ソフトウェア工学 115－5
（1997．7．31）

ソフトウェアコマースのためのカタログ記述言語 SCL

青山 幹雄*
*新潟工科大学 情報電子工学科
〒945-11 柏崎市藤橋 1719
mikio@iee.niit.ac.jp

山下 利夫†　小堀 慎介†
†(株)日本総合研究所 先端情報技術センター
〒102 東京都千代田区一番町 16
{yamasita, kobori}@tyo.aitec.jri.co.jp

ソフトウェアパッケージやコンポーネント（部品）をネットワーク上で電子取引するソフトウェアコマースのために仕様とカタログ情報を記述する SCL(Software specification and Commerce Language)を提案する．ソフトウェアは電子的に流通が可能であるので，仕様と取引の情報が一体として表現できることが望ましいと考え，SCL は HTML の拡張として定義した．1つの部品のカタログが Web の1ホームページとして段階的に表現できる．さらに，部品選択を支援するために，コンテキスト表現として部品を取り巻く構造をデザインパターンで記述し，動的挙動をユースケースのシナリオで記述できる表現をとった．さらに，部品を Web 上で試行できるプレイアブルの概念を提案する．

SCL: A Software Specification and Commerce Language

Mikio Aoyama*
*Dep. of Information and Electronics Eng.
Niigata Institute of Technology
1719 Fujihashi, Kashiwazaki, 945-11
mikio@iee.niit.ac.jp

Toshio Yamashita† and Shinsuke Kobori†
†Advanced Information Technology Center
The Japan Research Institute, Limited
16 Ichibancho, Chiyoda-ku, Tokyo 102
{yamasita, kobori}@tyo.aitec.jri.co.jp

This article proposes *SCL(Software specification and Commerce Language)* which is intended to present the specification and commerce information of software components and packages. Since software can be distributed through the Internet, it is desirable to represent the specification and commerce information in a single language. Thus, SCL is designed as an extension of HTML so that SCL works for a Web-based on-line catalogue of software components. To provide a better understanding on software components, SCL can specify the context and dynamic behavior of software components with design pattern and Use-Case scenario. Furthermore, we proposes *playable* concept so that users can play with software components on the Web.

1．まえがき

ソフトウェア部品やパッケージをネットワーク上で電子取引を行うソフトウェアコマースのための記述言語 SCL (Software specification and Commerce Language)を提案する．ソフトウェアの仕様と取引情報を統合している点で新しい枠組みを提供する．筆者らがソフトウェア CALS (Continuous Acquisition and Life-cycle Support)の次世代実証実験で開発しているコンポーネントブローカを実現する言語である．

2．問題点とアプローチ

2.1 問題点

コンポーネントウェア，分散オブジェクト環境の発展に伴い，ソフトウェア部品の利用が広がっている[Aoya96a]．ソフトウェアはネットワーク上で流通できることから，Web 上でソフトウェアの販売も行われている[Aoya96b]．

一方，ソフトウェア CALS コンソーシアム[Naga96]の次世代実証グループでは，今後のソフトウェア開発の基盤技術としてコンポーネントウェアの研究，開発，実証を行っている[Aoya96b]．ここで，筆者らのグループは，ネットワーク上での部品の流通基盤としてコンポーネントブローカを開発している．ブローカ実現のためには，部品の仕様やベンダ情報などの様々な取引情報の表現が必要となっている．

しかし，CORBA-IDL [OMG95] や CDL (Component Description Language)[OMG97] あるいは幾つかのアーキテクチャ記述言語ADL (Architecture Description Language)[Garl97] は取引に関する情報は提供していない．本稿では，部品を中心にソフトウェアの仕様と取引を統合した表現言語を提案する．

2.2 表現の視点

部品などの記述には，次の 2 つの視点がある．

(1)部品利用者

(2)部品提供者

SCL は，取引のためのカタログとして使用することから，利用者の視点を中心とする記述を採用する．

2.3 記述言語の設計方針

(1)仕様と取引情報の統合

インタフェース仕様に加え取引のための情報を統合して記述可能とする．

(2)インタフェース記述(実装からの独立)

