社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－SE－147（5）
2005／3／17

宣言的なプログラム解析が可能な RDF に基づく細粒度ソフトウェアリポジトリ

吉田　一[†]　山本　晋一郎[††]　阿草　清滋[†]

本報告では，ネットワークを介した幅広い環境から利用でき，プログラムの宣言的な解析を可能とする，新しい細粒度ソフトウェア・リポジトリを提案する．細粒度ソフトウェア・リポジトリは，プログラム中の文や式のような細粒度情報を格納している．本研究では，このソフトウェア・リポジトリに RDF を適用する．RDF はセマンティック Web におけるメタデータ記述の枠組みであり，さまざまな物に関するデータを有向グラフを用いて記述する標準となっている．CASE ツールが幅広い環境からプログラムの情報を照会できるように，RDF に基づくリポジトリのモデルを設計し，このリポジトリをネットワーク上に発行した．リポジトリのモデルはソフトウェア要素間の基礎的な関連を表現しており，CASE ツール開発者は新たな関係を推論規則によって宣言的に定義することができる．このような宣言的なプログラム解析は，CASE ツール開発を本質的な機能の実装に集中させることが可能になる．

A fine-grained software repository based on RDF for declarative program analysis.

Hajime Yoshida,[†] Shinichirou Yamamoto[††] and Kiyoshi Agusa[†]

In this report, we propose a new fine-grained software repository which is accessible from wide range of environments through the network and capable of declarative program analysis. A fine-grained software repository has fine-grained information such as statements and expressions in programs. In this research, we apply RDF to the software repository. RDF is a framework of metadata description on the Semantic Web, and becomes a standard of data description using directed graphs about various types of object. We designed the RDF-based repository model and deployed the repository onto the network so that CASE-tools can query program informations from wide range of environments. And the model describes primitive relationships between software elements so that CASE-tool developer can define new relations declaratively by inference rules. Such declarative program analysis makes CASE-tool development essential.

1. はじめに

理解支援ツールなどの下流 CASE ツールでは，対象となるプログラム・ファイルの内部に含まれている要素を把握する必要がある．したがって，下流 CASE ツールを開発するには，対象となるソース・プログラムの構文解析および意味解析を行う機能が必須となる．しかし，一般にソフトウェア文書の解析は複雑な処理であり，この解析機能を実装するには多くの労力を必要とする．このため個々の CASE ツールが個別に解析機能を実装することはソフトウェア工学発展の障害になっていた．

Sapid[1),2)] (Sophisticated APIs for CASE tool Development) はこのような CASE ツール開発・保守における作業の効率化を目的として提案された，汎用の CASE ツール・プラットフォームである．Sapid は以下のコンポーネントによって構成されている．

- ソフトウェア文書の解析を行う解析器
- 解析で得られた情報を管理するリポジトリ
- リポジトリの情報を読み書きするための API

解析器はソースプログラムの手続き内部の細粒度要素までを解析し，データベースであるソフトウェア・リポジトリに格納する．細粒度要素にはステートメント，式，コメントなどが含まれ，下流 CASE ツール開発に

† 名古屋大学 大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†† 愛知県立大学 情報科学部
Faculty of Information Science and Technology, Aichi Prefectural University

必要な情報を含んでいる．細粒度要素を含むこのソフトウェア・リポジトリを**細粒度ソフトウェア・リポジトリ**と呼ぶ．CASE ツール開発者は，提供される API を利用することで，必要なソフトウェア要素の情報を取得することが可能となっている．

Sapid の導入によって CASE ツール開発・保守における作業を効率化することが可能である．しかし，API が特定のプログラミング言語向けに設計されているなどの理由から，プラットフォームの機能を利用できる環境に制限があった．また，汎用の照会言語などを用いて情報照会を行うことができないため，汎用技術による実装が要求されてきた．

