OpenStack環境でのオーケストレーション定義のための GUI エディタの実現と評価

松岡 亨一[1]　川口 貴大[1]　横山 和俊[1]

概要：近年，IT リソースを迅速に確保する目的やシステムのコストを抑制するため，IaaS が普及している．この IaaS 基盤を構築するソフトウェアに OpenStack がある．OpenStack では，複数のインスタンス，ネットワークを一括構築するオーケストレーション機能が Heat により定義されている．しかし，オーケストレーションを定義するテンプレートファイルは記述に要する時間が大きいことや、記述ミスが起きやすいなどの問題がある．そこで、本研究では，GUI を用いたエディタを実現する．また，評価により仮想環境定義に関する時間コスト及び学習コスト削減できることを確認した．

1. はじめに

OpenStack は OpenStackCommunity によって開発されたオープンソースの IaaS 基盤ソフトウェアである [1]．OpenStack では，複数のインスタンスやネットワークを一括構築するオーケストレーション機能が Heat により提供されている．しかし，オーケストレーションを定義するテンプレートファイルは，記述に時間がかかり，また，記述ミスが起きやすい問題がある．そこで GUI を用いたオーケストレーション定義エディタを実現し，短時間かつ容易に仮想環境を定義可能とした．本稿ではその実現内容と評価について述べる．

2. GUIエディタの提案と実装

2.1 問題点

従来の Heat では，テンプレートファイルをテキストエディタで記述する．そのため，下記のような 3 つの問題が生じる．

1) 構成情報の把握が難しい
   構成情報を全てテキストで記述しているため，全体の構成をテンプレートファイルからは把握しづらい．
2) 書式の学習に時間がかかる
   テンプレートファイルの書式が複雑である．例えば，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．文中のインデントの深さで入力内容を区別する等複雑な書式が多い．このため，書式の学習に時間がかかる．
3) テキストの作成に時間がかかる
   仮想環境のシステム構成をテキストで記述するため，作成に時間がかかる．

2.2 解決方針

問題点を解決する方法として，仮想環境のシステム構成を定義する GUI エディタを検討する．GUI エディタを実現する方針を以下に示す．

1) 構築中のシステム構成全体を GUI により表示する．つまり，インスタンスやネットワークの構成を一目で確認できる機能を提供する．
2) 書式を学習する必要性を排除する．具体的には，構成要素の定義に必要な項目のみを入力すればよく，書式に関する記述を不要にする機能を提供する．
3) テキスト入力を極力排除する．つまり，記述内容が決まっている項目については，プルダウンメニューにより選択する機能を実現する．

2.3 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の概略図を図 1 に示す．実現したエディタは，大きく 2 種類の画面がある．構成確認画面は，全体構成を示す画面である．作成者は，この画面でインスタンスやネットワークなどの構成要素を追加する．追加した構成要素は，詳細設定画面でその情報を定義する．例えば，インスタンスの詳細設定では，ディスクイメージや接続するネットワークを定義する．作成者は，システム構成の定義が終わると，出力ボタンを押下し，テキストのテンプレートファイルを出力する．

[1] 高知工科大学

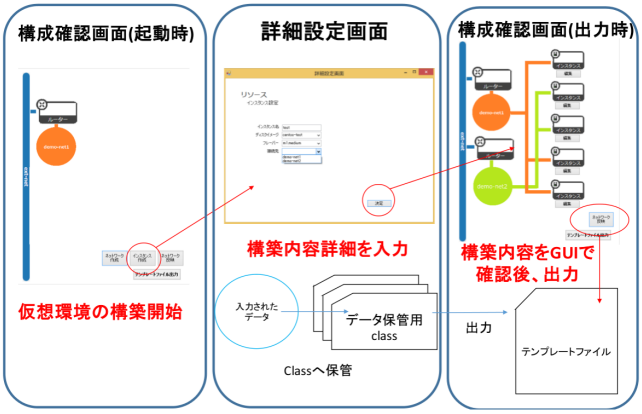

図 1　オーケストレーション定義エディタの概略

(1) 構成確認画面

構成確認画面を用いることにより，1 つ目の解決方針であるシステムの全体構成の可視化を定義している．図 2 は，3 つのサブネットと 5 つのインスタンスにより構成されるシステムである．構成確認画面により demo-net1 には 2 つのインスタンス，demo-net2 にも同様に 2 つのインスタンス，demo-net3 には 1 つのインスタンスが接続されていることが確認できる．「ネットワーク反映」を押下することで設定した情報を反映し描画を行うため，構成情報を確認しながら仮想環境を構築することが可能になる．