(3) ベンダ／テクノロジニュートラル

複数の de facto な部品標準(ActiveX/DCOM, Java Beans, CORBA)へマッピング可能なベンダ，テクノロジから独立した定義を可能とする．

(4) 複合部品の定義

複合部品とは複数の部品を組み合せて実現されている部品である．複数の複合部品が共通の部品を利用する場合があるので，部品の構成を定義できる必要がある．

(5) Web 上で表現

部品情報をインターネット上で提供，検索可能とするため，Web 上で部品を記述できるようにする．すなわち，

1 ホームページ＝ 1 部品仕様記述

となる．

(6) 意味情報(セマンティクス) のオープンな表現

部品を利用する上でその意味情報が重要であるので，インタフェースのシグネチャに加えて意味情報を表わせる仕組みを提供する．

(7)段階的で拡張可能な仕様表現

必要に応じて段階的に情報を提供，参照でき

る．これによって，多様な情報の提供と簡潔な記述という部品表現に対する相反する要件を満たす．ただし，現段階では一般に閉じた形式で定義することが困難であるので[Garl97]，必要に応じて追加できる拡張可能な仕様記述とする．

(8)プレイ可能(Playable)

部品を試行できる機構を提供する．ユーザの指示により動的挙動を視覚的に示すことである．ただし，この目的を果たせるなら，必ずしも実行する必要はないと考えている．

2.4 インタフェース仕様の表現方法

(1)仕様表現モデル

部品とそれを組み合せたソフトウェアのアーキテクチャの表現モデルとして，幾つか提案されている．大別して，次の2つのモデルがある．

(a)静的構造モデル

部品インタフェースとその静的な組み合せの構造としてモデル化する．

(b)動的挙動モデル

部品間の関係を部品間のメッセージ交換として動的挙動でモデル化する．

SCLでは，部品利用者の視点から静的構造モデルを採用し，インタフェースに動的挙動を表現できるよう拡張した．

さらに，静的構造の表現モデルとして，次の2つの方法がある．

1)部品とコネクタなどの部品間の関係を分離して記述する方法[Garl97, Shaw95].

2)部品インタフェースとそれを取り巻く関係を一体化して記述する方法[Ning96, OMG97].

SCLでは，部品カタログとして部品毎に独立して記述できるよう，部品単位で一体化して記述する方法を採用する．

(2)コンテキストと関係の表現：デザインパターンコンポーネント

部品の選択においてはコンテキスト，すなわち，部品が使われる状況を知ることが重要である．これを表現するために，部品を取り巻く局所的な構造を表現することにした．ここでは，デザインパターン[Gamm95]の概念を応用してUML(Unified Modeling Language)[Rati97]により部品を取り巻くクラス間の構造を表現する．これによって，部品の提供するサービスとそのインタフェース仕様，さらにサービスを提供するコンテキストを一まとまりに表現できる．

(3)表現方法

次のような表現方法をとる．

(a)実際への適用を考慮して，インタフェース記述はCORBA-IDL [OMG95]の上位互換とする．さらに，OMGで検討中のCDL仕様[OMG97]も考慮する．

(b)Web上に表現可能：HTMLの拡張(メタタグ)とする（XMLへの対応も考慮）．

(c)インタフェース記述部と設計情報などの付加情報と分離して記述する．

(d)付加情報は省略可能とする．

3．表現方法

3.1 表現の構造

SCLの情報構造を，図-1に示す．コマース情報と仕様情報から成る．それぞれ，段階的表現が可能である．簡潔さと豊富な情報の記述という相反する要求を満たすよう，それぞれの情報の特性と記述の目的に適した記述方法をとる．

(1)コマースプロトコル仕様記述部

販売カタログとしての情報を記述する．次の3つの部分から構成される．最小限の情報であり，評価情報以外は省略できない．

(a)部品の名称と提供するサービス(機能)，版数，仕様の概要など部品の概要を示す．サービス

はテキストで記述する．カテゴリは，部品が適用できるアプリケーションドメインもしくは機能分類を示す．カテゴリはキーワードを複数記述できる．

(b)ベンダや購入，ライセンス条件など，取引のための情報を示す．

(c)評価情報の提供

部品の第 3 者による評価や推奨などの評価情報があれば提供する．この情報はブローカの役割を実現するために必要である．ブローカが部品を評価し，優れた部品を利用者に推奨できる機構を実現する．