本研究の目的は，細粒度ソフトウェア・リポジトリの機能を向上させ，より多くの CASE ツール開発に役立つツール・プラットフォームを実現することである．リポジトリの機能要件として，次の 2 点を満たすこととする．

- ネットワークを介した広範な利用環境から情報を読み書きできること
- CASE ツール開発に必要な情報を照会言語を用いて宣言的に取得できること

そこで本研究では，RDF[16](Resource Description Framework) に基づいた細粒度ソフトウェア・リポジトリを提案する．RDF はセマンティック Web におけるメタデータ記述の枠組みである．メタデータは有向グラフで記述され，ソフトウェア解析情報の記述に適している．このグラフを格納管理する関連技術もまた整備されており，記述したグラフを Web を介して幅広く共有することが可能になる．

2. 従 来 研 究

Sapid の従来研究では，リポジトリの利用環境改善をねらい，XML を用いたリポジトリ形式 XSDML[3] を提案した．XSDML は細粒度ソフトウェア・リポジトリを XML を用いて実装する方式であり，この提案によってリポジトリの利用環境を従来より改善することができた．しかし，逆に XML を利用することに伴う制約も生じており，この課題の解決が必要とされてきた．

主要な課題の一つはファイルをまたがった要素間の関連表現である．XSDML では，要素間の構文的な親子関係を示す関連を，DOM における親要素，子要素として表現し，ファイル内部の要素間に存在する定義・参照の関連を，IDREF 型属性からの参照として表現している．しかし，XML の仕様によって，IDREF 型属性が指し示す要素は同一ファイル内の要素に限られる．よって DOM を用いたファイルをまたがる要素間のトラバースは実現できない．

2.1 ファイル間関連に対する考察

ファイル間関連を適切に扱うため，まず，簡潔に参照先ファイルのパスを属性として持たせる方法を検討する．この方法では簡潔な表現が可能なものの，DOM API はパスを単純に文字列としてしか認識しない．ファイル境界を意識せず，統一的な方法で参照をトラバースするには，DOM の上位に新たな API を設けて実装の違いを隠す必要がある．しかし新たな API を設けることは DOM の他に独自モジュールを必要とするため，利用環境の観点からは好ましくない．

次に，XLink[18](XML Linking Lanuage) を用いた方法を検討する．XLink は XML 文書間のハイパーリンクを表現する手段を提供する．しかし xlink:onLoad 属性などの文書表示の用途に特化した属性が含まれ，論理構造を表現する適切な方法とはいえない．また，XLink では XPointer[19](XML Pointer Language) によってアンカーを指定する必要があるが，XSDML には細粒度の要素について既に ID が付加されているため，このような機能は冗長である．

最後に，複数のソース・プログラム情報を一つの巨大な DOM ツリーで表現し，Xindice[7] などのネイティブ・XML データベースにツリーを格納する方法を検討する．この方法では，XPath(XML Path Language) による照会を利用して関連をたどることができるようになるが，逆に全てのリンクは参照先の要素を持つことが要求される．このため，ファイル単位での情報の追加や削除を行うことが困難になる．

このような XML の特徴は，XML が文書交換の技術であり，ファイル間関連を含む巨大なデータベースとしての利用に向かないことを示している．そこで，ファイル間関連を適切に扱うには他の標準技術の導入を検討する必要があり，本研究では RDF に着目することとした．

3. ソース・プログラムの 2 つのビュー

プログラムから得られる細粒度の構成要素には，シンタックス・ビューとセマンティクス・ビューの 2 つのビューに大別でき，これらは互いに補完しあっている．

シンタックス・ビューは，ソース・プログラムの文書構造を表現するビューである．シンタックス・ビューの特徴は，一つの文書を対象に，その文書に含まれる構文要素の親子関係を木構造で表し，また要素の出現順序を保存する点である．ソースプログラムの表示，編集および付箋機能などを提供する CASE ツールでは，

