平成 27 年度

学士学位論文

OpenStack 環境でのオーケストレーション定義を容易にする GUI エディタの実現

1160304　　川口　貴大

指導教員　　横山　和俊

2016/02/19

高知工科大学 情報学群

要 旨

OpenStack 環境でのオーケストレーション定義を容易にする GUI エディタの実現

川口　貴大

近年，IT リソースの迅速な確保，コスト削減等の目的からシステムの基盤として IaaS の需要が高まっている．通常，システムは複数マシンの構成となり，大規模なシステムになるにつれ，構成定義に時間が掛かる．

OpenStack は最も開発が進んでいる IaaS 基盤ソフトウェアの一つである．OpenStack は Heat と呼ばれるコンポーネントでシステム自動構築をする．Heat は，IT リソースの構成情報を記述した設計図 (テンプレートファイル) を読み込ませることで，その構成情報を基に自動的にシステムの構築・起動を行う．そのため，テンプレートファイルの作成はシステムを構築する上で重要な役割を担っている．しかし，テンプレートファイルは書式が複雑であり，記述を行う際には Heat 独自の知識を要する．またテキストファイルであるため，記述量が膨大になり，さらに構成要素の関係が把握し難い問題もある．

本研究では GUI を用い，オーケストレーション定義を行うことにより、従来のテキスト入力における問題点を解決する新規テンプレートファイル作成ツールの開発を行い，従来の方法に比べて短時間かつ容易にシステム定義できることを示した．

キーワード　OpenStack，IaaS，Heat，オーケストレーション

Abstract

Recentry, there has been a growing demand for IaaS. The reason is rapid securing of IT resources, cost reduction. A system is usually the construction of more than one machines, and as it becomes large-scale, it takes time for a definition.

OpenStack is one of the IaaS foundation software development is advancing most. OpenStack is to automate system building by the component called Heat. Heat is to make them read the plan in which composition information on an IT resource was written, and a system is built automatically based on the composition information. Therefore when building a system, the making of a template file is charged with the important role. But a form is complicated and when describing, a template file needs original knowledge. There is a problem that a relation between the problem that the description amount becomes enormous and a component is difficult to grasp because it's a text file.

The new template file generation tool which settles a problem in conventional input in text-entry mode was developed by giving an orchestration definition using a GUI by this research, and a short time and the thing you can do system definition of easily were indicated compared with a conventional way.

key words OpenStack, IaaS, Heat, orchestration

目次

図目次

表目次

第 1 章

はじめに

サーバー仮想化や通信ネットワークの技術進歩に伴い，クラウドコンピューティングが普及している．一般ユーザー向けに提供されるサービスや，企業内で利用される専用アプリケーションなど，多くのサービスがクラウドを用いて提供されており，クラウドコンピューティングにおいて IaaS の需要が高まっている．総務省が公開している「平成 27 年度版情報通信白書」第 5 章第 2 節によると，図 1.1 に示す通り市場規模における IaaS の規模が，2018 年時予測の段階で 2012 年時規模の約 4 倍にまで増加する．[1] そのため，今後更に IaaS の需要が高まると考えられる．

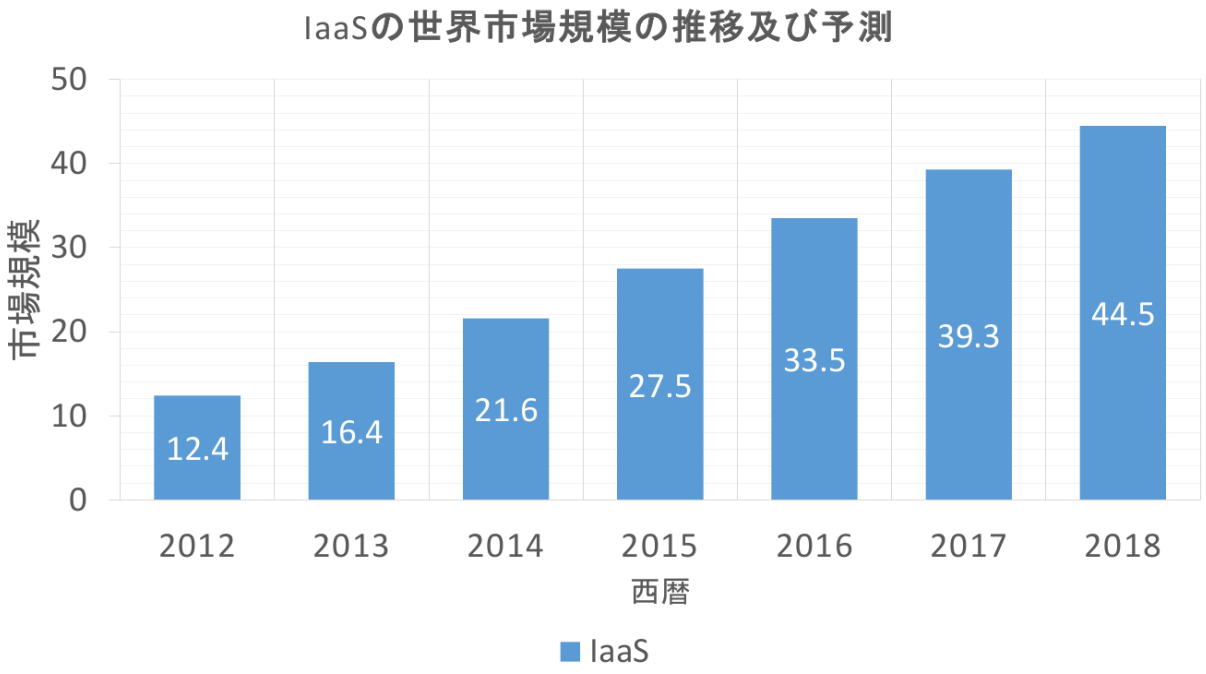

図 1.1　クラウドサービスの世界市場規模の推移及び予測

IaaS(Infrastructure as a Service) とは，システムの稼働に必要なサーバー，ストレージ，ネットワークなどのインフラを，ネットワーク経由で提供するサービスのことである [2]. IaaS を提供する側は物理サーバーや物理ネットワークを仮想化し IaaS 基盤を構築してサービスを提供する．IaaS を提供される側は，仮想化されたリソースを組み合わせシステムを構築，サービスを提供させることが可能となる．IaaS の代表的なサービスとして,「Amazon Web Services」や,「Microsoft Azure」等が挙げられる．これらのサービスは，提供しているベンダが 1 つであり，それぞれ独自仕様で開発された IaaS 基盤を用いている．

それに対し，多くのベンダが参加するコミュニティで開発されている IaaS 基盤構築ソフトウェアとして OpenStack が存在する [3]．OpenStack はオープンソース・ソフトウェアとして開発されており，Amazon Web Service や Microsoft Azure　と違い独自仕様で開発されていない．コミュニティに参加している企業は，オープンソース・ソフトウェアとして開発された OpenStack をベースとして IaaS を提供している．

OpenStack は，2010 年 10 月に初期バージョンである Austin がリリースされ，その後定期的に新しいバージョンを公開した後，2015 年 10 月 16 日に最新バージョンの Liberty が公開されている．機能別にコンポーネントが分かれており，各コンポーネントが相互に連携して動作する．OpenStack の主要コンポーネントを表 1.1 に示す.

表 1.1　OpenStack の主要コンポーネント

コンポーネント	機能
Glance	仮想マシンで使用されるゲスト OS の管理
Cinder	ブロックストレージにてゲスト OS 等を永続管理
Neutron	仮想ネットワークの管理
Horizon	OpenStack の操作管理を行う WebUI の提供
Swift	オブジェクトストレージの提供
Heat	仮想環境構築のためのオーケストレーション機能の提供

Heat とは，本来 OpenStack 利用者が手動で，各コンポーネントに指示を出し行っている仮想環境構築の手順を自動化する機能を提供している．自動化の手順としては，各コンポーネントを実行するために必要な項目を「Heat テンプレートファイル（以降テンプレートファイルと呼ぶ）」に記述後，テンプレートファイルを読み込むことで各コンポーネントで実行される内容を定義した後，自動で実行し仮想環境を構築を行うというものである．尚，テンプレートファイルには独自の書式が存在する [4].

OpenStack の各コンポーネントを自動化することができる Heat だが，現状問題が存在する．図 1.2 に問題点を示す.

表 1.2　Heat の問題点

問題点	詳細
テンプレートファイルから構成情報を把握しづらい	構成情報を全てテキストで記述しているため, 一見して構築途中または構築完了後の構成をテンプレートファイルからは把握しづらい.
テンプレートファイルの書式が複雑	入力内容を把握しづらい. どのコンポーネントへ命令を出すか，また各コンポーネント内で分岐している命令文の選択も複雑な書式といえる. また, 文中のインデントの深さで入力内容を区別するという特殊な書式もある.
膨大なテキスト記述量	手動入力で仮想環境のシステム構成について記述するため, 記述に膨大な時間がかかる.

