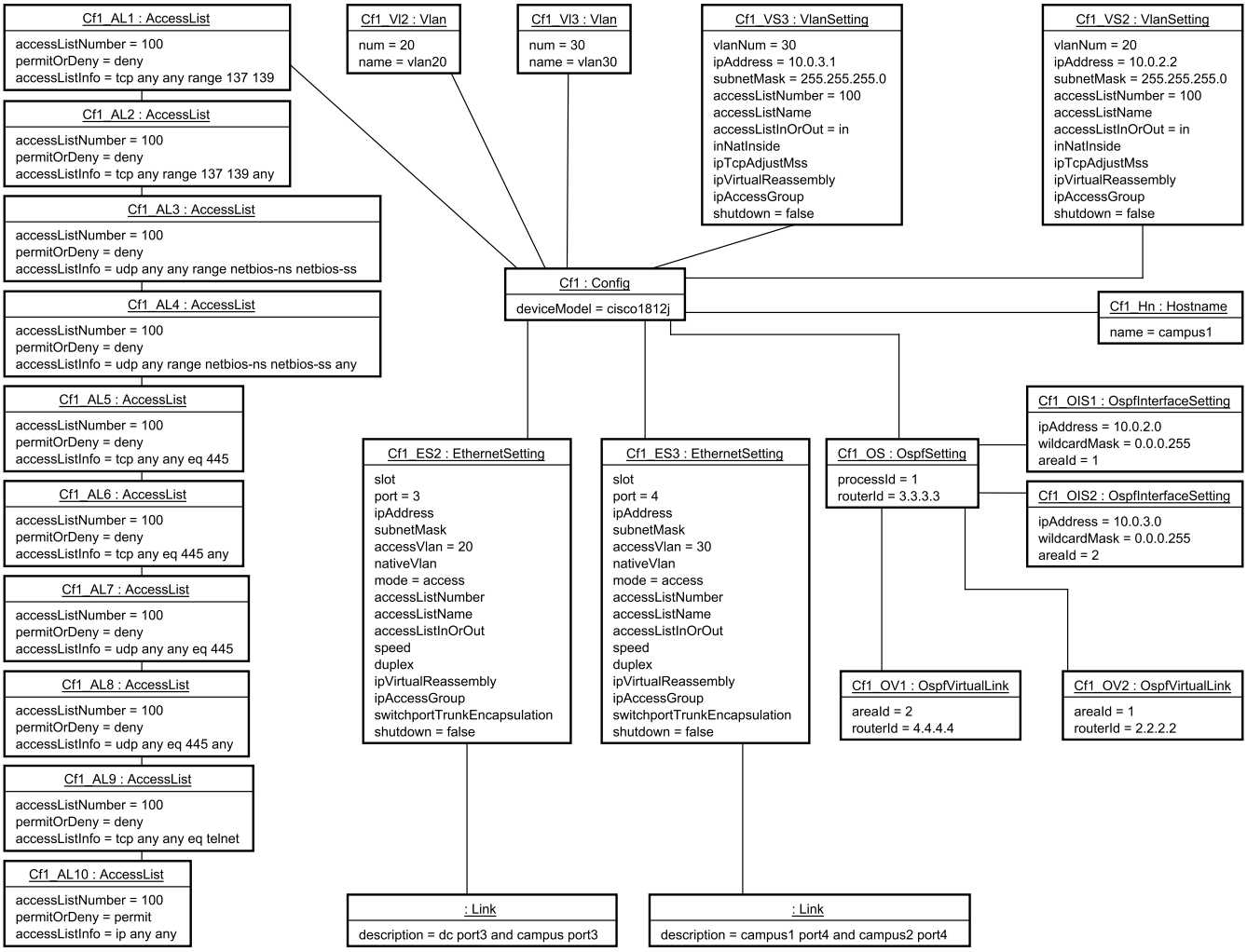

図 7　Tobe ネットワーク構成モデル（図 5 `campus1` に該当）

Fig. 7　ToBe network configuration model (campus1 in Fig. 5)