プログラムの文書構造を扱う必要があり，このビューの情報を利用している．リポジトリがこのような情報を提供することによって，ツール開発において煩雑になりがちなソース・プログラムの字句・構文解析を必要とせず，見通しよい開発ができる．

セマンティクス・ビューは，ソフトウェア文書の構文要素を解釈して得られるソフトウェア構成要素，および言語に組み込まれているソフトウェア構成要素間の関係を表現するビューである．このため，セマンティクス・ビューは静的解析に重要な役割を持っており，利用者の要求に応じてさまざまな情報を提供する必要がある．セマンティクス・ビューの特徴は，含まれるデータがクロスリファレンス表の役割を果たしており，構成要素間の多様な関係を有向グラフによって表現する点である．プログラム解析を必要とする CASE ツールは，主にセマンティクス・ビューの情報を必要としており，リポジトリがこの情報を提供できることがプラットフォームの普及に必須の条件である．

XSDML はシンタックス・ビューの表現にとっては有効な手段である．しかし，ファイル間にまたがる要素間の関連を適切に扱えないため，セマンティクス・ビューの表現を行うには適していない．そこで，セマンティクス・ビューのデータについて考察すると，ここで表現されるデータは，本質的にシンタックス・ビューの要素に対するメタデータと見なすことができる．例えば，シンタックス・ビューの要素であるプログラム中のクラス定義は，その構文要素が通常のデータであり，対応するテキスト列を解釈して得た派生元クラスやクラス・メンバの情報を，データに対するデータすなわちメタデータであるとみなす．

このような考察から，RDF をセマンティクス・ビューの表現に導入し，XSDML と相互に補完させることは自然である．RDF には XML 同様に豊富な関連技術が整備されており，XML を利用することで得られた汎用性を損ねることはない．なお，以降の節では解析対象となる文書を Java(TM) ソース・プログラムとして議論を行う．

4. セマンティクス・ビューと RDF

RDF のデータモデルは，URI[20](Uniform Resource Identifier) によって識別される**リソース**と，それらの関係を表現するための形式的なモデルである．RDF では，図 1 のように，データをラベルつき有向グラフによって表現する．RDF ステートメントとは，subject - 特定のリソース，predicate - 指定されたプロパティ，object - リソースに対するプロパティの値からなる 3 項組である．プロパティの値である object は，別のリソースへの参照であるか，単純に文字列を保持するリテラルである．このモデルは関係データベースの表を表現できる能力も併せ持つ[17]．

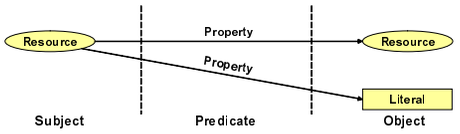

図 1 RDF によるメタデータ記述

複数の RDF ステートメントは有向グラフを構成する．リソースはグラフのノードに対応し，同一の URI を持つリソースは同一のノードとして解釈される．この構成された有効グラフは RDF グラフと呼ばれ，RDF グラフを格納管理する関連技術として各種の RDF リポジトリが整備されている．

RDF グラフを用いて細粒度ソフトウェア・リポジトリを構築するには，細粒度のソフトウェア構成要素の実体をリソースとしてとらえ，これら実体間の関係をプロパティとして RDF ステートメントを記述していけばよい．このうち，プログラム中で実体が定義される要素については，図 2 に示すように，XSDML 要素に対して RDF リソースを直接対応させる．XSDML では Java プログラムを JX-model[3] によってモデル化した．JX-model はシンタックス・ビューのモデルである．これに対応して，Java プログラムのセマンティクス・ビューを RDF のグラフで表現するモデルを，以後 **JR-model** と呼ぶことにする．