第 2 章

オーケストレーション定義エディタの提案

2.1　オーケストレーション定義エディタの概要

オーケストレーション定義エディタとは，従来手動で行っていたテンプレートファイル作成を GUI ベースで作成補助をすることによりテンプレートファイル作成にかかる時間を削減し，容易に Heat を用いた仮想環境構築を可能にするエディタである．オーケストレーション定義エディタの概略図を図 2.1 に示す．

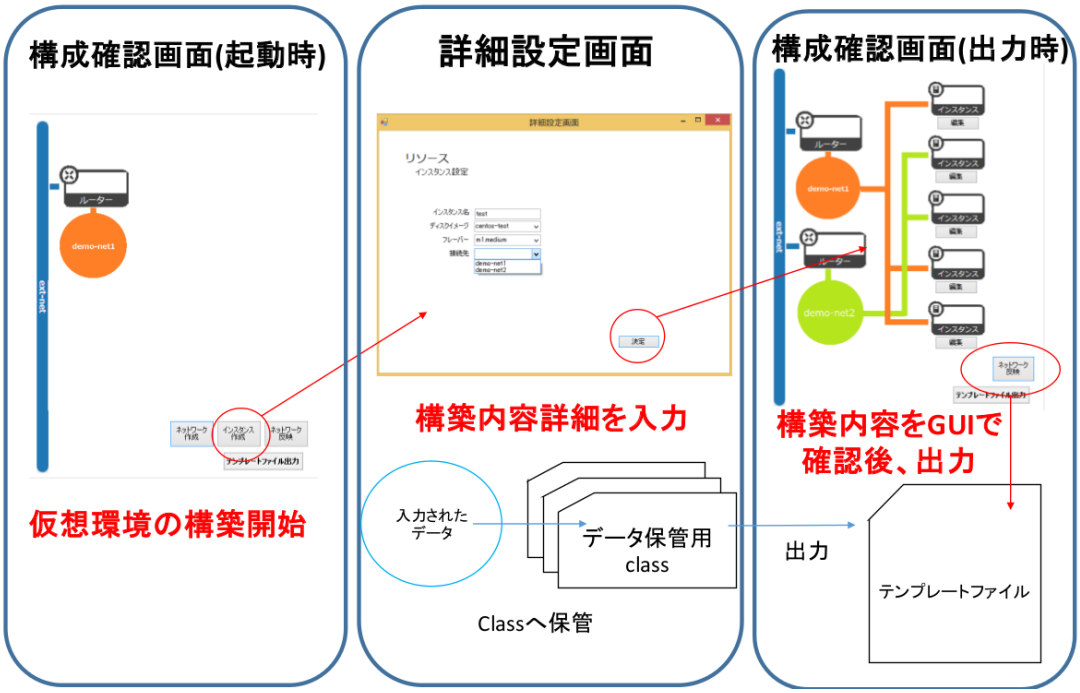

図 2.1　オーケストレーション定義エディタの概略

2.2　オーケストレーション定義エディタの要件

オーケストレーション定義エディタで取り扱う Heat は OpenStack 内のコンポーネントであるため利用者は Heat を使用する際には OpenStack についての最低限必要な知識を身に着けている必要がある. それ以外の前提知識を身につけていなくとも利用者がスムーズに仮想環境を構築するために表 2.1 にオーケストレーション定義エディタに求められる要件を定義した.

表 2.1　オーケストレーション定義エディタに求められる要件

要件	理由
操作インターフェイスは GUI	構築中のシステム構成を可視化するため
利用対象者は OpenStack に関する基本的な知識を有した学生	Heat は OpenStack 内のコンポーネントであり, OpenStack 内 GUI である Dashboard の利用者の多くは学生以上であるため
インスタンス名入力項目以外の手動入力方式を廃止	テキスト入力量を削減し, テンプレートファイル作成にかかる時間を短縮するため
入力項目の明確化	予め正しい入力内容へ誘導することで, テンプレートファイル読み込み時のエラーを抑止

2.3　Heat で扱うリソース

オーケストレーション定義を行う Heat では, 複数個のリソースを取り扱っている. お互いに依存しあうリソースについて記述を行うことで仮想環境を構築する.

2.3.1　リソースの依存関係

Heat で取り扱うリソースはそれぞれ他のリソースに依存している. 依存しているリソースを参照することで仮想環境を稼働させる. 依存関係を表 2.2 と図 2.2 に示す.

表 2.2　リソースの依存関係表

リソース	依存している他のリソース
ネットワーク及びサブネット	外部へ接続する必要があるため, 外部ネットワークを参照する. サブネットでは使用する IP アドレス範囲を指定.
ルーター	接続先を指定するためにネットワークを参照する.
ルーターインターフェイス	ネットワークとルータを接続している. どのネットワークが自身の依存するルータに接続されるのか管理している.
インスタンス	接続先を指定するためにネットワークを参照している. 参照したネットワークに応じて, 予めサブネットで範囲指定しておいた IP アドレスを割り振り, ルータを介して外部ネットワークへ接続できる.

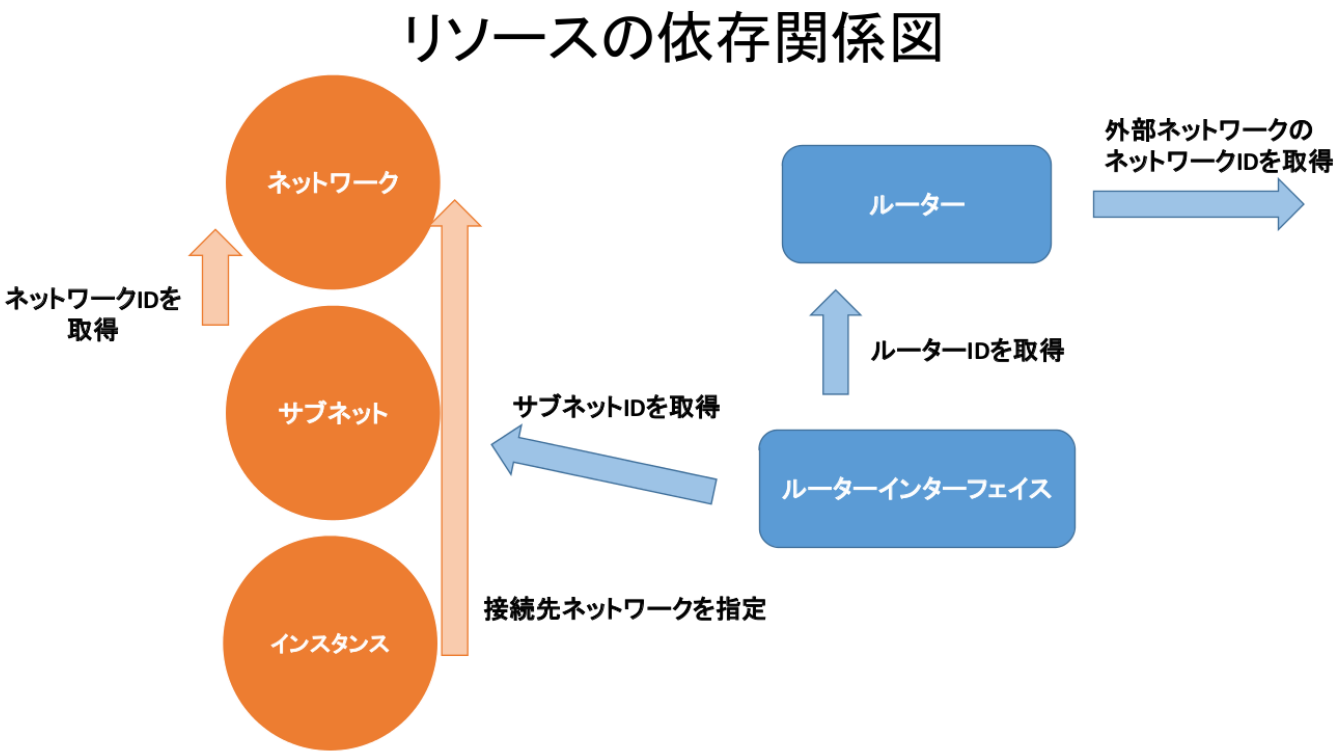

図 2.2　リソースの依存関係図

構成上最も外側のネットワークを参照できるのは内部に存在するネットワークである. 外のネットワークへの出口を確保することで, セグメント内から外部へネットワークを通じて接続できるようにしている. サブネットではセグメント内で使用する IP アドレスの幅を設定する. ここで設定された IP アドレス範囲内にインスタンスを接続する.

Heat では, これらリソースの依存関係, 参照先をテンプレートファイルへ記述後，テンプレートを読み込むことで自動で仮想環境を構築する.