Cmd. Type	Spec. Item Group	Spec. Item	Proc. Type	ID	Command	Modal	Dep. ID	Condition
header				1	enable			
header				2	configure terminal			
template	Hostname	name	set	3	hostname <name>		2	
template	Hostname	name	unset	4	no hostname		2	
template	AccessList	accessListNumber/permitOrDeny/accessListInfo	set	5	access-list <accessListNumber> <permitOrDeny> <accessListInfo>		2	
template	Vlan	num	set	6	vlan <num>	TRUE	2	
template	Vlan	name	set	7	name <name>		6	
template	Vlan	num	unset	8	no vlan <num>		2	
template	VlanSetting	vlanNum	set	9	interface vlan <vlanNum>	TRUE	2	
template	VlanSetting	ipAddress/subnetMask	set	10	ip address <ipAddress> <subnetMask>		9	
template	VlanSetting	accessListNumber/accessListInOrOut	set	11	ip access-group <accessListNumber> <accessListInOrOut>		9	
template	VlanSetting	vlanNum	unset	12	default interface vlan <vlanNum>		2	
template	VlanSetting	vlanNum	unset	13	no interface vlan <vlanNum>		12	
template	IpRoute	network/addressPrefix/ipAddress	set	14	ip route <network> <addressPrefix> <ipAddress>		2	
template	IpRoute	network/addressPrefix/ipAddress	unset	15	no ip route <network> <addressPrefix> <ipAddress>		2	
template	EthernetSetting	*	set/unset	16	interface fastethernet <port>	TRUE	2	
template	EthernetSetting	ipAddress/subnetMask	set	17	ip address <ipAddress> <subnetMask>		16	
template	EthernetSetting	port/speed	set	18	speed <speed>		16	
template	EthernetSetting	port/duplex	set	19	duplex <duplex>		16	
template	EthernetSetting	port/shutdown	set	20	no shutdown		16	<shutdown> == false
template	EthernetSetting	port/shutdown	set	21	shutdown		16	<shutdown> == true
template	EthernetSetting	ipAddress	unset	22	no ip address <ipAddress>		16	
template	EthernetSetting	mode	set	23	switchport mode <mode>		16	
template	EthernetSetting	accessVlan	set	24	switchport access vlan <accessVlan>		16	
template	EthernetSetting	port	unset	25	default interface fastethernet <port>		2	
template	OspfSetting	processId	set	26	router ospf <processId>	TRUE	2	
template	OspfSetting	routerId	set	27	router-id <routerId>		26	
template	OspfInterfaceSetting	ipAddress/wildcardMask/areaId	set	28	network <ipAddress> <wildcardMask> area <areaId>		26	
template	OspfVirtualLink	areaId/routerId	set	29	area <areaId> virtual-link <routerId>		26	
mode_after				30	exit			
footer				31	exit			
footer				32	copy running-config startup-config			

図 8　機器設定コマンドテンプレート（`Cisco 1812-J`, `Cisco 892` に対応）

Fig. 8　Template of configuration commands for network elements (based on Cisco 1812-J and Cisco 892)

表 1　機器設定コマンドテンプレートの構成要素
Table 1　Notation elements of the template of configuration commands.

項目名	説明
Cmd. Type	機器設定コマンドの種別を表す．具体的に次の種別がある． template　項目値のラベルに応じて必要性が決まる機器設定コマンドを表す． header　機器設定の最初に実行する必要がある機器設定コマンドを表す． footer　機器設定の最後に実行する必要がある機器設定コマンドを表す． mode-before　モードが変更される直前に実行する必要がある機器設定コマンドを表す．ここでモードとは，設定可能な設定項目群を定めた状態を指し，また，モード変更とは設定可能な設定項目群を定めた状態を別の状態に切り替える行為を指す． mode-after　現在のモードで必要な機器設定コマンドを全て発行し終えた直後に実行する必要がある機器設定コマンドを表す． 上記の種別は必ずしも全てを使う必要は無く，例えば図 8 では mode-before は記述していない．
Spec. Item Group	仕様項目グループを表す．ここでの仕様項目グループは，どの機器設定コマンドが必要かを判断する情報の一つになる．例えば，この要素が Hostname という値をとる場合，Hostname の特定の仕様項目の項目値に特定のラベルがあるときに，同じ行の機器設定コマンドが必要になることを意味する．本要素は，template コマンドに対してのみ有効となる．
Spec. Item	仕様項目を表す．ここでの仕様項目は，どの機器設定コマンドが必要かを判断する情報の一つになる．例えば，この要素が port/shutdown という値をとる場合，“port か shutdown のいずれか” の項目値に特定のラベルがあるときに，同じ行の機器設定コマンドが必要になることを表す．なお，*は “Spec. Item Group が有する全ての仕様項目のいずれか” を表す略記法である．本要素は，template コマンドに対してのみ有効となる．
Proc. Type	要新規設定 set，要設定解除 unset またはその双方いずれか（set/unset）を意味するラベルを表す．例えば，この要素が set という値をとる場合，項目値のラベルが set であるときに，同じ行の機器設定コマンドが必要になることを意味する．本要素は，template コマンドに対してのみ有効となる．
ID	機器設定コマンドを識別するための番号を表す．
Command	機器設定コマンドを表す．ここでの機器設定コマンドは，部分をメタモデルの仕様項目名で抽象化できる（例：hostname <name>）．<と>に挟まれた文字列（例：<name>）は仕様項目名を表す．機器設定コマンドが具体化されるときは，<仕様項目名>（例：<name>）が項目値（例：Router1）に置き換えられる．本要素は，template コマンドに対してのみ有効となる．
Modal	機器設定コマンドがモード変更を生じるか否かを表す．モード変更が生じる場合は TRUE と表記し，生じない場合は空値となる．本要素は，template コマンドに対してのみ有効となる．
Dep. ID	先に実行しておく必要がある機器設定コマンド（優先コマンド）を ID で表す．これはコマンド発行順序の制約となる．例えば，予めモード変更が必要な機器設定コマンドは，当該モード変更を生じる機器設定コマンドの ID を与える．優先コマンドが存在しない場合は空値とする．
Condition	機器設定コマンドの要否を判断する項目値の条件を表す．空値の場合は，恒真とみなす．本条件を満たさないとき，当該機器設定コマンドを不要とみなす．条件記法は等価演算子（==）のみを用い，例えば<仕様項目名> == 項目値（例：<shutdown> == true）と表す．本要素は，template コマンドに対してのみ有効となる．

注：略称の正式名称は次の通りである．Cmd. Type = Command Type; Spec. Item Group = Specification Item Group; Spec. Item = Specification Item; Proc. Type = Process Type; Dep. ID = Dependent ID.