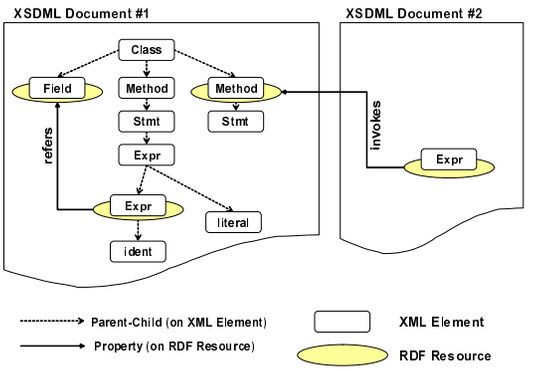

図 2 リソース・プロパティの割り当て

4.1 ソフトウェア構成要素の URI

URI はリポジトリ内のソフトウェア構成要素を識別

する文字列であり，RDF グラフで表現する個々のリソースは URI にって識別される必要がある．まず，ステートメント，式といったプログラム中の特定の構文要素の URI を規定する．URI の役割として，その表記はプログラムの構文要素を容易に特定できる書式にすべきである．したがって，URI は必然的にファイルの場所を示す表記に XML 要素の ID を付加した書式となる．ファイルの場所を示す表記には Java のパッケージシステムによる階層化を適用する．例えば，以下の URI はパッケージ `java.io` に含まれるファイル `Writer.java` を示す．

```
"xsdml:/java/io/Writer"
```

そして，以下の URI によって対応する XSDML 文書中の ID 3755 を識別する．

```
"xsdml:/java/io/Writer#3755"
```

この URI によって，構文要素の大域的な識別が可能となった．しかし，プリミティブ型を表す実体などはプログラム中のテキストで定義されない．このため型を識別するためには，構文要素を識別する URI のみでは表現できない．そこで，プリミティブ型や配列型を含む一般的な型を識別するために，新たに型の完全修飾名 (Fully Qualified Name) に基づく URI を規定する．型の完全修飾名は，Java(TM) コンパイラが大域的に型を識別するために用いており，URI として適切である．以下に型を識別する URI の例を示す．

```
"fqn:/int"                  (Primitive "int")
"fqn:/int[]"                (Array of "int")
"fqn:/java/applet/Applet"
                  (Class "java.applet.Applet")
```

5. RDF リポジトリを用いたデータ共有

RDF リポジトリは，一般に RDF グラフの追加・削除を行う永続記憶機能，推論規則に従ってグラフを展開する推論機能，特定のリソースを宣言的に取得する照会機能などを持つ RDF 関連技術である．RDF リポジトリの永続記憶機能は，RDF グラフのデータを MySQL(TM)[6] などの関係データベースへ格納することを可能にする．この永続記憶機能によって，広大なグラフを扱うアプリケーションを容易に開発できるようになる．

本研究では，RDF グラフの構築および発行のために，openRDF[21] によって開発されている RDF リポジトリ Sesame rev.1.1[22] を用いる．Sesame は Tomcat 上の Web アプリケーション・サーバとして動作し，HTTP 経由でさまざまな機能を提供する．Sesame を用いた場合のツール・プラットフォーム構成を図 3 に示す．

mkJRmodel は JR-model RDF グラフを構築するモジュールである．ここで，mkJRmodel の入力には，あらかじめファイル間参照が解決された XSDML 文書群を与える．この参照解決済み XSDML の生成には，立命館大学の丸山氏によって開発された Sapid JTool を用いている．JTool はまず Sapid を用いて JX-model XSDML 文書群を生成し，続けてファイル間参照が解決されていない XSDML JX-model 文書群の参照解決を行う．そして，参照先の情報を XML 属性として付加した JX-model level3 XSDML 文書群を出力として得ることができる．mkJRmodel はこの参照先の情報を利用して，XSDML のツリーを RDF グラフに変換する．

mkJRmodel が構築した RDF グラフは順次 Sesame サーバに直接送信して，あらかじめ指定した MySQL データベースにグラフデータを格納させる．このデータベースが細粒度ソフトウェア・リポジトリとなる．セマンティクス・ビューのデータが収められたこの細粒度ソフトウェア・リポジトリを以後 RSSDB (RDF-based Semantic Software DataBase) と呼ぶことにする．