2.4　テンプレートファイルへの出力補助方法

仮想環境構築を補助する Heat だが，テンプレートファイル作成には多大な時間がかかり，更には記述の為に Heat に関する専門知識が必要であるため手軽に利用できない. そこでテンプレートファイル作成を容易にするため，オーケストレーション定義エディタではテンプ

レートファイルへの出力補助を行う．

オーケストレーション定義エディタでは，ユーザに入力されたデータは一度「データ保管用 Class」へ保管する．(図 2.3) このデータ保管用 Class にはオーケストレーション定義エディタからであれば何度もアクセスと編集が可能である．そのため構築途中で既に記述をした項目を修正することも容易である．入力された複数のデータをデータ保管用 Class に保持し続けておき，テンプレートファイルへ出力を行う際にデータ保管用 Class を呼び出し Class 内に保管されているデータをテンプレートファイルへ出力する．

```
public class InstanceConfig
{
    public String instanceName;
    public int imageNum;
    public int flavorNum;
    public int networkNum;
    public string imageName;
    public string flavorName;
    public string networkName;

    public InstanceConfig()
    {
        this.instanceName = "";
        this.imageNum = 0;
        this.flavorNum = 0;
        this.networkNum = 0;
        this.imageName = "";
        this.flavorName = "";
        this.networkName = "";

    }
}
```

図 2.3　データ保管用 Class

第 3 章

オーケストレーション定義エディタの実装

3.1　動作環境

本研究で作成されたオーケストレーション定義エディタはプログラミング言語である C#を使用して作成されているので，動作には「Microsoft .NET Framework 4.6」環境が必要である．

3.2　モジュール構成

図 3.1 に，オーケストレーション定義エディタ内モジュールの構成を示す．オーケストレーション定義エディタは構成確認画面についての処理を行っている「MapForm.cs」と，詳細設定画面についての処理を行っている「EditorForm.cs」，そして EditorForm.cs で受け取ったデータを一時保存しておくための「InstanceConfig.cs」から構成されている．各モジュールで処理を終えた後，出力情報をテンプレートファイルに出力する．その後 Heat へ読み込ませることで仮想環境を構築する．

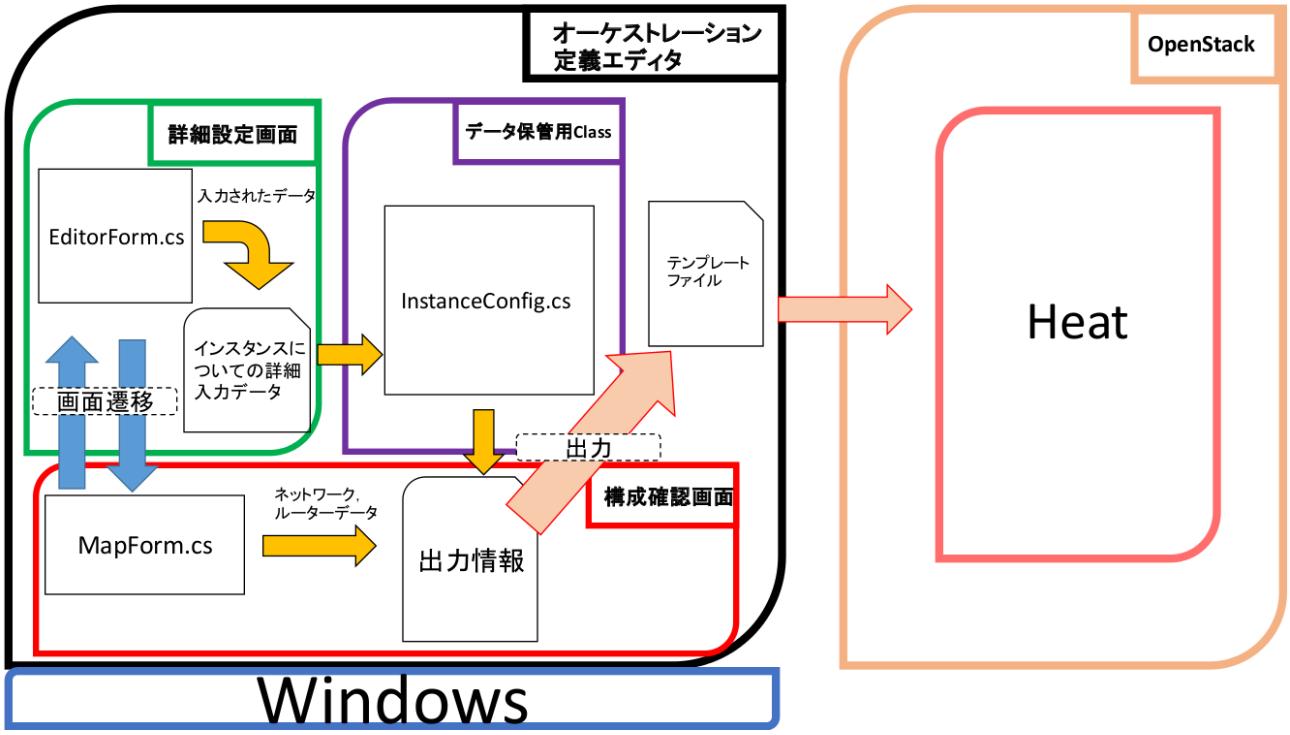

図 3.1　モジュール構成

3.3　画面構成

オーケストレーション定義エディタは 2.1 の概略図で示したとおり，大きく分けて 2 つの画面から構成されている．オーケストレーション定義エディタに表示される「構成確認画面」と呼ばれる画面と，構成確認画面内から遷移できる「詳細設定画面」という画面である．以下に各画面の詳細を示す．

3.3.1　構成確認画面

構成確認画面では，これから構築する構成を GUI ベースで確認することが可能である．図 3.2 に示す通り，オーケストレーション定義エディタ起動時には予め 1 セグメントのみ準備されている．これは，仮想環境を構築するのであれば最低 1 セグメントはネットワークが必要だからである． 1 セグメント目のネットワークに関する記述はどの構成でも必要な記述であるため，予めオーケストレーション定義エディタの構成確認画面ではルーターとネットワークを作成済みの状態で起動される．

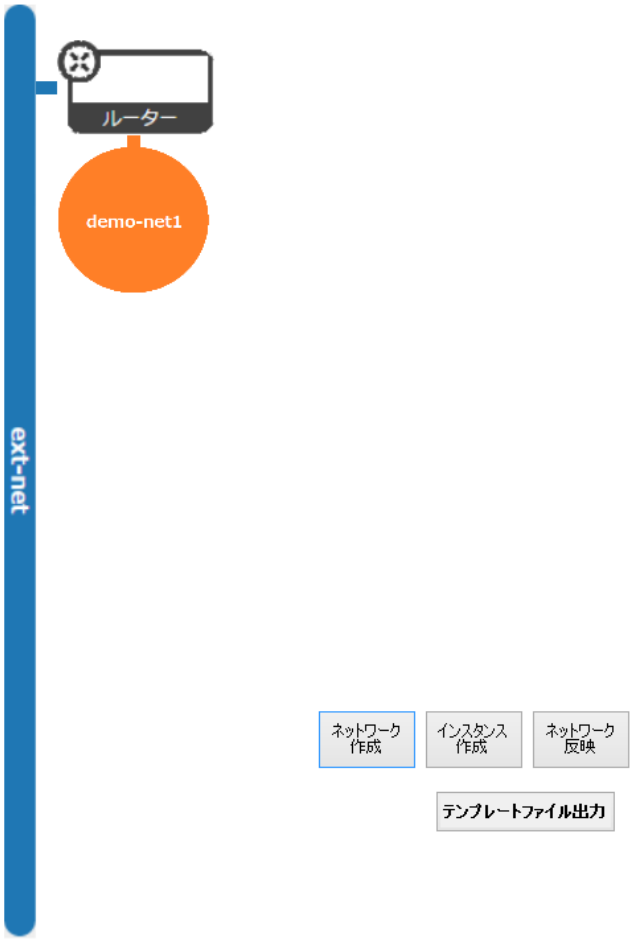

図 3.2　オーケストレーション定義エディタ-構成確認画面その 1

3.3　画面構成

オーケストレーション定義エディタ起動時の画面を図 3.3 に示す．１セグメント目のルーターとネットワーク以外にセグメントを増やしたい場合は，画面下部のネットワーク作成ボタンを押すことで次のセグメントを追加する．追加されたネットワークは，接続先ネットワークとして詳細設定画面から参照可能となる．

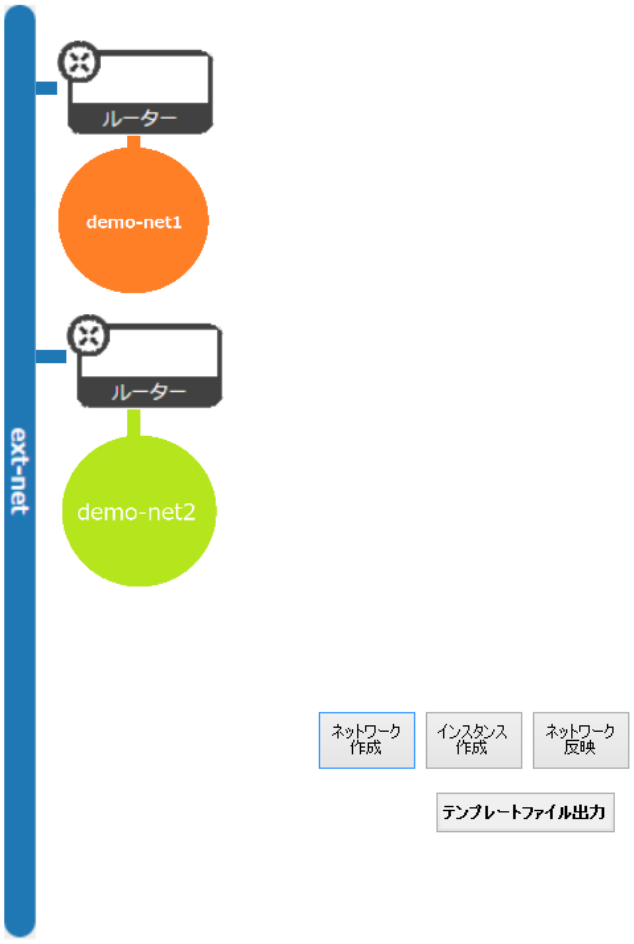

図 3.3　オーケストレーション定義エディタ-構成確認画面その 2

インスタンスを作成したい場合は，画面下部のインスタンス作成ボタンを押すことで新規作成するインスタンスを追加できる．ただしここで作成ボタンを押しただけではインスタンスに関する詳細設定ができていない状態であるため，作成後はインスタンスアイコンの下部