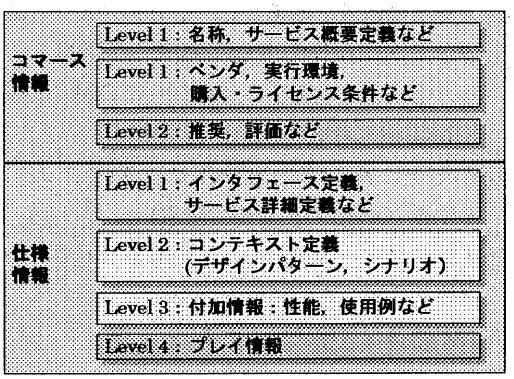


図-1 表現モデル

(2)仕様記述部

仕様記述部は次の 4 つの部分から成る．

(a)インタフェース定義

1 つの部品は 1 つあるいはそれ以上のインタフェースを持つことができる．インタフェース仕様は図-2 の形式の CORBA-IDL の上位互換となる拡張 IDL で記述する．従って，シグネチャの規則は CORBA-IDL と同一である．また，継承規則も CORBA-IDL 同じくインタフェースの継承のみである．さらに，インタフェース毎の提供機能を service としてテキストで記述できる．component は CORBA-IDL の module，CDL の component に対応する．

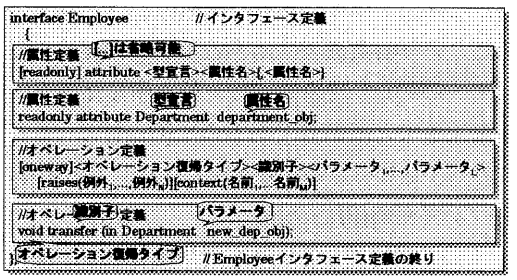


図-2 インタフェース定義

(b)コンテキスト定義：デザインパターン

コンポーネントとそれが直接関係するクラス群の静的構造と動的挙動を記述可能とする．構造表記は，UML [Rati97]のクラス構造の表記法を用いる．挙動は，代表シナリオ(ユースケース)を UML のメッセージトレース図を用いる．これは部品の理解を深めるための情報であり，必要に応じて記述する．

(c)付加情報部

部品の性能など次のような情報を記述する．

1)性能：プログラム量，メモリ量，実行時間などの性能情報があれば記述する．

2)利用に関する付加情報：使用例や注意事項，FAQ(Frequently Asked Questions)などの利用者に対する付加情報があれば記述する．

(d)プレイ情報

プレイを行うための，起動情報あるいは実行したシナリオを利用するための情報．

3.2 意味情報の表現方法

意味情報の定義を行うため，CORBA-IDL に対して，次の拡張を行った．

(1)インポートによる複合部品の定義

従来のインタフェース定義は，コンポーネントが提供するサービスのみを規定していた．これをここでは，エクスポート(Export)インタフェースと呼ぶ．一方，部品には単一の部品から

成る単一と複数の部品から成る複合部品がある．複合部品のインタフェースは，単一部品のインタフェースの継承，集約(組み合わせ)などで実現できる．部品を利用するためには，このような関連する部品とその関係が分かることが望ましいが，従来のインタフェース定義では表現の手段を提供していない[Ning96]．このため，Composingとして構造記述の枠組みを設け，この中でインポート(Import)インタフェースとして直接インポートしている部品のインタフェース名と部品名を記述できるように拡張する．

(2)挙動(事前・事後条件)の定義

インタフェースの起動条件などの挙動に関する条件を事前条件(precondition)と事後条件(postcondition)を表明(Assertion)として次のような論理式で定義する．

precondition(not Empty and Initialized)

ASLでは，事前条件とは別にメソッドを起動できる状態を定義できるが，状態は事前条件の一部と考えられるので，ここでは状態定義だけの表記は与えていない．状態定義の付加については，今後の検討課題である．