CASE ツールは Sesame の API を用いることで，リポジトリの情報を照会することが可能である．Sesame の API を用いる必要があるが，RDF グラフを一度取得してしまえば，他の RDF プロセッサを利用することも可能である．なお，RDF リポジトリの推論機構が用いる Inference Rules にはデフォルトで RDF の意味論に基づく推論規則が含まれている．7 節ではこの Inference Rules を拡張し，演繹推論によるプログラム解析を可能にする．

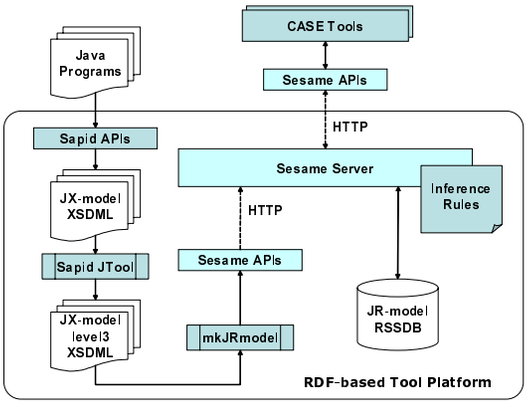

図 3 プラットフォーム構成図

5.1 RDF の枠組みを適用する利点

RSSDB の実装に RDF および RDF リポジトリを用いるメリットを以下にまとめる．次章では，リポジトリに格納するセマンティクス・ビューのモデル化を行う．

大域的な参照の表現

RDF のデータモデルは，URI が同一であれば同一の実体と解釈する意味論によって，極めて柔軟な有効グラフのデータモデルとなっている．このため，複数ファイルにまたがる要素間の関連を適切に表現し，ノード間のトラバース機能を合理的な方法で実現することが可能である．

関係データベースへの永続化

RDF はセマンティック Web 技術の一部として設計されており，本質的に巨大なデータベースを扱うことが可能である．有向グラフを関係データベースへ永続化することで，二次記憶の容量が許す限りの広大なグラフをトラバースできる．セマンティクス・ビューはプロジェクト全体のソース・プログラムから巨大なグラフを構成できる能力を必要とするため，関係データベースへの永続化は必須の機能である．

照会言語と推論機構

グラフ構造に対する RDF 特有の照会言語が利用でき，CASE ツール・開発者はプログラム中の特定の要素を宣言的に取得することができる．この照会機能に加えて，RDF の枠組みにはプログラム要素間の新たな関係を宣言的に定義できる推論機構が含まれる．この推論機構によって．CASE ツール開発者はセマンティクス・ビューから高度なビューを得るために，グラフを手続き的に操作することなく，専用の推論規則を用意すればよい．

6. Java セマンティクス・ビューのモデル

JR-model ver.1.0 は 9 種のクラスと 28 種のプロパティから成る．図 4 に JR-model のクラス図を示す．なお，モデルの詳細は本報告では割愛する．クラス間の矢印は JR-model プロパティであり，このうち名前にアンダーバーを含むプロパティは XSDML 要素の親子関係，すなわち構文上の包含関係に相当する．クラスとはリソースの型であり，リソースをグループ化する役割を持つ．プロパティには図に示した他に，任意のクラスに対して，識別子名を表す name，および要素の細かい種類を表す sort が適用できる．

ソースプログラムから構築される具体的な RDF グラフ例を，図 5 に示す．同一の URI はグラフ上の 1 つのノードとして解釈され，それぞれのソース・プログラムに対応する有向グラフがマージされる．このようにソフトウェア構成要素を RDF グラフとして表現するによって，リポジトリはファイル間にまたがる要素間の関連や，組み込みソフトウェア構成要素との関連を適切に扱うことができる．