にある編集ボタンをクリックして詳細設定画面に遷移しなければならない．尚，詳細設定画面で項目を入力した後構成確認画面に戻り画面下部のネットワーク反映ボタンを押すと，図 3.4 のように GUI で接続先が把握できる．

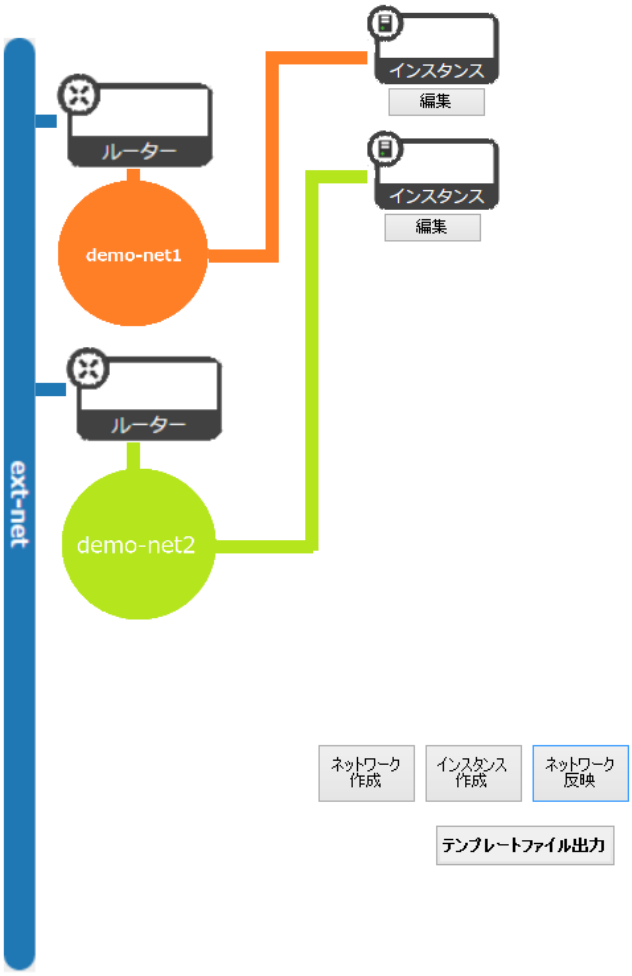

図 3.4　オーケストレーション定義エディタ-構成確認画面その 3

また，図 3.4 のように構成内容を決定した後でも，オーケストレーション定義エディタを終了させなければ構成内容の編集が可能である．図 3.4 の状態から一度インスタンスの接続先ネットワークを変更，作成するネットワークとインスタンスを増加させることも可能であり，図 3.5 と図 3.6 のように遷移する．

3.3　画面構成

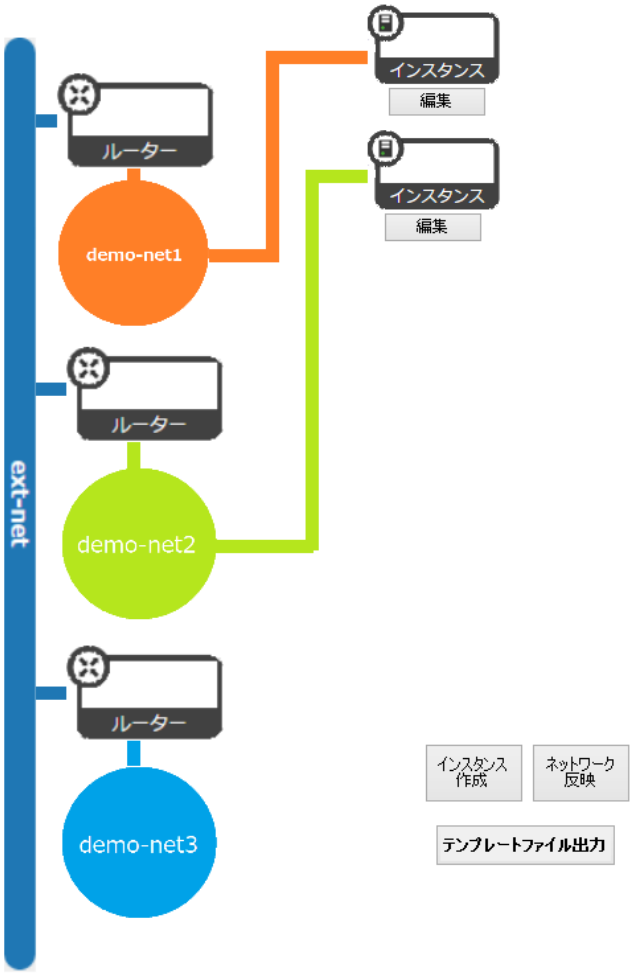

図 3.5　オーケストレーション定義エディタ-構成確認画面その 4

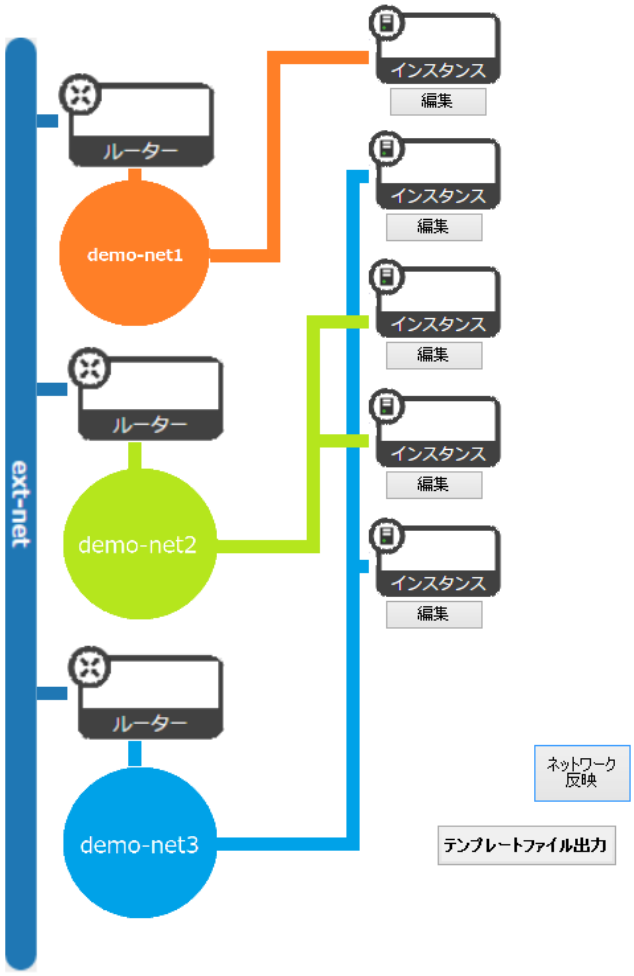

図 3.6　オーケストレーション定義エディタ-構成確認画面その 5

3.3.2　詳細設定画面

インスタンスに関する詳細設定は，構成確認画面から遷移できる詳細設定画面で行う．詳細設定画面の様子を図 3.7 に示す．詳細設定画面で入力できる 4 つの項目のうちインスタンス名を入力する項目 1 つを除き全てプルダウンメニューを採用している．インスタンス名のみ，自由に入力できる項目であるため手動入力形式であるが，その他 3 つの項目はある程度入力される内容が決まっているがインスタンス毎に入力内容が違う可能性がある．そこで選択肢を提供するためにプルダウンメニューを採用した．尚，構成確認画面でネットワーク

を追加した場合は「接続先」項目の選択肢が自動で追加される．入力されたデータは図 3.1，図 2.3 で示しているように「データ保管用 class」へ一時保存され，テンプレートファイルへデータを出力する際に入力される．