のインスタンスが，自分自身と同じ型のインスタンスを参照することになり，単方向リストを表現している．これにより，ネットワーク構成モデル内で登場する AccessList の機器設定手順は，関連づけされている順番から機器設定手順が生成されるアルゴリズムとなっている．また，ネットワークを運用する際，ACL に追加や削除などの変更を加える場合がある．一般的に，ACL の変更には次の 2 つの方法があると考えられる．

1. **既存の ACL を編集する方法**　ACL に書かれた条件文には，それぞれシーケンス番号が与えられている．シーケンス番号とは，ACL 内で定義した条件文それぞれに昇順に割り当てられた番号であり，ルールが実行される順序を表している．通常，シーケンス番号は指定が無い限り，等間隔（例：10 条件文 A 20 条件文 B

...）に与えられ，この番号を用いることで既存の ACL に変更を加えることができる．ACL を追加する場合は，等間隔に設定されたシーケンス番号の間に入れるようにシーケンス番号を付与して行う．たとえば，シーケンス番号 10 が割り当てられた条件文 A と 20 が割り当てられた条件文 B の間に条件文 C を追加したい場合は，条件文 C のシーケンス番号を 11 から 19 のいずれかを与えて追加することで，条件文との間に設定を追加することができる．一方で削除する場合については，削除したい条件文に割り当てられたシーケンス番号を指定することで条件文を削除することができる．また，シーケンス番号は等間隔に均すことができるため，設定変更の都度行う．

2. 新たに ACL を作成する方法　既存の ACL を残したまま新たに別の ACL を作成し，設定変更をする方法がある．1 の方法と異なり，シーケンス番号を考慮する必要がなく，既存の ACL の条件文を含め，新たに追加したい条件文を併せて ACL を新たに作成する．

本手法では，“2. 新たに ACL を作成する方法”を採用しており，ACL に変更があった場合，既存の ACL に変更は加えず，新たに ACL を作成し，適用し直すことで設定変更を行っている．

次に機器設定手順の生成アルゴリズムを説明する．モデルに `Config` グループ値が複数登場する場合は，下記 (1)–(7) の処理は `Config` グループ値ごとに行うものとする．

(1) コマンドを処理順で保持するリスト（以下，`list`）と，`Dep. ID` に基づく優先コマンド関係を階層構造にして保持するツリー（以下，`tree`）の 2 つを用意する．（**図 9** 右が `list` と `tree` のイメージとなる．）

(2) 図 9 に示すように，`header` コマンドを `list` に入れ，もし `list` 要素内のコマンド C_a に優先コマンド C_p があれば，C_p を上位とし，C_a を下位とするコマンド構造を `tree` に反映する．状況によっては，`list` 要素内のコマンド C_b に複数の優先コマンド C_{p1}，C_{p2}（ただし，$\mathrm{ind}(C_{p1}) < \mathrm{ind}(C_{p2})$）が存在する場合がある．ここで $\mathrm{ind}(C)$ は `list` 中のコマンド C の index を取得する関数とする．このときは，index の大きい（直近の）コマンド C_{p2} に対して，C_{p2} を上位とし，C_b を下位とするコマンド構造を `tree` に反映する．

(3) 設定解除コマンドを生成するため，AsIs モデルの `Config` を次のように処理する．

(a) `Config` を根ノードとした木構造としてモデルを捉え，行きがけ順深さ優先探索を行う（**図 10**）．ただし，別の機器への結線情報である `Link` に到達したときは，`Link` は処理せずに引き返して探索に戻る．

(b) 探索中に到達したノード（仕様項目グループ値）に対し，機器設定コマンドテンプレートにおける当該グループ値に対応づく `unset` 分類の `Command` を上から順に処理する（**図 11**）．つぎのどちらかの条件を満たしたときに，`Command` を項目値で具体化して `list` へ追加し，優先コマンドがあれば `tree` に反映する．

条件 1.

- `Spec. Item` に紐づく項目値に 1 つでも `unset` ラベルがある，かつ，
- 仕様項目名に対応する項目値がすべて非空値である（仕様項目名が 1 つも存在しない場合は真とみなす），かつ，
- `Condition` を満たす（`Condition` が空値の場合は真と