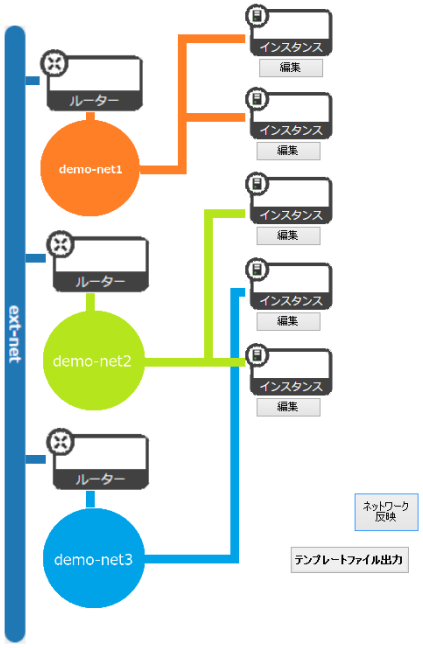

図 2　仮想環境の構成確認画面

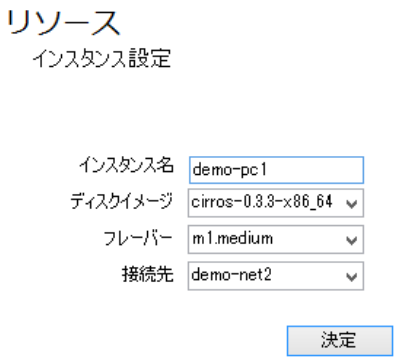

図 3　インスタンスの詳細設定画面

(2) 詳細設定画面

図 3 にインスタンスの詳細設定画面を示す．詳細設定画面では，インスタンス名，ディスクイメージ，フレーバー，接続先についての入力のみをすればよく，テンプレートの書式に関する知識は画面には表示せず，また入力も求めない．これにより，解決方針の 2 つ目であるテンプレート独自の書式を理解しなくても，作成者が情報を入力できることを可能にしている．また，3 つ目の解決方針である予め入力する内容が決まっている項目を対象に，プルダウンメニューにより選択を可能にしている．図 3 の例では，ディスクイメージ，フレーバー，接続先については，定義できる内容が決まっているためプルダウンメニューによる選択で入力を可能としている．

3. 評価

3.1 評価項目

1 章で示した Heat の問題点を解決できたかを検証する．評価項目を表 1 に示す．また，OpenStack の基礎知識を持っている学生 5 名により評価を実施した．

表 1　評価項目

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

学習時間は，各方式でテンプレートファイルを記述するために必要な前提知識を習得するためにかかった時間である．作成時間は，各方式でテンプレートファイルを作成するのにかかった時間である，作成インスタンス数とルータ

数を変化させて評価する．最後に，エラー回数は，各方式で作成したテンプレートファイルに誤りが混入した回数である．つまり，被験者が Heat にテンプレートファイルを読み込ませるときにエラーが発生し，やり直した回数を計測している．

表 2 は評価実験で作成したシステム構成の詳細を示している．I〜V までの 5 種類の仮想環境を対象にテンプレートファイルを作成した．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作成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．

表 2　作成するシステム構成

番号	セグメント数	インスタンス数
I	1	1
II	1	3
III	1	5
IV	2	5
V	3	5

3.2　評価と考察

3.2.1　学習時間

学習時間の評価結果を図 4 に示す．従来方式とオーケストレーション定義エディタ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短いことがわかる．これは，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自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である．一方，オーケストレーション定義エディタは，ボタン，プルダウンメニュー，及び 1 箇所のみの手動入力項目に関する説明があるだけで

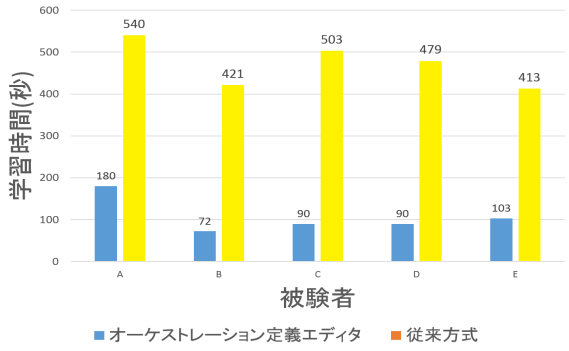

図 4　学習時間の評価結果

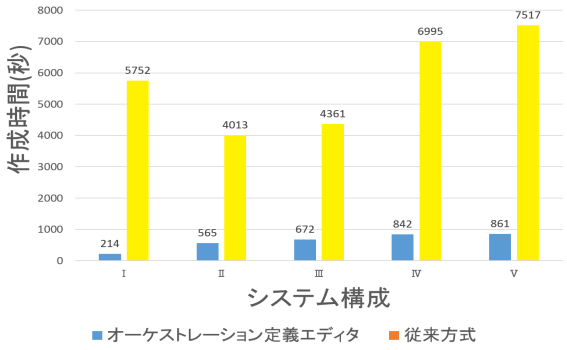

図 5　作成時間の評価結果

ある．つまり，学習すべき項目は従来方式に比べて大幅に少ないため，学習時間が短い．

3.2.2　作成時間

作成時間の評価結果の総計を図 5 に示す．3 つの評価項目のうち，最も大きな差となったのが作成時間である．従来方式では，5 人の被験者の作成時間は，4013 秒〜7517 秒かかっている．一方，オーケストレーション定義エディタを用いた場合，214 秒〜861 秒で作成できている．つまり，約 1/10 程度の時間でテンプレートファイルが作成できており，実現したエディタの有効性が確認できる．

従来方式で作成に時間がかかる理由は，記述するテキストの分量が多いのに加え，書式の確認に時間がかかるからである．今回の評価では，事前の学習に 413 秒〜540 秒の時間をかけている．しかし，事前の学習だけでは，テンプレートファイルの書式を完全に把握することができず，作成途中で書式を確認しながら記述を行う被験者が多かった．つまり，独自の書式を使いこなすためには，資料を読むだけでなく，実際に記述する経験が必要であった．実際，従来方式でテンプレートファイルの作成回数が増えると，テキストによる手動作成の時間が短縮される被験者もいた．しかし，オーケストレーション定義エディタを使った場合の短縮とは比べられない程度の短縮幅であった．

3.2.3　エラー発生回数

システム構成ごとのテンプレートファイル作成時エラー回数の総計を表 3 に示す．オーケストレーション定義エディタを使用したとき従来方式に対して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ないためである．

4.　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記述する問題点を解決するために

表 3 システム構成別エラー発生回数総計

番号	エディタ (回)	従来方式 (回)
I	0	23
II	0	11
III	0	11
IV	0	12
V	0	11

オーケストレーション定義エディタを実現した．また，従来方式よりテンプレートファイル作成時間を大幅短縮できることを確認した．

参考文献

[1] OpenStack （入手先 ⟨https://www.openstack.org/⟩） (2016.07.22).