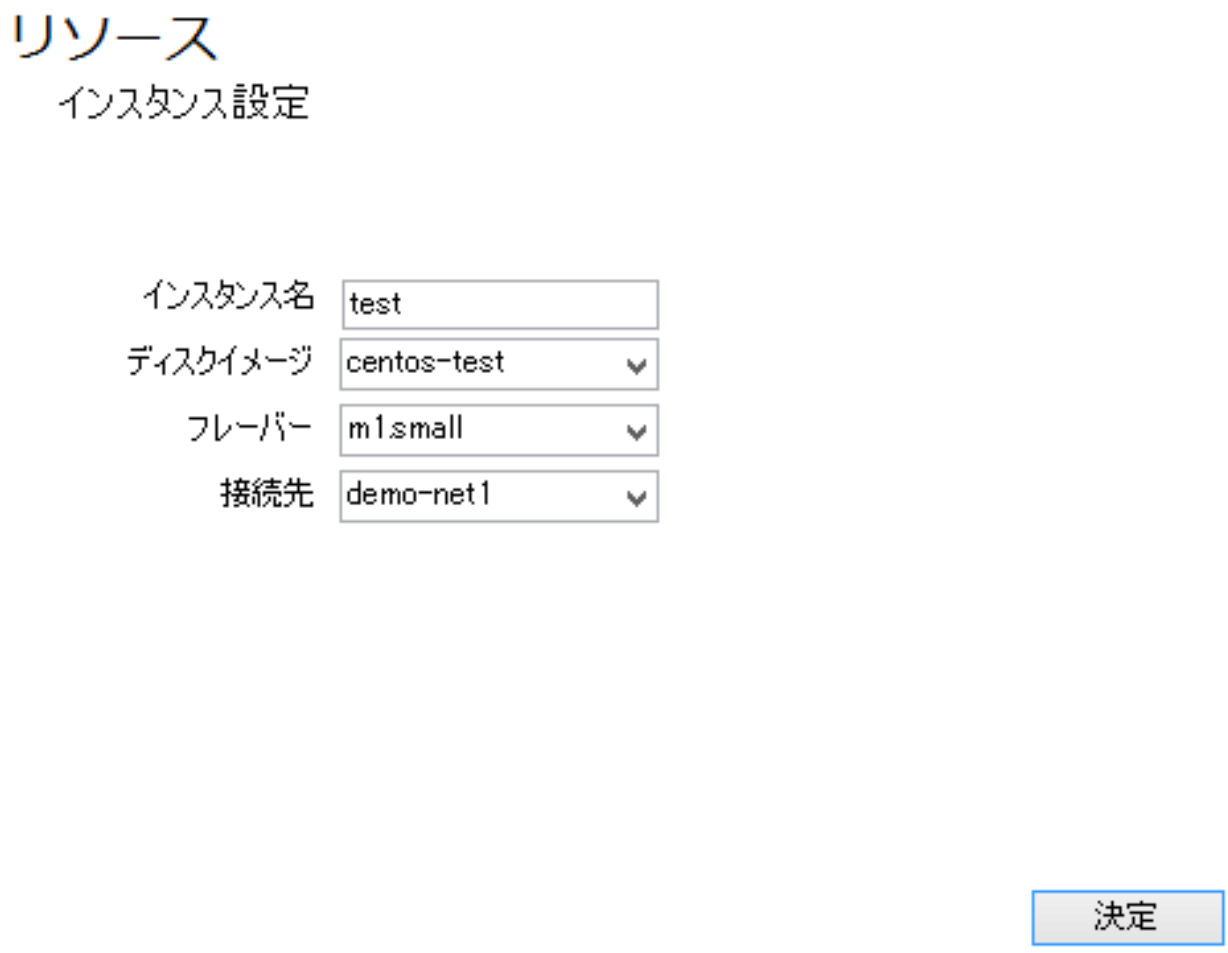

図 3.7　オーケストレーション定義エディタ-詳細設定画面

3.4　テンプレートファイル出力の流れ

オーケストレーション定義エディタで利用者から受け付けた入力データは,「3.3.1 構成確認画面」と「3.3.2 詳細設定画面」で説明した内容に沿い，最後にテンプレートファイル出力ボタンを押すことで，テンプレートファイルへと出力される．しかしこれでテンプレートファイルへ入力されるべき内容が全て入力されるわけではない．利用者側から入力された内容しかテンプレートファイルへ出力されず，本来ならばテンプレートファイルに入力されるべき内容が一部入力されていないままである．そこで，利用者側で入力しなかったデータがどのように扱われるか説明を行う．

3.4.1　その他利用者側で入力しなかったデータについて

構成確認画面と詳細設定画面で入力されなかったデータは，図 3.1 中に表されている「ネットワーク，ルーターデータ」として MapForm.cs から出力情報として自動で準備される．その後テンプレートファイルへ出力するタイミングで，ネットワーク，ルーターデータを自動でテンプレートファイルへ出力する．尚，この時同時にデータ保管用 class に一時保存された入力データも出力される．自動で入力されるデータ量は膨大であるため，図 3.8 にデータの一部と自動入力の様子を示す．

3.4　テンプレートファイル出力の流れ

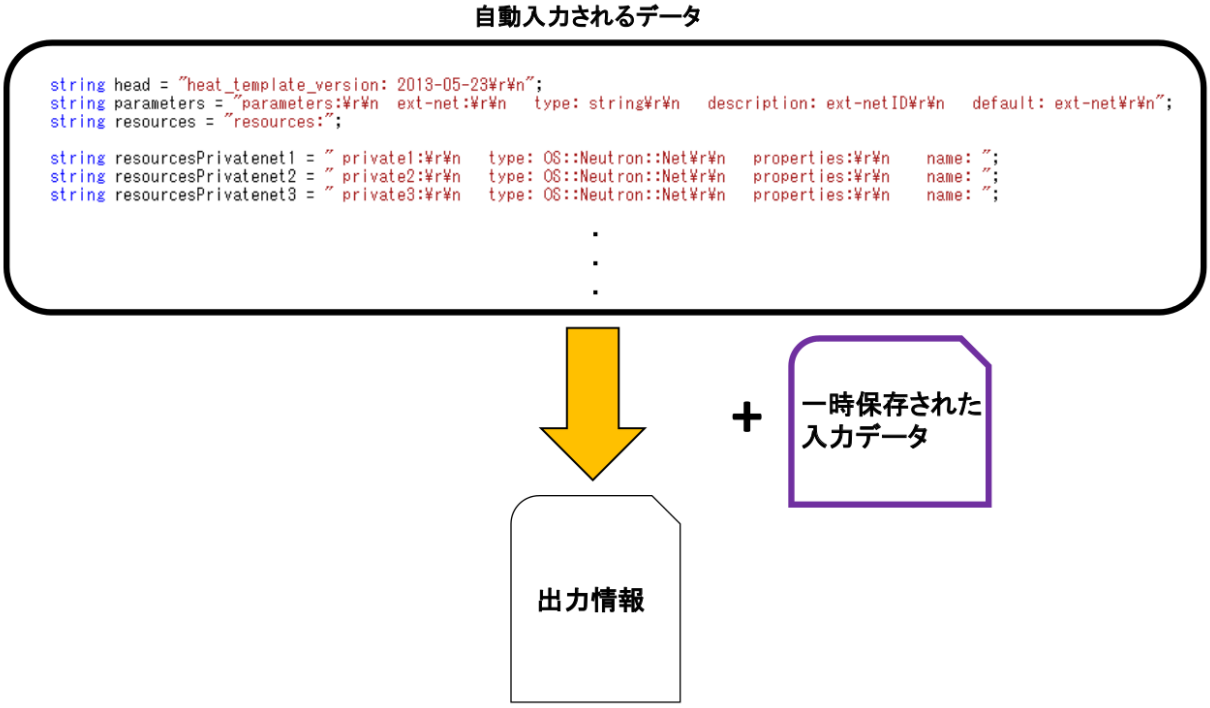

図 3.8　自動入力の様子

第 4 章

評価

4.1 評価の目的

1 章「はじめに」中で提示した「Heat の問題点」を解決できたか検証，従来方式との差を計測するために被験者の，各手法に関する前提知識の勉強時間, 作成インスタンス数とルーター数の増加させての作成所要時間, テンプレートファイル読み込み時にチェックされる記述ミスによるエラー発生数を計測, 従来方式とオーケストレーション定義エディタ使用時それぞれで同じシステム構成を構築してもらい比較する.

4.2 評価内容

従来方式, オーケストレーション定義エディタそれぞれでシステム構築をする場合に以下の表 4.1 に示している評価要素についてデータを記録する.

表 4.1　評価要素

評価要素	詳細
学習時間	各方式についての前提知識を説明用ドキュメントを用いて学習するのにかかった時間
作成所要時間	各方式それぞれテンプレートファイル作成開始からテンプレートファイルをエラー無しの状態で正常に読み込ませるまでに要した時間
エラー発生回数	テンプレートファイルを Heat に読み込ませた時にエラーが発生した回数. 正常に読み込み完了となるまで修正したテンプレートファイルを読みこませ直し続けるのでその都度エラーが発生すればカウントする

4.3　評価環境

本評価実験は, オーケストレーション定義エディタの動作環境を満たしている「Microsoft .NET Framework 4.6」環境で評価を行った.

また, 本評価実験で構築する仮想環境システム構成を以下に示す.

I．(1 セグメント 1 インスタンス構成)

作成するネットワーク（ルーター）数は 1，インスタンス数も 1.

II．(1 セグメント 3 インスタンス構成)

作成するネットワーク（ルーター）数は 1，インスタンス数は 3.

III．(1 セグメント 5 インスタンス構成)

作成するネットワーク（ルーター）数は 1，インスタンス数は 5.

IV．(2 セグメント 5 インスタンス構成)

作成するネットワーク（ルーター）数は 2，インスタンス数は 5.

Ⅴ．(3 セグメント 5 インスタンス構成)

作成するネットワーク（ルーター）数は 3，インスタンス数は 5.

Ⅰ，Ⅱ，Ⅲ 3 つの構成は単純にインスタンスの数のみを増加させ，残る Ⅳ，Ⅴ のシステム構成は基本的な構成は Ⅲ と同じだが, 作成するルーターの数（内部のネットワーク, サブネットの数）を増加させている. これは, インスタンスに関する記述のみを増加させた場合とルーターに関する記述を増加させた場合, 記述増加量の差があるため双方の記述所要時間とエラー発生回数の増加幅やデータの動きに差が生まれると予測したためである.

4.4　結果

評価実験の結果を示す．各被験者の学習時間の計測結果を表 4.2 に示す．被験者別の作成所要時間を表 4.3 から順に表 4.7 まで示す．また，被験者別のテンプレートファイル作成時エラー発生回数を表 4.8 から順に表 4.12 まで示す.