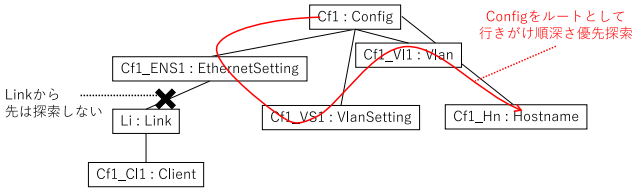

図 10　仕様項目グループの探索

Fig. 10　Overview of traversal algorithm for specification item groups

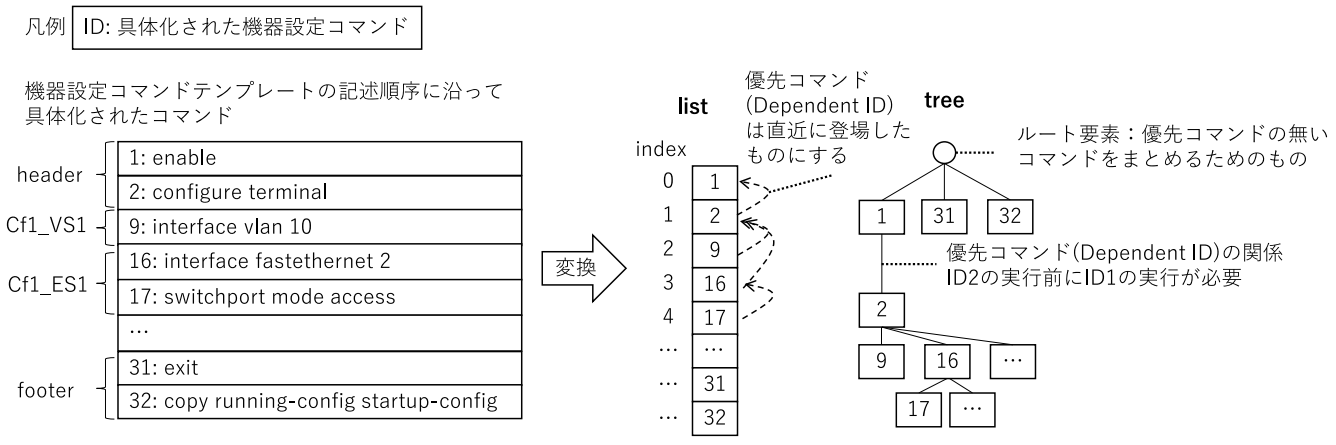

図 9　header,template,footer コマンドの整理

Fig. 9　Processing of header, template, and footer commands

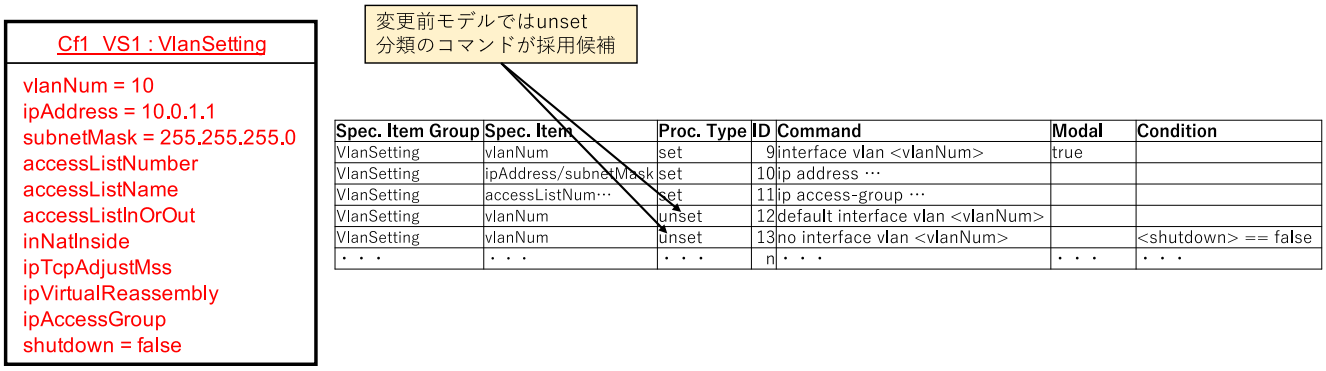

図 11　仕様項目グループの変更に応じるコマンドの選定

Fig. 11　Selection of commands according to changes in specification item groups

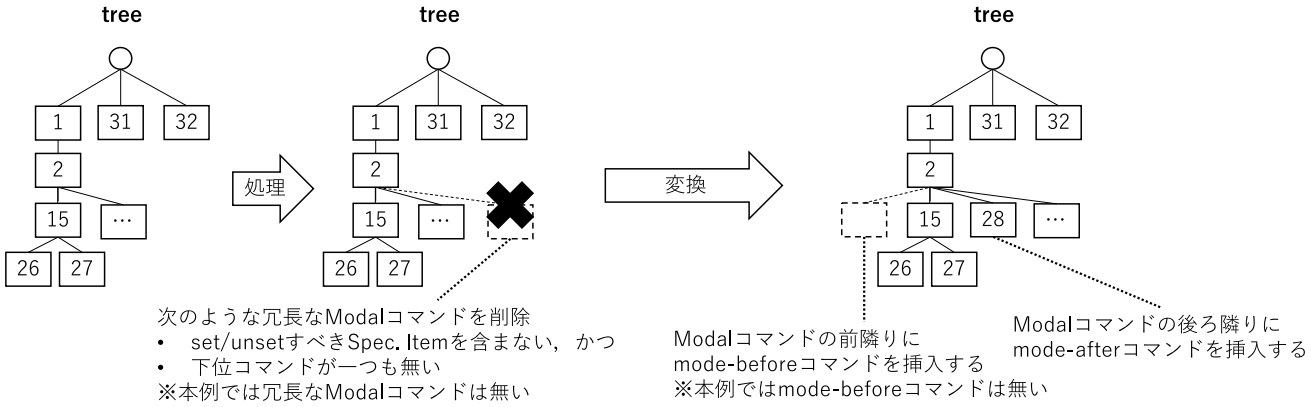

図 12　Modal コマンドの整理と mode-before と mode-after コマンドの挿入

Fig. 12　Processing of modal commands and insertion of mode-before and mode-after commands

みなす）．

条件 2.　Modal が true である．（当該コマンドの真の必要性は後で判定する．）

(4) 新規設定コマンドを生成するために，ToBe モデルの Config に対し，設定解除コマンドと同様な手順で Command を処理する．ただし，この手順では，前述の設定解除コマンドの手順中の“unset”の部分を“set”に読み替えるものとする．