3.3 HTMLの拡張

すべての表記はHTMLの拡張として定義している．各項目は内容を記述する代わりにhttpのアドレスでリンク可能とする．これによって，記述の簡略化が図れ，また，柔軟性が増す．

4．記述例

Mediatorパターンを利用したダイアログボックスを管理するFontDialogDirectorコンポーネント[Gamm95]を例に，記述法を示す．なお，太字はキーワードを表わす．イタリックは省略可能項目を表わす．//は注釈を示す．

Component[コンポーネント]
// SCL Version 1.0
Name[名称]: FontDialogDirector
Service[サービス]:ダイアログ内のウイジェットの操作に対するイベントの制御を行う．
Category[分類]: GUI
Version[版]: 1.0
Date[作成日時]:
　Year/Month/Day Hour:Minutes
Change[変更履歴]:
　http://www.niit.ac.jp/nise/index.html
Location[提供場所]:
　http://www.niit.ac.jp/nise/index.html
Media[提供媒体]: Network
Vendor[ベンダ]
Name[ベンダ名]: NISE Lab.
Contact Address[問合せ先]:
　http://www.niit.ac.jp/nise/index.html
Purchase and License Condition[購入使用条件]
Price[価格]: 1,000円
License Condition[ライセンス]:
　Use: Floating License
　Re-distribution: Limited as .....
Support Condition[保守条件]:
Run-Time Environment [実行環境]
OS: Windows95, WindowsNT Version 4.0
Object Broker[ブローカ]: CORBA 2.0, DCOM
Conformance[確認済み実行環境]:
　Netscape Navigator, Version 3.0
　//列挙する
Container[コンテナ]:
GUI: Windows95, WindowsNT Version 4.0
Design-Time Environment[開発環境]
OS: WindowsNT Version 4.0
Environment: Visual C++, Version: 4.2
Language[言語]: C++

Application Framework: MFC, Version: 4.2

Endorsement[推奨]:

Recommended-by[推 薦 者]: SCALS Consortium

Evaluation[評価]:

Interface [インタフェース]

Interface:

FontDialogDirector : DialogDirector {

void WidgetChanged(in Widget*);

Precondition(.. and ..);

Postcondition(.. or ..);

service:ウイジェットの変更通知を受けて対応する操作を行うイベントを送信する.

void CreateWidgets();

};

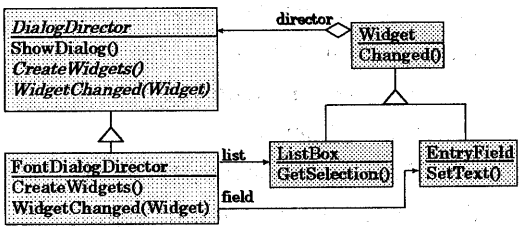
Composing [組み合せ条件]

Import[インポートコンポーネント]:

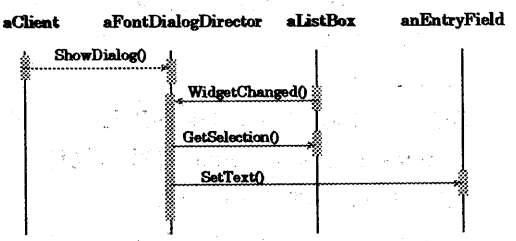
CreateWidgets from DialogDirector, Version: > 1.1

Design Specification[設計仕様]

Design Pattern[設計パターン]:



Scenario[シナリオ]:



Performance[性能]

Storage Size[プログラム量]:

Memory Size[メモリ量]:

Execution Time[実行時間]:

Usage[利用に関する付加情報]

Applications[適用例]:

FAQ:

Play[プレイ]

Test-Oracle:

本記述例は HTML のメタタグを用いて図-3 のように Web 上に表現できる.

