コンテナ型仮想化によるモデリング教育向けコンパイルサーバの設計

Fundamental Design for Educational Compilation Server with Container-based Virtualization Method

中野 敬久[*1], 香山 瑞恵[*2], 永井 孝[*3], 原 舜弥[*1]
Takahisa NAKANO[*1], Mizue KAYAMA[*2], Takashi NAGAI[*3], Shunya HARA[*1]
[*1] 信州大学大学院総合理工学研究科
[*1]Graduate School of Science and Technology, Shinshu University
[*2] 信州大学工学部
[*2]Faculty of Engineering, Shinshu University
[*3] ものつくり大学総合機械学科
[*3] Department of Mechanical and Production Engineering, Institute of Technologists
Email: 19w2085a@shinshu-u.ac.jp

あらまし：本研究の目的は，モデリング教育支援環境の改良である．ここではモデリング学習に用いるモデル駆動開発ツールのコンパイルサーバを対象とし，Docker を用いてコンパイル環境のコンテナ型仮想化を行い、モデル駆動開発のターゲットロボットの種類ごとに独立したコンパイル環境を現行の学習支援環境に組み込むための設計を試みた．本稿では既存環境の概要を述べ，改良に際しての課題を示し，新たな環境の設計成果を述べる．
キーワード：モデリング教育，モデル駆動開発，コンパイルサーバ，コンテナ，仮想化，コンパイル環境

1. はじめに

大学の情報系学科や情報系の専門学校など，高等教育における専門情報教育において，モデリング能力を習得することで「捉える力」や「考える力」、「表現する力」などを養成することがカリキュラム標準として指向されるなど，対象をモデル化する能力の育成が重要視されている [(1)]．

モデル化する能力を習得するにあたり，モデル駆動開発(Model Driven Development．以下，MDD)ツールを用いた教育事例では，clooca [(2)] や astah が利用されている．学習者はこれらのツールを利用して、与えられた課題に合わせてモデルを記述する．この学習で用いられている既存の MDD ツール群には現状運用面での問題点がある．本稿では，その問題点を整理し，それらを解決した新環境を提案する．

2. 既存の MDD ツール群

2.1 概要

MDD を用いた学習環境の概要図を図 1 に示す．MDD ツール群はモデルエディタ，モデルコンパイラ，コンパイルサーバからなる． 学習者は clooca 等のツールのモデルエディタ機能を用いてモデル図を記述する．続いてモデルコンパイラによりモデル図がソースコードに変換される． ソースコードを与えられたコンパイルサーバは実行コードを生成し，その実行コードは学習者の使用端末にダウンロードされる．学習者はこの実行コードを開発対象であるロボット等に転送し挙動を確かめ，モデル図の妥当性を検証する．

2.2 コンパイルサーバの動作

既存のコンパイルサーバの動作概要図を図 2 に示す．既存のコンパイルサーバには，モデルコンパイラが生成したソースコードを受け付けるアプリケーション(図 2 左下 front)，負荷分散のために builder(後述)にコンパイル要求を振り分けるアプリケーション(図 2 下部 cheif)，ソースコードから実行コードへ

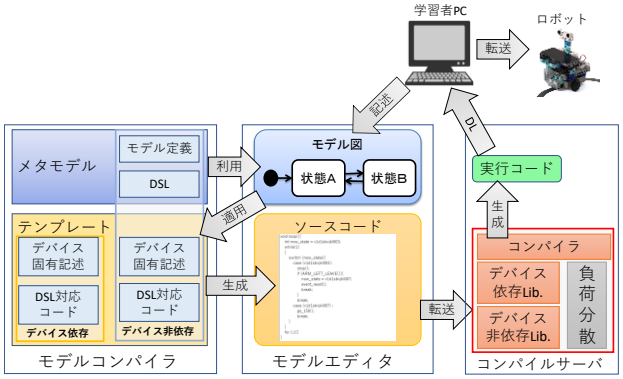

図 1: MDD を用いた学習環境の概要図

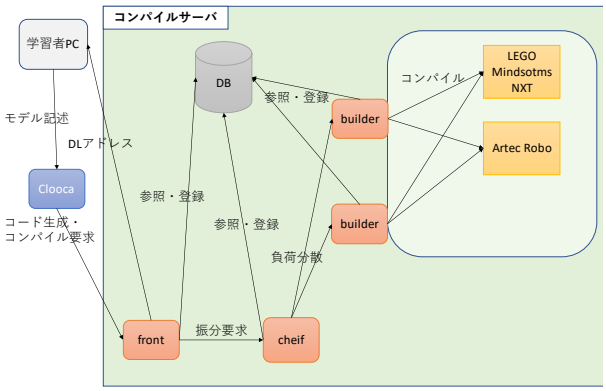

図 2: 既存のコンパイルサーバの動作概要図

の変換を命令するアプリケーション(図 2 中央 builder)の 3 種が実装されている．各アプリケーションはコンパイルに用いられる各データをデータベースに登録，および参照し，順次コンパイル要求を処理していく．ソースファイルをコンパイルサーバが受け取ると，これらのアプリケーションが動作して実行コードが記述された実行ファイルが学習者へ渡される仕組みとなっている．