表 4.2　各被験者学習時間計測結果

被験者	オーケストレーション定義エディタ	従来方式
A	180(秒)	540(秒)
B	72(秒)	421(秒)
C	90(秒)	503(秒)
D	90(秒)	479(秒)
E	103(秒)	413(秒)

4.4 結果

表 4.3　被験者 A-作成所要時間計測結果

システム構成	オーケストレーション定義エディタ	従来方式
(I)	27(秒)	685(秒)
(II)	87(秒)	597(秒)
(III)	105(秒)	485(秒)
(IV)	113(秒)	594(秒)
(V)	160(秒)	660(秒)

表 4.4　被験者 B-作成所要時間計測結果

システム構成	オーケストレーション定義エディタ	従来方式
(I)	28(秒)	1216(秒)
(II)	166(秒)	931(秒)
(III)	190(秒)	745(秒)
(IV)	124(秒)	862(秒)
(V)	160(秒)	730(秒)

表 4.5　被験者 C-作成所要時間計測結果

システム構成	オーケストレーション定義エディタ	従来方式
(I)	36(秒)	1328(秒)
(II)	148(秒)	886(秒)
(III)	106(秒)	1532(秒)
(IV)	228(秒)	2070(秒)
(V)	162(秒)	2990(秒)

表 4.6 被験者 D-作成所要時間計測結果

システム構成	オーケストレーション定義エディタ	従来方式
(I)	71(秒)	1323(秒)
(Ⅱ)	117(秒)	916(秒)
(Ⅲ)	139(秒)	868(秒)
(Ⅳ)	188(秒)	1173(秒)
(Ⅴ)	145(秒)	1946(秒)

表 4.7 被験者 E-作成所要時間計測結果

システム構成	オーケストレーション定義エディタ	従来方式
(I)	52(秒)	1200(秒)
(Ⅱ)	107(秒)	683(秒)
(Ⅲ)	137(秒)	731(秒)
(Ⅳ)	189(秒)	2296(秒)
(Ⅴ)	229(秒)	1191(秒)

表 4.8 被験者 A-テンプレートファイル作成時エラー発生回数

システム構成	オーケストレーション定義エディタ	従来方式
(I)	0(回)	6(回)
(Ⅱ)	0(回)	3(回)
(Ⅲ)	0(回)	0(回)
(Ⅳ)	0(回)	0(回)
(Ⅴ)	0(回)	2(回)

表 4.9 被験者 B-テンプレートファイル作成時エラー発生回数

システム構成	オーケストレーション定義エディタ	従来方式
(I)	0(回)	7(回)
(II)	0(回)	3(回)
(III)	0(回)	3(回)
(IV)	0(回)	0(回)
(V)	0(回)	0(回)

表 4.10 被験者 C-テンプレートファイル作成時エラー発生回数

システム構成	オーケストレーション定義エディタ	従来方式
(I)	0(回)	6(回)
(II)	0(回)	1(回)
(III)	0(回)	5(回)
(IV)	0(回)	7(回)
(V)	0(回)	4(回)

表 4.11 被験者 D-テンプレートファイル作成時エラー発生回数

システム構成	オーケストレーション定義エディタ	従来方式
(I)	0(回)	3(回)
(II)	0(回)	3(回)
(III)	0(回)	3(回)
(IV)	0(回)	2(回)
(V)	0(回)	4(回)

表 4.12 被験者 E-テンプレートファイル作成時エラー発生回数

システム構成	オーケストレーション定義エディタ	従来方式
(I)	0(回)	1(回)
(Ⅱ)	0(回)	1(回)
(Ⅲ)	0(回)	0(回)
(Ⅳ)	0(回)	3(回)
(Ⅴ)	0(回)	1(回)

4.5 考察

4.5.1 学習時間について

従来方式，オーケストレーション定義エディタ双方の学習時間を比較すると，全ての被験者がオーケストレーション定義エディタについての学習時間が従来方式よりも短いことがわかる．(図 4.1)

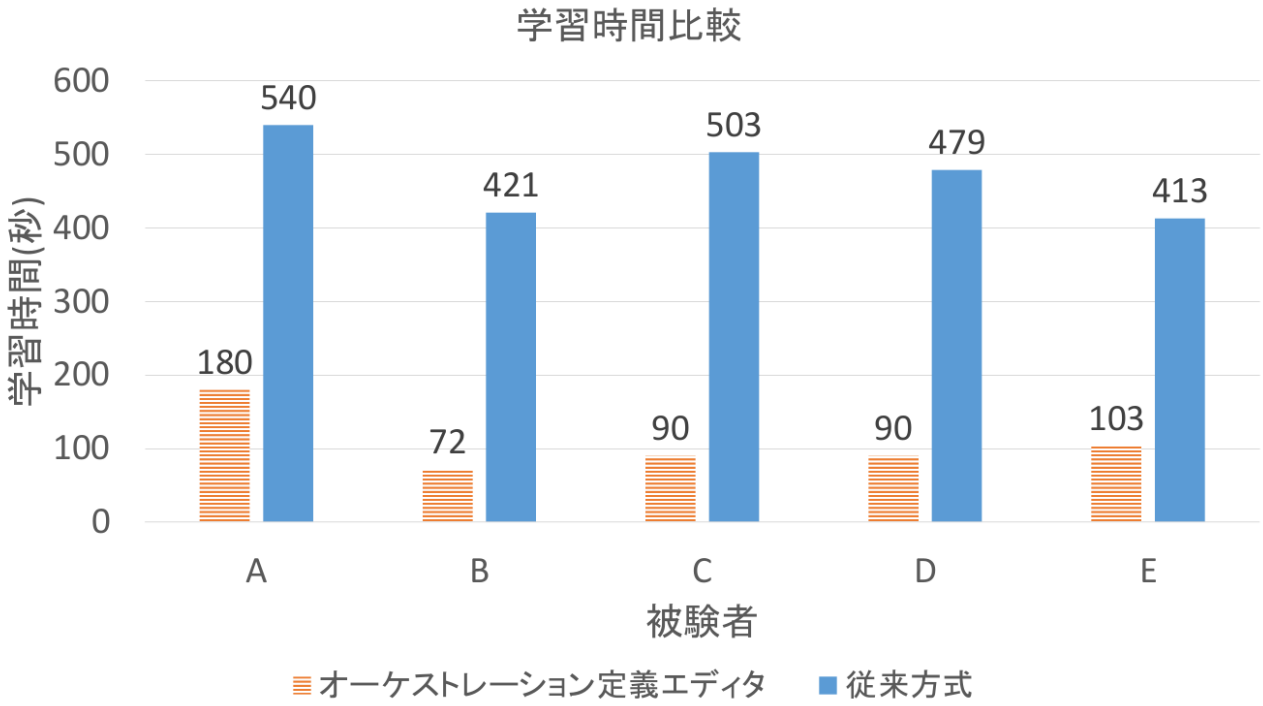

図 4.1　学習時間比較

従来方式では OpenStack に関する基本的な知識の他に，テンプレートファイル独特の書式について知識が必要であるため，学習すべき項目が多岐に渡ったからだと考えられる．

一方オーケストレーション定義エディタはボタンとプルダウンメニュー，1 箇所のみ手動入力項目に関する説明があるだけで学習すべき項目は従来方式に比べて大幅に少ないからだと考えられる．

以上のことから，学習時間の面において従来方式よりもオーケストレーション定義エディタのほうが導入が容易であることが示せたと考えられる．

4.5.2　作成所要時間について

記録する 3 つの評価要素のうち，従来方式とオーケストレーション定義エディタ大きく差がでたのが作成所要時間である．従来方式は 5 人の被験者間で作成所要時間に大きなば

らつきがあり，なおかつ作成所要時間そのものも長時間に及んでいた．一方オーケストレーション定義エディタは 5 人の被験者間で作成所要時間にばらつきはほとんど無く，作成所要時間そのものも従来方式に比べると大幅に少ない．従来方式において最も短い所要時間でテンプレートファイルを作成した被験者 A と最も長い所要時間で作成した被験者 C の比較を図 4.2 に示す．