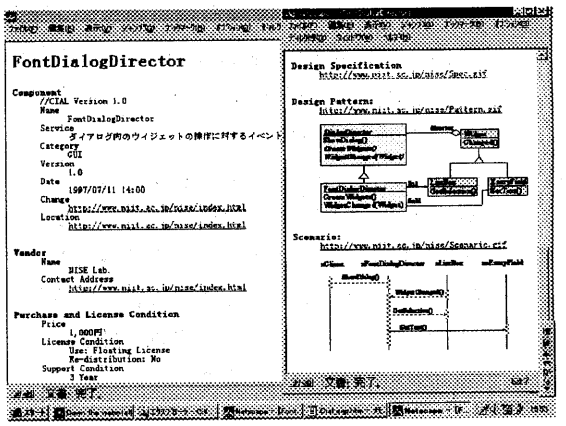


図-3 記述例の Web 表現

5. SCL 支援環境

5.1 CORBA-IDL との相互変換

SCL から CORBA-IDL, CDL, COM-IDL を生成する. 逆に, CORBA-IDL, CDL のソースからSCLのインタフェース記述部の一部を生成する. さらに, 現在検討中の CDL への対応も考慮する.

5.2 コンポーネントプレイヤ

部品再利用を妨げる主因の一つは，部品が再利用できるかどうか判断することが難しいことにある．この判断のための重要な情報は，部品のセマンティクス，特に，挙動の情報である．しかし，従来の部品リポジトリでは非形式的で曖昧なテキスト記述しか提供されなかった．また，CORBA などの IDL はセマンティクス情報を定義できないなどの問題があった．この問題に対する一アプローチとして，Web 上で部品を試行し，その挙動を視覚的に見ることができる支援環境コンポーネントプレイヤを提案する．詳細は，別稿として報告する予定である．

6．評価

6.1 関連研究

部品表記はソフトウェアアーキテクチャ記述言語 ADL(Architecture Description Language) と共に最近研究が活発に行われている．部品インタフェースの視点とアーキテクチャ記述の 2 つの視点から研究されている．部品インタフェースの記述を中心としたアプローチでは，CORBA-IDL を拡張した ASL[Ning96]や CDL [OMG97]がある．SCL では，部品カタログとしての利用と実際の開発への適用を考慮し，これらと同様のアプローチをとった．しかし，SCL ではカタログとしての利用を前提に，上記仕様記述言語では欠けているベンダなどの非機能的な情報やデザインパターンを応用したコンテキストの表現を新たに提案した．

一方，ADL としては様々な言語が提案されている．代表例としては，UniCon [Shaw95], Rapide [Luck96]がある．これらは，モデル化における着眼点によって異なる．例えば，UniCon がコンポーネントとそれを接続するコネクタの接続を表わす静的なモデルであるのに対し，Rapide はイベントの交換による挙動の抽象化モデルである．現段階では，ADL を閉じた形式の言語として定義することは困難であると考えられている[Garl97].

6.2 他のインタフェース記述言語との比較

表-1 に代表的なアーキテクチャ記述言語である ASL, ACME, Rapide と SCL の比較を示す．SCL はコンポーネントブローカに適用するためのモデルとして開発したので，同様のアプローチをとる ASL に近い．

表-1 ADL の比較

ADL	ASL	ACME	Rapide	SCL
インタフェース	CORBA-IDL の拡張	Components としてシグネチャを記述	プロセス (CSP ベース)	CORBA-IDL の拡張
結合モデル	静的構造 (RPC)	静的構造 port)	動的挙動 (CSP)	静的構造 (RPC)
結合の表現	インタフェースの一部として provides, requires の関係を記述	コネクタ定義として独立に記述．コネクタはコンポーネントとポートで結ばれる．	インタフェースの一部として provides, requires の関係を記述	インタフェースの一部として provides, requires の関係を記述
セマンティクス	インタフェースにプリ／ポストコンディションを定義	ポートに対する Role として記述	イベントに対するアクション	インタフェースにプリ／ポストコンディションを定義
構成情報	Configuration の中で付加情報として定義	なし	なし	Configuration の中で付加情報として定義
コンテキスト	なし	なし	なし	デザインパターンで記述