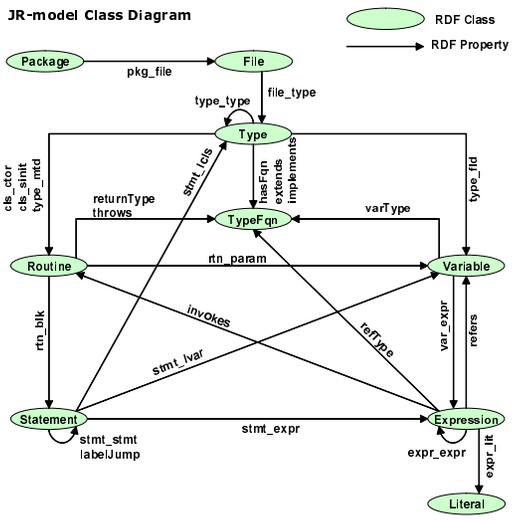

図 4 JR-model クラス図

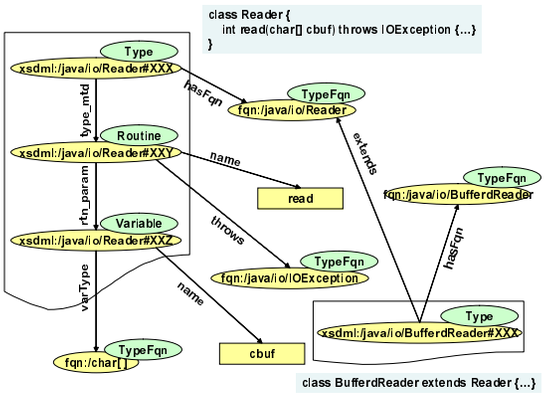

図 5 セマンティクス・ビューの JR-model による表現

JR-model グラフを構築する mkJRmodel のコア・プログラムは，約 500 行の Java プログラムによって記述されている．この他に，XSDML の操作ライブラリとして約 450 行，語彙定数を提供する約 100 行の Java プログラムが含まれる．なお，現在のところ，Package，File クラスには対応していない．

6.1 CASE ツールからの情報照会

RDF を用いることの大きなメリットは，データベースとしての機能を提供する関連技術が豊富に利用できる点である．Sesame では RDF グラフに対する照会言語

として，SeRQL[23] を提案している．なお，SeRQL には照会結果をテーブルとして返す SeRQL-SELECT と，結果を新たな RDF グラフとして返す SeRQL-CONSTRUCT 2 種類の動作がある．

図 6 に示す SeRQL-SELECT クエリは，変数参照のうちフィールド変数を参照している箇所について，その参照元の式 (E)，参照先の変数 (V)，参照先変数の名前 (N) の組を返す．ただしこの例では，変数に jr:name, jr:sort プロパティが設定されていない組は照会結果に含まれない．実行効率については 7.1 節で示している．

```
SELECT *
FROM {E} jr:refers {V} jr:name {N};
                       jr:sort {"Field"}
USING NAMESPACE jr =
  <http://www.sapid.org/2005/01/31-jr-model#>
```

図 6 フィールド変数参照を抽出するクエリ文

この例から，RSSDB によって細粒度のソフトウェア情報を幅広く共有できることがわかる．RSSDB を共有するために必要な Sesame サーバの設置にはある程度のコストがかかるが，クライアントである CASE 開発者は Sesame API 以外の特別なソフトウェアを必要としない．さらに宣言的な情報照会が可能であるため，CASE ツールに必要なソフトウェア解析の手続きは，DOM を用いた XML 文書のトラバースにより解析する事例に比べて簡潔である．

7. 演繹推論による新たな関係の導出

RDF の枠組みは，ソフトウェアによるメタデータの自動処理をサポートする推論規則を持つ．Sesame の推論機構は，RDF が標準で用いる推論規則をカスタマイズすることが可能である．そこで，JR-model のグラフが示す既存のソフトウェア要素間の関係から，便宜上有益な新たな関係を導出するために，JR-model に必要な推論規則を加える．