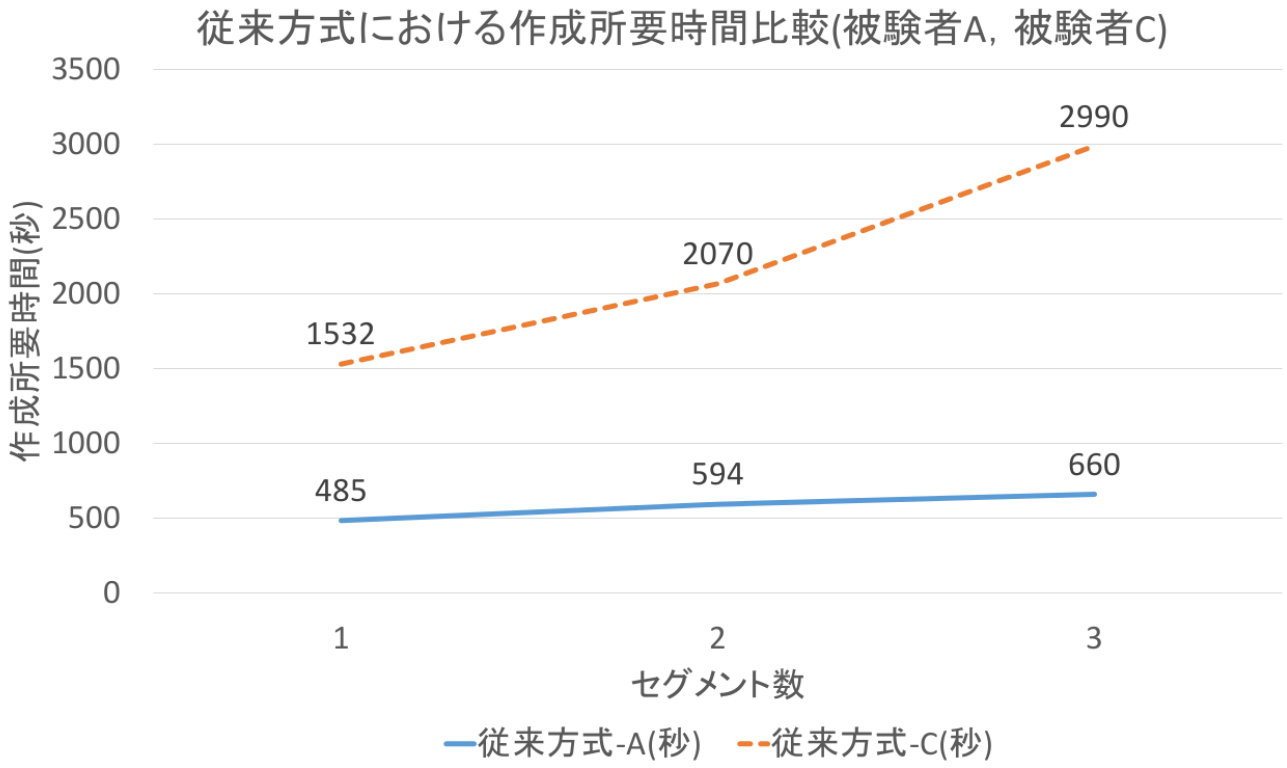

図 4.2　従来方式における所要時間比較 (被験者 A，被験者 C)

4.5　考察

一方同じ被験者 A と被験者 C がオーケストレーション定義エディタを使用して同じ内容のテンプレートファイルを構築した場合，図 4.3 のような結果が出た．また，構築する内容が似ている (Ⅲ)，(Ⅳ)，(Ⅴ) を比較した場合図 4.4 のようになった．

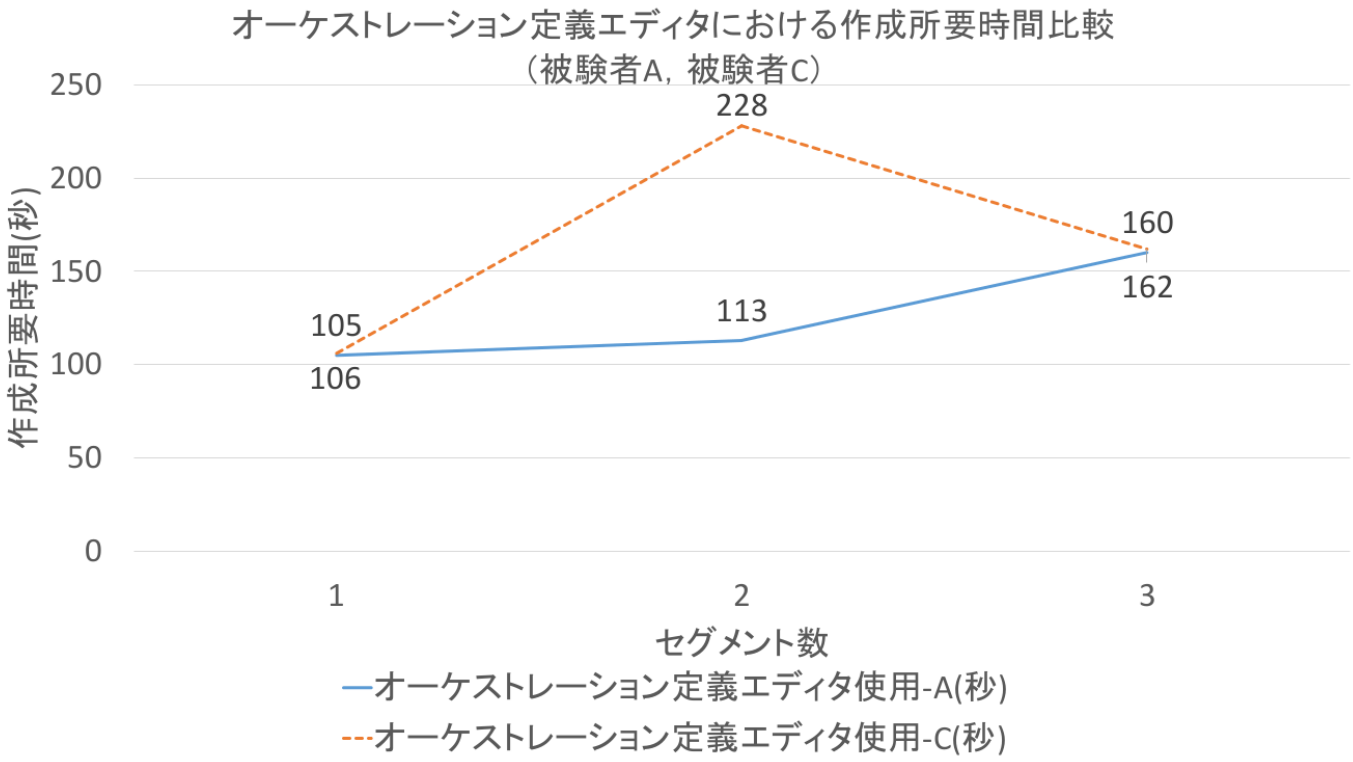

図 4.3　オーケストレーション定義エディタにおける所要時間比較 (被験者 A，被験者 C)

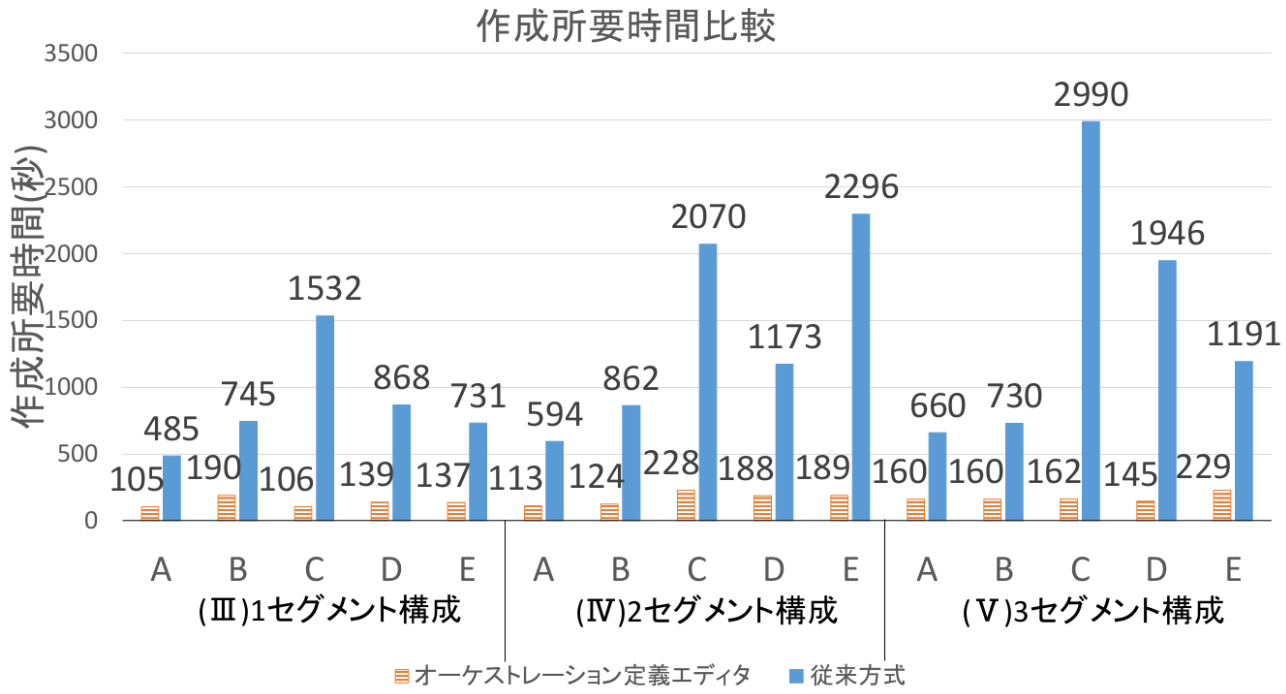

図 4.4　作成所要時間比較

従来方式ではテンプレートファイルを作成する場合手動入力で全ての項目を入力しなければならない．入力する際はテンプレートファイル独自の書式に注意しながら入力しなければならず，事前の学習で内容を把握しきっていなければ正確に入力することは困難である．本評価実験において従来方式の事前の学習時間は最長でも 540 秒であり，被験者たちは 10 分に満たない学習時間ではテンプレートファイル独自の書式について満足に理解できなかったと考えられる．しかし，従来方式でのテンプレートファイル作成回数が増える毎に従来方式の手動入力に慣れてくる被験者も現れ，システム構成の内容が複雑になっても所要時間の増加幅が少なくなっていった被験者もいた．ただし従来方式の手動入力に慣れることができなかった被験者はシステム構成の内容が複雑になるにつれて所要時間がより多く増加していった．

以上のことから，従来方式でのテンプレートファイル作成所要時間には個人差が生じやすく，なおかつ作成所要時間は膨大になりがちであるといえる．一方オーケストレーション定義エディタを使用することにより作成所要時間の個人差を無くし，作成所要時間そのものを