(5) footer コマンドを header コマンドと同様に処理する．

(6) mode-before コマンドと mode-after コマンドを次のように処理する（図 12）．

(a) Modal が true な機器設定コマンド（Modal コマンド）について，Spec. Item に紐づく項目値に set および unset のラベルが無く，かつ，tree 上で下位コマンドが 1 つも無いものを冗長な Modal コマンドとして tree から削除する．

(b) tree を行きがけ順深さ優先で探索して，Modal コマンドの前隣りに mode-before のコマンド（群）を追加し，Modalコマンドの後ろ隣りに mode-after のコマンド（群）を追加する．このとき，mode-before や mode-after のコマンドがそれぞれ複数存在する場合は，順序は機器設定コマンドテンプレートに定義された順とする．

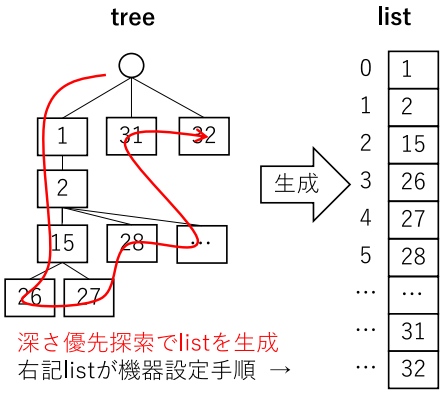

図 13　機器設定手順の生成

Fig. 13　Generation of configuration commands for network elements

(7) tree に対し，図 13 に示すように行きがけ順深さ優先探索をして到達したノード（機器設定コマンド）順に

list を再作成することで，最終的な機器設定手順を生成する．

3. 生成された機器設定手順について

2.2.2 項で詳述した，生成された機器設定手順について詳解する．紙面の都合上，生成された例として，一部のみを掲載している．適用事例の Cf1 に着目し，機器設定手順を生成してみると，**図 14** で示すように，変更差分のみが生成されていることが分かる．また，設定削除をするためのコマンドが生成されている点からも，不要になった情報を差分から検出できていることが確認できる．生成された機器設定手順は，各ネットワーク機器のコンソールモードから発行（コピーしてペースト）することで，ToBe ネットワークに必要な設定が行われる．

4. 評価

図 4 と図 5 で示したネットワークを適用事例とし，期待される機器設定手順を生成できるかどうかの観点から提案手法の有効性を評価した．本評価では，提案手法を実行するツール（提案ツール）を 2.2.1, 2.2.2 項で詳述したアルゴリズムを参考に Java で試作して用いた．また本手法では，モデルの作成に astah [14] を用いており，astah では作成したモデルの情報を Java プログラムから抽出するための API を提供している．その API を利用することで，モデル間差分の検出に必要な情報の取得し，ラベルの付与をアルゴリズムに沿って行っている．また，その情報に付与されたラベルと機器設定コマンドテンプレートを用いることで，実行可能な機器設定手順の生成を行う．さらに，提案ツールより生成された機器設定手順の正しさを確認するために，実機を用いて検証を行った．その際，ネットワーク運用管理のエキスパート 1 名の助力を得て，ルーティングテーブル，断線時の経路変更が正しく行えるかどうかの計 2 つの項目について確認をした．

```
enable
configure terminal
default interface vlan 10
no interface vlan 10
no ip route 192.168.40.0 255.255.255.0 192.168.30.2
no ip route 192.168.50.0 255.255.255.0 192.168.40.1
interface fastethernet 2
exit
default interface fastethernet 2
no vlan 10
router ospf 1
router-id 3.3.3.3
area 2 virtual-link 4.4.4.4
network 10.0.2.0 0.0.0.255 area 1
network 10.0.3.0 0.0.0.255 area 2
area 1 virtual-link 2.2.2.2
exit
interface vlan 30
ip address 10.0.3.1 255.255.255.0
exit
interface vlan 20
ip address 10.0.2.2 255.255.255.0
exit
exit
copy running-config startup-config
```

図 14　実際に生成された機器設定手順の一部（Cf1 に該当）
Fig. 14　Generated configuration commands (for Cf1; partially)

4.1 適用事例のネットワーク

本評価で用いる適用事例のネットワークの構成について詳解する．今回用いた適用事例は，各キャンパスの拠点間の相互接続を小規模化し，表している．

4.1.1 変更前の AsIs ネットワーク

変更前の AsIs ネットワーク（図 4）は，以前，信州大学で運用されていたネットワークを再現したものである．ルーティングには，スタティックルーティングを採用しており，ルートを手動設定し，冗長構成が考慮されていない運用形態となっている．そのため，一か所でネットワーク障害が発生してしまうと，障害発生個所の拠点からの通信が途絶えてしまうようなネットワーク構成となっている．また，それぞれのネットワーク機器の名前（Hostname の name）を campus1 からcampus6 までとしている．また，campus1 から得たルーティングテーブルの情報を リスト 1 に示す．このとき，C は，connectedを意味し，直接接続されていることを表す．また，S は，static を意味し，設定したスタティックルートを表す．

