3N-7

コンポーネントベースソフトウェア開発のための形式的仕様記述と検証に関する研究

富田 隆彦† 高田 眞吾† 飯島 正† 土居 範久‡†
† 慶應義塾大学大学院理工学研究科 ‡ 中央大学理工学部

1 はじめに

近年，形式的仕様記述言語の分野では，ソフトウェアの分散化や複雑化に対するアプローチとして，状態に基づく言語とプロセス代数の統合が注目されている．

本研究では，コンポーネントベースソフトウェア開発でより高い信頼性と効率性を実現することを目的として，統合手法のひとつ Object-Z/CSP に注目し，アーキテクチャ記述言語の概念を取り入れた仕様記述と検証の方法を提案する．

2 コンポーネントベースソフトウェア開発

コンポーネントベースソフトウェア開発 (CBSD) では，コンポーネント*を組み合わせることによって目的とするソフトウェアを構築する．このため構造的な側面からソフトウェアを捉えることが重要となる．アーキテクチャ記述言語 (ADL) は，この構造的な側面からソフトウェアを記述する言語である．特に，Wright[1] に代表される形式的な言語を記述に利用した ADL は，コンポーネントの振舞や相互作用を厳密に表現することができる．ソフトウェアの分散化や並行化が進むなか，不正な振舞に対する形式的な検証が可能となる点は有効である．しかし，次に示す点で十分ではない．

- 不正な振舞の検出を行うことは可能だが，要求された機能を実現しているかについての検証はできない．
- 起こり得る振舞を網羅することが必要となるため，規模が大きく複雑な処理を行なうコンポーネントの場合に記述が非常に煩雑になる．また，データ構造をともなう処理の記述にも不向きである．

これら信頼性や効率性に関わる問題点への対応として，本研究では，次に示すような形式的仕様記述を利用するアプローチをとる．

- 開発対象となるソフトウェアへの要求を，形式的な言語を用いて明確に仕様化する．この要求に対応する具体的な仕様を，コンポーネントの仕様の組み合わせとして表現する．そして，具体的な仕様が要求を満足することを示す．
- データ構造や複雑な逐次的処理の記述に適した形式的仕様記述言語をコンポーネントの仕様記述に用いる．

"Formal Specification and Verification for Component Based Software Development", Takahiko Tomida, Shingo Takada, Tadashi Iijima and Norihisa Doi, Keio University

*ここでコンポーネントは，ブラックボックス型のソフトウェア部品であるとする．

3 Object-Z/CSP

本研究では，2 章で述べたアプローチに適した形式的仕様記述言語として，Object-Z/CSP[4] に注目する．

Object-Z/CSP は，Object-Z と CSP (Communicating Sequential Processes) を組み合わせてソフトウェアの仕様を形式的に記述するアプローチである．性質の異なる言語を組み合わせることで，複雑な逐次データ処理の記述とプロセス間の相互作用の記述を両立することができる．

Object-Z のクラスは，CSP の標準的な意味モデルである failures/divergences semantics によって意味を与えられている．このため，Object-Z のクラスを CSP のプロセスとして扱うことができる．

4 提案

本研究では，Object-Z/CSP で ADL の概念を導入した仕様の記述と詳細化の方法を提案する．また，詳細化を行なう際の仕様の検証について述べる．

4.1 提案の概要

ADL は，コンポーネントやコネクタという概念を用いることで，ソフトウェアを構造的側面から適切に記述することができる．そこで，ADL の主要な構成要素であるコンポーネントとコネクタを，Object-Z/CSP の枠組内で定義する．また，Wright など一部の ADL で定義されているコンフィグレーションについても，構成要素として導入する．

抽象的な仕様からの詳細化に関しては，具体的な仕様を新たに導出していく従来の方法ではなく，ADL から取り入れた要素の組み合わせによって具体的な仕様を構成していく方法を提案する．

これによって，2 章で述べた問題点を克服し，より信頼性と効率性の高い CBSD を実現することができる．

4.2 仕様の構成要素

提案方法において，仕様の記述は，次に示す 3 つの要素から構成される．

コンポーネント 論理的なモジュール単位としてのコンポーネントを特に**論理コンポーネント**と呼ぶ．また，実装を伴う既存のコンポーネントの機能を表現した仕様を，特に**既存コンポーネント**と呼ぶ．本研究において，コンポーネントとは，論理コンポーネントと既存コンポーネントのいずれかを表す．

コンポーネントは Object-Z のクラスとして記述する．

コネクタ ADL と同様にコンポーネント間の相互作用を記述するための要素である．

コンポーネントをどのように組み合せるかは，生起されるイベントの時間的な順序関係から規定する．コネクタは CSP のプロセスとして記述する．

コンフィグレーション 複数のコンポーネントとコネクタによって構成される，複合型のコンポーネントである．コンフィグレーション内部の構成を隠蔽し，外部へのインタフェースのみを公開することで，コンポーネントと同様に扱うことができる．これにより，仕様を段階的に構成する．