例えば，メソッド呼び出しグラフの構築には，Routine クラスのリソース間での呼び出し関係を把握することが必要である．しかし，JR-model のスキーマでは，メソッド呼び出しを，Expression クラスの inkoves プロパティで表現している．このため，手続き的にメソッド呼び出し関係を求めるには，expr_expr のような包含関係を何度もたどる必要がある．そこで，メソッド間の呼び出し関係を Routine クラスのプロパティとして，表現できるようにリポジトリを拡張する．

このプロパティを推論によって導くには，表 1 に示す推論規則を追加する．規則 jr1, jr2 は，expr_expr の反射的推移関係 expr_expr* を定義する．まず，あるリソースが Expression 型であれば，自身に対して expr_expr* 関係が成立するとし，これを解釈 E に加える．さらにプロパティ expr_expr* と expr_expr を連接して到達できる 2 つのリソース間にも expr_expr* 関係が成立するとし，これを解釈 E に加える．同様に 規則 jr3, jr4 は，stmt_stmt の反射的推移関係 stmt_stmt* を定義している．

表 1 メソッド呼び出し関係を定義する推論規則

	If E contains:	then add:
jr1	R1 [rdf:type] [jr:Expression]	R1 [jr:expr_expr*] R1
jr2	R1 [jr:expr_expr*] R2 R2 [jr:expr_expr] R3	R1 [jr:expr_expr*] R3
jr3	R1 [rdf:type] [jr:Statement]	R1 [jr:stmt_stmt*] R1
jr4	R1 [jr:stmt_stmt*] R2 R2 [jr:stmt_stmt] R3	R1 [jr:stmt_stmt*] R3
jr5	S1 [jr:stmt_expr] E1 E1 [jr:expr_expr*] E2 E2 [jr:invokes] M1	S1 [jr:s-invokes] M1
jr6	M1 [jr:rtn_blk] S1 S1 [jr:stmt_stmt*] S2 S2 [jr:s-invokes] M2	M1 [jr:m-invokes] M2

このように，expr_expr* 関係と stmt_stmt* 関係を定義できれば，図 7 に示すように，規則 jr5 を適用して，ステートメント粒度での呼び出し関係 s-invokes を定義できる．同様に，この s-invokes プロパティを利用すれば，規則 jr6 を適用して，目的としているメソッド粒度での呼び出し関係 m-invokes を定義できる．

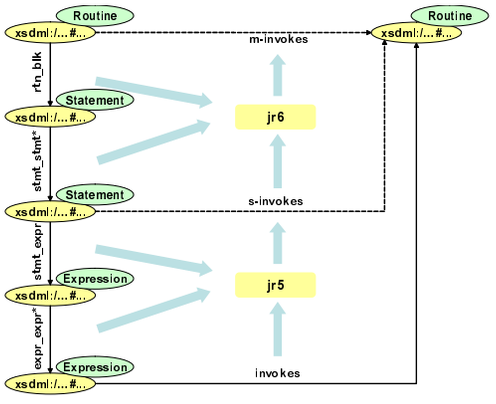

図 7 s-invokes, m-invokes 関係の追加

新しく定義した m-invokes プロパティは Rouine ク

ラスのリソース間の呼び出し関係を表している．したがって，このプロパティのみを含む部分グラフを抽出すれば，メソッド呼び出しグラフを取得することができる．部分グラフの抽出には図 8 に示す SeRQL-CONSTRUCT クエリを用いる．

```
CONSTRUCT *
FROM {R1} jr:m-invokes {R2}
USING NAMESPACE jr =
  <http://www.sapid.org/2005/01/31-jr-model#>
```

図 8 メソッド呼び出しグラフを抽出するクエリ文

このように，ソフトウェア構成要素間の新しい関係を見つけるために，推論規則による宣言的な定義を用いることは，従来必要であった手続き的な処理や照会を用いたグラフの更新を不要にし，簡潔なプログラム解析を可能にする．