4.1.2 変更後の ToBe ネットワーク

次に，変更後の ToBe ネットワーク（図 5）は，現在，信州大学で運用されているネットワークを再現したものである．ルーティングには，ダイナミックルーティングとスタティックルーティングを併用しており，主にダイナミックルーティングでは OSPF プロトコルを採用しているため，エリアを複数に分けた，冗長構成が考慮されている運用形態となっている．そのため，一か所でネットワーク障害が発生した場合には，迂回経路に自動で切り替わるようなネットワーク構成となっている．また，それぞれのネットワーク機器の名前（hostname）を，campus1 から campus7，dc（データセンタ），top（出入口）としている．

4.2　機器設定手順の評価

自動生成された機器設定手順の評価について詳解する．一般に一回の構成変更に対して必要かつ妥当なコマンド列は複数仮定できるが，本評価では（1）ネットワーク運用管理のエキスパートが受容でき，かつ（2）実ネットワーク機器を用いて構築した（AsIs の）ネットワークへ適用した後に期待された動作かどうかを，ルーティングテーブルと経路確認により評価する．特に（2）の動作確認については，ICMP プロトコル [15] を使用したネットワーク断線時の経路の確認を行い，期待とおりに動作するかを確認した．ここでの期待とおりの動作とは，各機器間（図 5 で示したネットワーク機器間）のパケット送受信において欠損が無いことと，経路情報が問題なく得られるか（図 5 で示したネットワーク機器間の経路情報）の 2 つの条件を満たすものである．

4.3　準備

本評価で用いた具体的な入力データ，機材は以下のとおりである．

- AsIs モデル（図 6）
- ToBe モデル（図 7）
- 機器設定コマンドテンプレート（図 8）
- Cisco 1812-J * 7 台（Version 12.4(15)T11）
- Cisco 892 * 2 台（Version 15.1(4)M5）

また，AsIs モデルと ToBe モデルは，モデリングツール astah により作成し，機器設定コマンドテンプレートは CSV 形式により表現した．

次に，実験手順を以下に示す．

(1) 実験者が，入力データを提案ツールに入力して機器設定手順を生成する
(2) ネットワーク運用管理のエキスパートが，生成された機器設定手順に問題があるかどうかを確認する
(3) 実験者が，ToBe ネットワークを構築するため，生成された機器設定手順を実機に発行する
(4) 実験者が，ネットワークの運用管理のエキスパートの監視の元，ルーティングテーブル，断線時の経路を確認することで，期待どおりの振る舞いかどうかを確かめる

4.4　結果

生成された機器設定手順を実機に発行し，ルーティングテーブルの確認と traceroute コマンドによる経路の確認，さらには，意図的に断線障害を起こし，目的のネットワーク機器への経路が正しく表示されるかどうかの確認を行ったところ，正常にネットワークが動作していることを確かめた．一方で機器設定手順の生成において，`Config` を根ノードとした仕様項目グループの探索では，`Config` の子ノードにあたる仕様項目グループ値間に探索順序の制約が与えられていないため，実行によっては区切られたコマンド列（例：`vlan` と `hostname`）間が入れ替わる結果が得られることがあったが，本例においては，機能的には問題がないことを確認した．

4.4.1　ToBe ネットワークのルーティングテーブル

AsIs ネットワークの各ネットワーク機器と新しく追加したネットワーク機器に対し，実際に出力された機器設定手順を発行した．その際の，`campus1` から得たルーティングテーブルをリスト 2 に示す．このとき，`O` は，LSA Type1, Type2 によって得られた経路情報を表しており，`O;AI` は，LSA Type3 によって得られた経路情報を表している．いずれも，OSPF プロトコルが適用されているため，異なるエリア間でも，問題なく経路情報が得られていることが分かる．

4.4.2　断線障害によるルーティングテーブルの様子

OSPF プロトコルが問題なく適用されているかどうかを確かめるために，意図的に断線障害を起こし，経路情報を確認した．`campus4` から `top` に向けて，traceroute コマンドを発行した様子をリスト 3 に示す．このとき，すべての経路に関して，`campus4` から `top` までのルートを全て網羅していることから，動作に問題がないことが確認できる．次に，(`campus4`) から `campus5` 間のネットワーク（10.0.6.0/24）を断線させた場合の経路の様子をリスト 4 に示す．このとき，断線されたネットワーク（10.0.6.0/24）を経由せず，迂回経路を判断し，`top` までの経路を示していることから，期待どおりの振る舞いをしていることが分かった．

4.5　考察

評価結果に基づき，期待どおりのネットワーク機器設定手順が得られたことは，提案手法に一定の有効性があったことを示していると言える．これにより，提案手法では，モデル間差分が適切に得られ，そこから必要な機器設定コマンドが機器設定コマンドテンプレートを通じて妥当に得られた．

加えて，機器設定手順を自動生成することの効果として，エンジニアによる記述の抜け漏れの防止や誤解への気づきやエンジニア間の記述ポリシーの違いの発見，そして設計段階における機器設定手順の生成によるエンジニアの負担軽減にも役立つ見込みを得た．一方で，本手法を適用するにあたり必要となるモデルを作成するエンジニアへの負担については考慮していない．特に仕様項目値の重複によって，同一ネットワークの IP アドレス等の重複や，妥当ではない値を付記したモデルの不備による不適切な機器設定手順の生成へとつながる可能性が考えられる．そこで，これらの対策として，エンジニアがモデルを作成しやすいように，モデル作成段階でモデルチェックが可能な静的検証 [16] ツールの開発や，ネットワークシミュレータ

```
C    192.168.30.0/24 is directly connected, Vlan30
C    192.168.10.0/24 is directly connected, Vlan10
S    192.168.40.0/24 [1/0] via 192.168.30.2
C    192.168.20.0/24 is directly connected, Vlan20
S    192.168.50.0/24 [1/0] via 192.168.40.1
```