なお，コンフィグレーションについては，次のように定義する．Object-Z/CSP によって記述されるコンポーネントを $Comp_1$，$Comp_2$, ..., $Comp_n$，CSP によって記述されるコネクタを $Conn$，外部から観測可能なコンフィグレーションのチャネルの集合を $extl\_chans$，隠蔽される内部チャネルの集合を $intl\_chans$ とする．このとき，コンフィグレーション $Conf$ は，

$$Conf = Conn||(Comp_1||\ldots||Comp_n)\backslash intl\_chans$$

となる．但し $extl\_chans$，$intl\_chans$ はそれぞれ，

$$\begin{aligned} extl\_chans &= \alpha Conf \\ intl\_chans &= \alpha Comp_1 \cup \alpha Comp_2 \cup \ldots \cup \alpha Comp_n \end{aligned}$$

である．ここで || は並行動作，\ は隠蔽の演算子である．αP はプロセス P のアルファベットを示す．

コネクタについては，コンポーネントをどのように組み合わせるかを規定すると共に，コンフィグレーション外部との相互作用についても記述する必要がある．このため，コネクタ $Conn$ のアルファベットは，

$$\alpha Conn = extl\_chans \cup intl\_chans$$

となる．

4.3 詳細化

詳細化とは，抽象的な仕様をより具体的な仕様へと洗練していく作業である．

本研究では，コンポーネントの組み合わせによるソフトウェアの構成過程を詳細化として扱う．目的とする機能を記述した論理コンポーネントを**抽象的な仕様**，コンポーネントとコネクタによって構成したコンフィグレーションを (抽象的な仕様を満たす) **具体的な仕様**として位置づける．詳細化は，コンフィグレーションの構成要素に論理コンポーネントが含まれている限り，段階的に行うことができる．勿論，論理コンポーネントをあえてコンフィグレーションの構成要素に残し，新たに実装を行なうコンポーネントの仕様として扱うこともできる．

提案する方法での詳細化の概念を図 1 に示す．図中では，論理コンポーネントを破線の矩形，既存コンポーネントを実線の矩形，コネクタを丸角の矩形で表す．また，コネクタとコンポーネントを結ぶ細い実線はチャネル，論理コンポーネントからコンフィグレーションへのびる太い破線矢印は，詳細化を表現している．

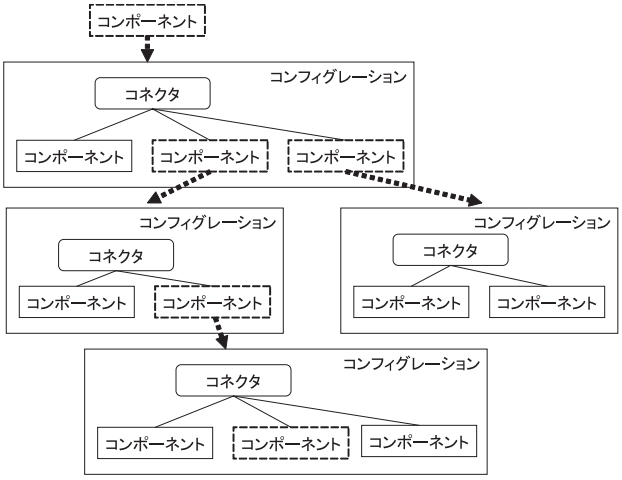

図 1: 提案手法での詳細化の概念図

4.4 検証

本研究の提案する方法では，詳細化の前後で仕様の構造が大きく変化する．このため，具体的な仕様が抽象的な仕様を満足するかについて，厳密な意味で検証を行なうことは難しい．そこで，各構成要素を CSP のプロセスとみなし，詳細化を行った場合の前後の仕様を，振舞の面から比較する方法をとる．検証時は，記述した仕様のうち Object-Z による記述部分を CSP の記述に変換する．検証系には FDR[2] を利用する．記述の変換については，Kassel らの研究 [3] を参考とした．

5 まとめ

本研究では，CBSD でより高い信頼性と効率性を実現するために，Object-Z/CSP で ADL の概念を取り入れた仕様記述と詳細化および検証を行なう方法について提案した．

謝辞

本研究は，文部科学省科学技術振興調整費「環境情報獲得のための高信頼性ソフトウェアに関する研究」の支援による．

参考文献

[1] R. Allen, D. Garlan: “Formal basis for Architectural Connection”, ACM Trans. on Software Engineering and Methodology, Vol.6, No.3, pp.213-249, 1997.

[2] Formal Systems (Europe) Ltd, “Failures-Divergences Refinement: FDR2 User Manual”, *http://www.fsel.com/*, 1997.

[3] G. Kassel, G. Smith: “Model Checking Object-Z Classes: Some Experiments with FDR”, Proc. of Asia-Pacific Software Engineering Conference, pp.445-452, 2001.

[4] G. Smith: “A Semantic Integration of Object-Z and CSP for the Specification of Concurrent Systems”, Proc. of Intl. Symp. of Formal Methods Europe, pp.62-81, 1997.