大幅に削減できる．よって作成所要時間の面でオーケストレーション定義エディタの有用性を示せたと考えられる．

4.5.3　エラー発生回数について

表 4.2 と**??**のとおり，従来方式では，まだテンプレートファイル記述経験の浅いうちに作成したものである (I) 作成時のエラー発生回数が最も多い．その後被験者がテンプレートファイル記述に慣れ，エラー発生回数は減少するが，(Ⅳ) においてインスタンス数だけでなくルーター（ネットワーク）数を増加，つまり新たな追加記述要素が加わることでエラー発生回数は増加した．被験者 5 人が従来方式にて発生させたエラー回数について図 4.5 に示す．

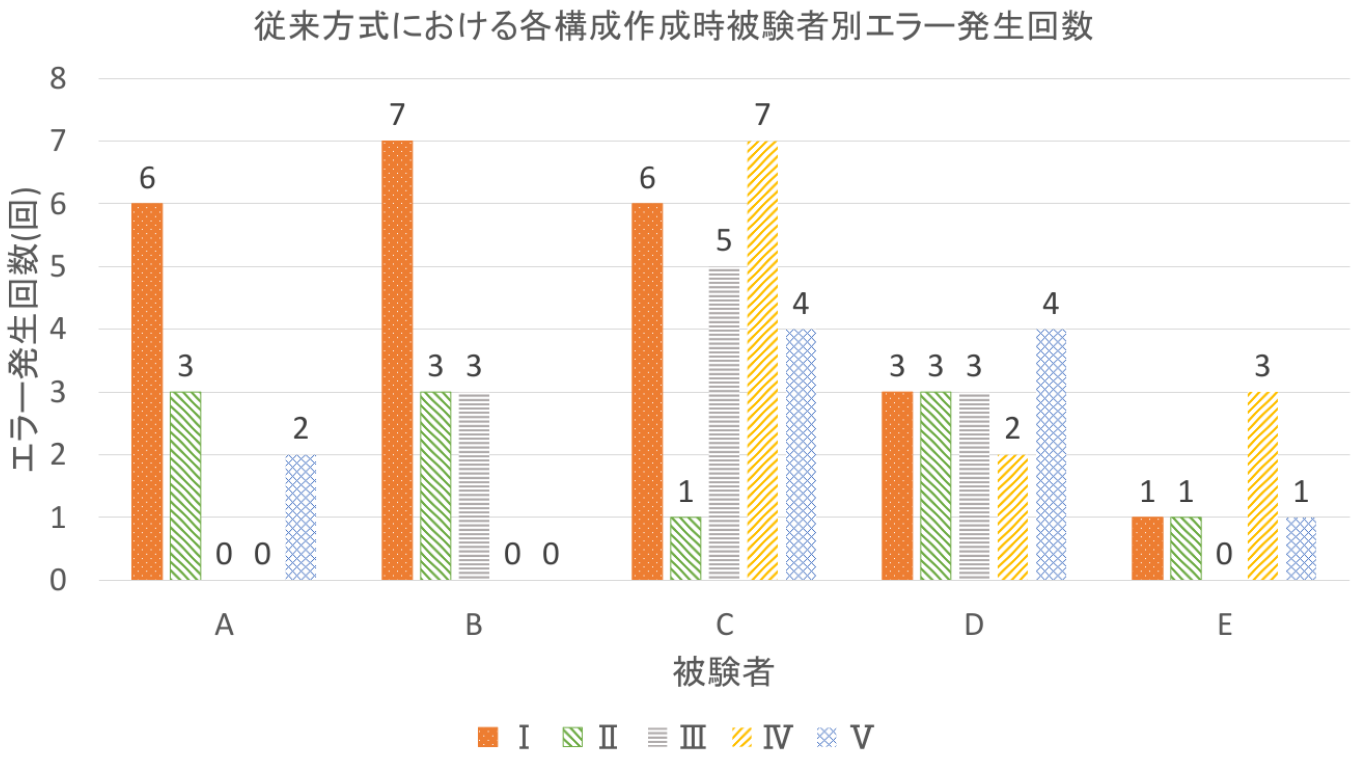

図 4.5　従来方式における各構成作成時被験者別エラー発生回数

一方オーケストレーション定義エディタを使用した際は表**??**のとおりテンプレートファイル作成時にエラーは発生しなかった．これは，入力される内容が決まっている項目は予め文字列データとして保存，出力時に所定の位置へ出力をしており，尚且つテンプレートファ

イル毎に入力内容が異なるものに関してはプルダウンメニューを用いて選択肢を用意し利用者に選択させ決定後にテンプレートファイルに出力しているので，利用者が記述内容をミスし得ない状態だからである．

よって，利用者はテンプレートファイルに関する専門知識を意識すること無く容易且つ正確にテンプレートファイルを作成できるといえる．

第 5 章

おわりに

5.1 研究のまとめ

本研究では，OpenStack 内コンポーネントである Heat を使用したシステム定義をテンプレートファイルに記述するうえで問題点となっていた「膨大なテキスト記述量」と,「構成情報把握のしづらさ」,「テンプレートファイルの書式の複雑さ」を解決するためにオーケストレーション定義エディタを提案し，実現した.

操作インターフェイスを GUI にすることにより，構築中でも構成内容を把握できるようになった．複雑な書式でテキストのみで入力されたテンプレートファイルの構成内容を，GUI を用いて可視化させることができた．これにより「構成情報の把握しづらさ」を解決した.

オーケストレーション定義エディタでは，構築毎に入力内容が変わらない項目は予め文字列データとして保管しておく．これらの内容は出力時に自動で適切なデータだけをテンプレートファイルへ出力する．一方構築毎に内容が変わる項目は専用のデータ保管用 Class を用意し，利用者から入力された内容を保管する．尚，保管用 Class へデータを保存する項目は詳細設定画面とよばれる画面でプルダウンメニューにより選択肢を提供される．その後出力時にテンプレートファイルへ出力される．これらにより，従来テキストで記述していた内容をほとんどプルダウンメニューによる選択と自動入力で記述できるようになり，利用者はテンプレートファイルの複雑な書式を意識すること無く，プルダウンメニューを選択するだけで項目の入力が可能となった．これにより問題点である「テンプレートファイルの書式の複雑さ」と「膨大なテキスト記述量」を解決できた.

第４章の評価実験により，オーケストレーション定義エディタを用いることによりテンプレートファイル作成時のエラー発生回数を激減させることができ，自動入力等の補助によりテンプレートファイル作成所要時間も大幅に減少させることができた．また，オーケストレーション定義エディタを使用してテンプレートファイルを作成した際，５人の被験者の各作成所要時間に大きな差が開いていなかった．以上のことから，利用者の能力に依存すること無く短時間かつ容易で正確にテンプレートファイルを構築し，仮想環境を構築できることがわかった．

つまり，オーケストレーション定義エディタの実現によりテンプレートファイルに関する専門的な知識を意識させること無く，利用者の能力に左右されること無く短時間かつ正確，そして容易にテンプレートファイルを作成可能であることが示せた．

また，オーケストレーション定義エディタはテンプレートファイルへの記述補助エディタであるため，基本的に記述させる内容を追加することは容易である．つまり，Heat の機能内であれば機能や動作を追加できるので，拡張性は高いといえる．

5.2　今後の課題

オーケストレーション定義エディタの問題点として，起動するインスタンスを追加する際，１つずつしか追加できない点があげられる．これでは，仮に同じ内容のインスタンスを複数個起動させようとした場合でも，起動するインスタンスの数だけ同じ内容の記述をそれぞれの詳細設定画面で繰り返さなければならない．これを改善する案として詳細設定画面に「起動させるインスタンス数」という項目を追加し，同時に起動させるインスタンス数を指定できるようにする方法が挙げられる．この機能を追加することにより，同じ内容のインスタンスを複数個起動させたい場合であれば現時点のオーケストレーション定義エディタでの操作よりも操作回数を減少させることができ，テンプレートファイル作成所要時間も減少させることができる．

謝辞

本研究を進めるにあたり，研究指導及び論文審査の主査として，横山和俊教授には多くのご指導を頂きました．また，研究室での活動だけではなく就職活動や社会人としての心構えについて数多くのご指導を頂きました．心より感謝致します．本研究の副査をお引き受けいただき，数々のご質問，ご指摘を頂きました情報学群 岩田誠教授と鵜川始陽准教授に心より感謝致します．

分散処理 OS 研究室の同期である大崎康平君，別役速斗君，小池主馬君には，日頃の勉学をはじめ各種研究室内イベントや卒業研究でたくさんお世話になりました．ありがとうございました．また，分散処理 OS 研究室の先輩である畑翔太氏には勉学や研究内容について数多くの手助けをしていただきました．ありがとうございました．後輩である松岡亨一君，川村郁哉君，手塚詞央里さん，濱田哲郎君，舩越勇樹君，本研究の評価実験への協力をはじめとする，日頃の様々な場面において力になってくれて本当にありがとうございました．

最後に，様々な側面から私を見守り，支えてくださった家族に心から感謝します．

参考文献

[1] 総務省　平成２７年度版 情報通信白書 第５章第２節「ICT 産業のグローバルトレンド」 http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/pdf/index.html

[2] IBM クラウド・コンピューティング:IaaS とは-Japan http://www.ibm.com/cloud-computing/jp/ja/what-is-iaas.html

[3] OpenStack https://www.openstack.org

[4] OpenStackCloudSoftware Heat Documentation http://docs.openstack.org/developer/heat/index.html