7．今後の課題

(1)インタフェース仕様記述

本稿で提案したインタフェース定義は CORBA-IDL の基づき，OMG で検討中の CDL

を考慮した．さらに，MS-IDL による ActiveX (DCOM)[Brow96] と Java Beans[Cabl97a, Cable97b, IBM97], Java RMI を包含できるインタフェース記述が可能となるよう検討中である．また，これらの分散オブジェクト環境は固有の意味モデルや Java Beans の Introspection などコンポーネントに関する情報の固有な表現方法と支援環境を提供しているが，インタオペラビリティがない．SCL はこれらの分散オブジェクト環境でインタオペラブルな情報モデルを提供する．

(2)支援環境のプロトタイプ開発と評価

本稿で提案したプレイ可能なカタログ表現言語の支援環境のプロトタイプを開発している．また，インターネット上の様々な Web サイトで提供されている部品情報を収集し，SCL で統一的に表現できる環境のプロトタイプも開発している．さらに，この環境は SCL と IDL, CDL との変換も支援する計画である．

8．まとめ

ソフトウェアの取引情報とインタフェース仕様を統合して Web 上に表現する記述言語 SCL を提案した．現在，SCL の詳細を検討中である．

さらに，ソフトウェア CALS 実証実験の一環として，SCL を用いてインターネット上でソフトウェア部品の電子取引を行うコンポーネントブローカのプロトタイプを開発している．

最後に，本研究を進めるにあたりご討議頂いたソフトウェア CALS コンソーシアム次世代実証実験グループならびに INSTAC コンポーネントウェア WG の関係各位に感謝する．本研究は情報処理振興事業協会による企業間高度電子商取引推進事業の一環として行っている．また，情報サービス産業協会の HITOCC などの支援による．ご支援頂いた関係各位に感謝する．

参考文献

[Aoya96a] 青山幹雄，コンポーネントウェア:部品組立て型ソフトウェア開発技術，情報処理，Vol. 37, No. 1, Jan. 1996, pp. 91-97.

[Aoya96b] 青山幹雄，コンポーネントウェア：ソフトウェア CALS のめざす次世代開発像，JISA 会報，No. 44, Dec. 1996, pp. 58-69.

[Brow96] N. Brown, et al., *Distributed Component Object Model Protocol —DCOM/ 1.0*, http://www.microsoft.com.

[Cabl97a] L. Cable and G. Hamilton, *A Draft Proposal to Define an Extensible Runtime Containment and Service Protocol for Java Beans*, Version 0.95, http://splash.javasoft.com /beans/, 1997.

[Cabl97b] L. Cable and G. Hamilton, *A Draft Proposal for a Object Aggregation/ Delegation Model for Java and Java Beans*, Version 0.5, http://splash.javasoft.com/ beans/, 1997.

[Gamm95] E. Gamma, et al., *Design Patterns*, Addison-Wesley, 1995.

[Garl97] D. Garlan, et al., ACME: An Architecture Description Interchange Language, http://www.cs.cmu.edu, Jan. 1997.

[IBM97] IBM, et. al., *CORBA Component Imperatives*, ORBOS/97-05-25, May 1997.

[Luck96] D. Luckham, et al., *Guide to the Rapide 1.0 Language Reference Manuals*, The Stanford Rapide Project, Dec. 1996.

[Mowb97] T. J. Mowbray and R. C. Malveau, *CORBA Design Patterns*, John Wiley & Sons, 1997.

[OMG95] OMG, *The Common Object Request Broker: Architecture and Specification*, Revision 2.0, http://www.omg.org, Jul. 1995.

[OMG97] OMG Business Object Domain Task Force, *Business Object Facility*, Revision 1.0, Jan. 1997.

[Ning96] J. Q. Ning, A Component-Based Software Development Model, *Proc. IEEE COMPSAC '96*, Aug. 1996, pp. 389-394.

[Naga96]長野宏宣，ソフトウェア CALS の狙いと実証実験について，情報処理，Vol. 37, No. 12, Dec. 1996, pp. 1083-1088.

[Rati97] Rational Software Co., *UML Notation Guide*, Version 1.0, http://www.rational.com, Jan. 1997.

[Shaw95] M. Shaw, et al., Abstraction for Software Architecture and Tools to Support Them, *IEEE Trans. Software Eng.*, Vol. 21, No. 4, Apr. 1995, pp. 314-335.