2.3 コンパイルサーバの問題点

これまでこのツール群を用いた MDD 学習では小型ロボットを対象とした開発が取り上げられてきた．学習に用いられるロボットの種類は複数あるが，それらのコンパイル環境は全てサーバ上に直接展開されている．そのため現在は異なるコンパイル環境を必要とするロボットのコンパイル環境を共存させるためにツールチェーンを都度登録する必要があった．

また，コンパイル環境維持のための OS やライブラリのアップデートに制約が生じるなど，コンパイル用アプリケーション・サーバの環境を容易に変更することができない．

3. 提案環境の設計

3.1 提案環境の概要

2.3 の問題点を解決するために，コンパイル環境をコンテナ型仮想化（以下，コンテナ化）することにより，アプリケーション・サーバから独立させ，複数の異なるコンパイル環境を共存させることとした．

個々のロボット用のコンパイル環境をコンテナ化し，コンテナ内にコンパイラやライブラリなどのコンパイルツールを設置することで，コンテナに対してホストとなるコンパイルサーバ(以下，ホスト)の OS やライブラリのバージョンといったホスト環境に依存しない，独立したコンパイル環境が得られる．

3.2 コンテナ化

コンテナ化とは仮想化技術の手法の 1 つである．コンテナ化には Docker [3] を用いる．Docker とは Docker 社が開発しているコンテナ型の Linux 仮想環境を提供するオープンソースソフトウェアである．Docker はホスト OS のカーネルを共有し，ディストリビューション以上の層のみを仮想化している [4]．このため，コンテナ化は，ハイパーバイザ型の仮想環境と比較して，環境の立ち上げや削除を素早く行える．また，オーバーヘッドが少なく，環境を複数立ち上げた場合のサーバにかかる負荷が少ない．そのため，大量の処理を同時に実行できる．さらにコンテナは動的なスケーリングができるため効率的な負荷分散を行うことが可能になる．

3.3 既存環境へのコンテナ実装構成案

提案環境のコンパイルサーバの動作概要図を図 3 に示す．既存環境でのコンパイルは，図 2 中 builder がデータベースからソースコードをホスト上のディレクトリにコピーし，シェルコマンドを実行するこ

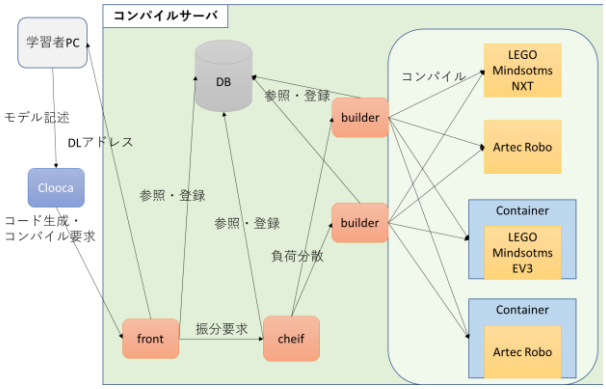

図 3: 提案環境のコンパイルサーバの動作概要図

とで実現していた．そのターゲットロボット判定後の動作を，新環境ではコンパイルをコンテナ内で行うため，図 3 中 builder を以下の動作となるように変更した．

1. ロボットに対応したコンテナ（図 3 右下 Container）を起動させる．
2. データベースからソースファイルを含む zip ファイルをコンテナとの共有ディレクトリにコピーする．
3. 2.とは別の共有ディレクトリに出力された実行ファイルとエラーログをデータベースに登録する．

なお，コンテナ内では，共有ディレクトリに zip ファイルが置かれると，コンパイルを実行し，実行ファイルを指定した共有ディレクトリに出力するスクリプトが実行される．

4. おわりに

本研究の目的は，MDD を用いたモデリング教育の支援環境の改良である．本稿では，MDD ツール群でのコンパイルサーバを対象に，複数ロボットの開発環境共存法の提案とその設計について述べた．

今後は本稿の設計に基づいて実装を行い，新環境による学習を実施していく．また，より多くのコンパイル要求を同時に処理するため，効率的な負荷分散を意識した改良を図る．

参考文献

(1) 情報処理学会，“カリキュラム標準 J17 カリキュラム標準ソフトウェアエンジニアリング領域(SE) J17-SE-報告書-20180319,”（2018）
(2) Hiya, S. et al. “Clooca : Web based tool for domain specific modeling,” CEUR Workshop Proceedings Volume 1115, Pages 31-35, （2013）
(3) Docker Inc., “Docker: Enterprise Container Platform” https://www.docker.com/, （2019/06/17 参照）
(4) Docker Inc.“What is a Container” https://www.docker.com/resources/what-container, （2019/06/17 参照）