リスト 1　campus1 から得たルーティングテーブル（AsIs）
List 1 Routing table of campus1 in the AsIs network

```
O IA    10.0.8.0 [110/3] via 10.0.2.1, 01:52:25, Vlan20
O IA    10.0.9.0 [110/2] via 10.0.2.1, 01:52:25, Vlan20
C       10.0.2.0 is directly connected, Vlan20
C       10.0.3.0 is directly connected, Vlan30
O       10.0.1.0 [110/2] via 10.0.2.1, 01:52:25, Vlan20
O IA    10.0.6.0 [110/4] via 10.0.3.2, 01:52:45, Vlan30
O IA    10.0.7.0 [110/4] via 10.0.2.1, 01:52:25, Vlan20
O IA    10.0.4.0 [110/2] via 10.0.3.2, 01:52:45, Vlan30
O IA    10.0.5.0 [110/3] via 10.0.3.2, 01:52:46, Vlan30
```

リスト 2　campus1 から得たルーティングテーブル（ToBe）
List 2 Routing table of campus1 in the ToBe network

```
campus4#traceroute 10.0.1.2
...
  1 10.0.6.1 0 msec
  2 10.0.7.1 0 msec
  3 10.0.8.1 0 msec
  4 10.0.9.1 0 msec
  5 10.0.1.2 0 msec 0 msec *
```

リスト 3　campus4 から top までの経路情報（ToBe）
List 3 Routing information from campus4 to top in the ToBe network

```
campus4#traceroute 10.0.1.2
...
  1 10.0.5.1 0 msec 0 msec 0 msec
  2 10.0.4.1 0 msec 0 msec 0 msec
  3 10.0.3.1 0 msec 0 msec 0 msec
  4 10.0.2.1 0 msec 0 msec 0 msec
  5 10.0.1.2 4 msec 0 msec *
```

リスト 4　断線発生時の campus4 から top までの経路情報（ToBe）
List 4 Routing information from campus4 to top in the ToBe network when the link failure

を用いたモデル作成後の動的検証 [17] ツールの開発をすることが望ましいと考えられる．

また，本評価の適用事例は，ACL や OSPF プロトコル，VLAN 等の機能に対応しているが，他にも様々な機能やプロトコル，そして他ベンダ機器による独自プロトコルなどが存在する．そのため，提案手法をさらに拡充し，そのうえで有効性を評価することが今後の課題となる．また，`Config` の子ノードの探索順序の制約化や，エンジニアのポリシーに応じた機器設定手順の生成に提案手法が柔軟に対応できるか否かは評価できていない．

その他，ACL の設定変更においては作成時のモデルに依存するため，既存の ACL に変更を加える場合については考慮していないことが明らかになった．本手法を用いた ACL の設定変更を行う場合には，既存の ACL に変更は加

えず，新たにモデル内に ACL に関する情報を記述し，機器設定手順を生成する．そして，新しく作成された ACL を適用することで ACL の設定変更を行う仕組みとなっている．しかし，これらの ACL の変更に関しては，変更作業の状況に応じて選択をすることが適切であり，エンジニアのポリシーによって異なると考えられる．そのため，既存の ACL に変更を加えられる機器設定手順の生成システムを実装し，エンジニアのポリシーによって生成方法を選択できることが望ましいと考えられる．

5. おわりに

本稿では，ネットワーク構成モデルに基づく機器設定手順の自動生成手法を提案した．提案手法は，2 つのネットワーク構成モデル間の差分から，機器設定コマンドテンプレートを通じて機器設定手順（機器設定コマンド列）を生成するものである．実運用ネットワークを適用事例として提案手法の有効性を評価した結果，期待どおりの機器設定手順を提案手法により生成することができ，提案手法が有効であったことを確認した．

今後の課題として，エンジニアのポリシーに応じた柔軟な機器設定手順を生成できるように提案手法を拡充し，その有効性を定量的・定性的に評価することが挙げられる．また，マルチベンダを前提とした機器設定コマンド生成に対する提案手法の汎用性評価や，未対応の様々なプロトコルを適用した実運用ネットワークでの提案手法の有効性評価が挙げられる．さらには，作業手順書の自動生成に向けた提案手法の拡充や従来手法との連携方法の模索も挙げられる．まずは提案手法のさらなる表現力の向上と本手法における適用範囲の限界の調査，また機器設定手順の作成ポリシーについて様々なエンジニアへの調査を行っていきたい．

また，今回提案したモデルの作成時間や妥当性の評価についても今後の課題となるが，ネットワーク自動化手法として挙げられる，Ansible や NETCONF を組み合わせたネットワークの自動生成手法に関しての連携方法についても考慮をしていきたい．

謝辞　本研究を実施するにあたり，信州大学ネットワークの管理・運用に当たっている NTT 東日本様のご協力を得ましたので，ここに感謝の意を表します．また，本研究は JSPS 科研費 JP17KT0043 の助成を受けたものです．

参考文献

[1] ジュニパーネットワークス株式会社：ipa @ IT 読者調査で分かったネットワーク運用管理者が困っていること．〈https://www.juniper.net/assets/jp/jp/local/pdf/additional-resources/atmarkit-junos-survey-jp.pdf〉（参照 2021-11-18）．

[2] みやたひろし：インフラ/ネットワークエンジニアのためのネットワーク技術 & 設計入門：サーバシステムを支えるネットワークはこうしてできている，SB ソフトバンク株式会社（2020）．

[3] 山浦亘平，井口信和：ネットワーク機器の設定に用いる作業手順書作成支援システムの評価，インターネットと運用技術シンポジウム論文集，Vol.2020, pp.49–55（2020）．

[4] 吉澤政洋，沖田英樹，上原敬太郎，垂井俊明：仮想ネットワークに関する文書作成を支援するネットワーク管理システムの実装および評価，情報処理学会論文誌，Vol.52, No.3, pp.1334–1347（2011）．

[5] Huang, H., Ruan, Y., Shaikh, A., Routray, R., Tan, C. and Gopisetty, S.: Building end-to-end management analytics for enterprise data centers, *2009 IFIP/IEEE International Symposium on Integrated Network Management(IM)*, pp.661–675 (online), DOI: 10.1109/INM.2009.5188875 (2009).

[6] Harel, D. and Rumpe, B.: Meaningful modeling: what's the semantics of " semantics"?, *Computer*, Vol.37, No.10, pp.64–72 (online), DOI: 10.1109/MC.2004.172 (2004).

[7] Graaf, B. and Deursen, A. V.: Visualisation of Domain-Specific Modelling Languages Using UML, *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*, pp.586–595 (online), DOI: 10.1109/ECBS.2007.77 (2007).

[8] Deursen, A. V., Klint, P. and Visser, J.: Domain-Specific Languages: An Annotated Bibliography, *SIGPLAN Not.*, Vol.35, No.6, pp.26–36 (online), DOI: 10.1145/352029.352035 (2000).

[9] Schmidt, D. C.: Guest Editor's Introduction: Model-Driven Engineering, *Computer*, Vol.39, No.2, pp.25–31 (online), DOI: 10.1109/MC.2006.58 (2006).

[10] Lopes, F. A., Santos, M., Fidalgo, R. and Fernandes, S.: Model-driven networking: A novel approach for SDN applications development, *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp.770–773 (online), DOI: 10.1109/INM.2015.7140372 (2015).

[11] Rumbaugh, J., Jacobson, I. and Booch, G.: *The Unified Modeling Language Reference Manual*, Addison-Wesley Professional (1998).

[12] マーチンファウラー：UML モデリングのエッセンス第 3 版，翔泳社（2020）．

[13] 鈴木彦文，宇井哲也，古川すみれ，湯原大二郎，成瀬愼，浅川圭史，永井一弥，長谷川理：分散型サービス不能攻撃による擬似攻撃の実施と UTM の検証に関する考察，学術情報処理研究，Vol.21, No.1, pp.21–28（2017）．

[14] Change Vision: Astah.〈https://astah.net/〉（参照 2022-11-21）．

[15] 竹下隆史，村山公保，荒井　透，苅田幸雄：マスタリング TCP/IP 入門編第 5 版，オーム社（2019）．

[16] 大西　淳：UML におけるモデル整合性検証支援システム，電子情報通信学会論文誌．D-1，情報・システム．1, Vol.84, No.6, pp.671–681（2001）．

[17] 佐竹柊路，鈴木彦文，小形真平，新井　凪，岡野浩三：リンク障害に対するネットワーク設計の自動検証手法の試案，研究報告インターネットと運用技術（IOT），Vol.2022-IOT-58, pp.1–8（2022）．

新井 凪

2021 年信州大学卒業．同年同大学大学院修士課程入学．ネットワークモデリングに関する研究に従事．

小形 真平（正会員）

2007 年芝浦工業大学卒業．2009 年同大学大学院修士課程修了．2012 年同大学大学院博士課程修了．博士（工学）．2012 年信州大学助教，2020 年より同准教授．モデル駆動工学，オブジェクト指向開発および要求工学に関する研究に従事．IEEE，ACM，情報処理学会，電子情報通信学会，日本ソフトウェア科学会各会員．

鈴木 彦文（正会員）

1992 年信州大学工学部情報工学科卒業．1994 年同大学大学院博士前期課程修了．1997 年同大学大学院博士後期課程単位取得退学．修士（工学）．1997 年長野工業高等専門学校電子情報工学科助手．2003 年東京大学大学院新領域創成科学研究科基盤情報学専攻助手．2005 年信州大学総合情報処理センター准教授．2009 年信州大学総合情報センター副センター長准教授，2023 年より信州大学情報基盤センター副センター長准教授，セキュリティ，ネットワーク，Ad-Hoc ネットワーク，情報教育の研究に従事．情報処理学会，電子情報通信学会，教育システム情報学会各会員．

岡野 浩三（正会員）

1990 年大阪大学基礎工学部卒業．1993 年同大学大学院博士後期課程中退．同年同大学助手．同大学講師，助教授，准教授等を経て 2020 年信州大学工学部教授．2002 年ケント大客員研究員，2003 年バーミンガム大客員講師．2023 年工学部数理 DS/AI 教育研究センター長．博士（工学）（大阪大学）．ソフトウェア要求記述，モデル検査，機械学習等の研究に従事．電子情報通信学会，情報処理学会，ソフトウェア科学会，IEEE CS 各会員．