7.1 実 行 効 率

空間効率および時間効率の測定を以下の環境で行った．なお，Sesame の推論機構は表 1 の規則によって拡張された状態である．

- Celeron® 2.0GHz CPU, 512MB RAM
- Windows® XP SP2 Operating System
- Java(TM)2 Software Development Kit 1.4.2_06
- Tomcat 5.0.27 Java Servret Container
- MySQL(TM) 4.0.21 DBMS / JDBC 3.0.16
- Sesame 1.1 RDF Repository

解析対象のサンプルとして JDK の demo フォルダに含まれている applets/SortDemo (以後 PRJ1) および jfc/Notepad (以後 PRJ2) を用いた．それぞれ，Java ソースファイル群のサイズ，JTool の生成する XSDML ファイル群のサイズ，RSSDB を格納している MySQL のデータディレクトリのサイズ，クエリによって返されるレコード数と応答時間を求めた．Query1 は図 6 の照会であり，Query2 は図 8 の照会を SELECT に置き換えた照会である．MySQL および Sesame はいずれも同一のコンピュータで動作させ，クエリ応答時間の測定には Sesame の Web Browser インタフェースを用いて行った．結果を表 2 に示す．

結果より，RSSDB のサイズはソースプログラムの 50〜70 倍となっている．この倍率は，個々の CASE ツールが RSSDB の情報を保持するには若干高いが，共有リポジトリのサイズとしては実用的な値である．照会の応答時間は，条件となるプロパティの数に応じて増えているが，実用的な時間で情報を所得可能であるといえる．推論によって定義した m-invokes を取得する場合でも，応答の遅延は見られない．RDF の枠組みでは，必要に応じたグラフのマージが可能であるため，専用のリポジトリ・サーバを設け，RSSDB の情報を管理させることができれば，さらに巨大なプロジェクトにも対応できると考えている．

表 2 空間・時間効率の測定結果

	PRJ1	PRJ2
Java Source Files [kB]	19.8	63.7
XSDML level3 Files [kB]	136	718
RSSDB Data Directory [kB]	1140	4160
Query1 Result [Records]	52	244
Query1 Response Time [ms]	130	530
Query2 Result [Records]	35	310
Query2 Response Time [ms]	20	80

8. 関 連 研 究

データ交換，可搬性などの利点から，XML ベースのソフトウェア情報データベースを提案している研究が行われている[8)~15)]．しかし，RDF の枠組みをソフトウェア・リポジトリに適用する研究は行われていない．XML ベースの各研究とも，大域的な関連情報を CASE ツールに提供することが難しい．本研究では RDF を新たな基盤としたソフトウェア・リポジトリを導入することで，これらの問題の合理的な解決を図っている．

9. おわりに

細粒度ソフトウェア・リポジトリには，シンタックス・ビューとセマンティクス・ビューが共存している．このうちセマンティクス・ビューはソース・プログラム全体の大域的な関連を扱う必要があり，XML のデータ交換フォーマットとしての特性から，セマンティクス・ビュー表現に XML を単純に利用することは適切ではない．

本研究では，細粒度のソフトウェア情報をネットワークを介して広く共有するため,，RDF に基づく細粒度ソフトウェア・リポジトリ RSSDB を提案し，実際に，MySQL データベースに RSSDB を格納する処理系を実装した．この結果，実用的な時間で照会言語を用いた宣言的なソフトウェア解析が可能になった．

RSSDB に格納されている RDF グラフは，要素間のプリミティブな関係の表現としており，演繹推論によって新しい要素間の関係を導出する方式をとっている．CASE ツール開発者は，新たな関係を導出するために推論規則を記述すればよく，手続き的なコード実装を必要としない．このため，開発者は本質的な機能

の実装に集中することが可能である．

Sapid の関連 CASE ツールでは，ソフトウェア・リポジトリの高度なビューとして，オブジェクト指向システムの依存グラフである OSDG[4](Object-oriented System Dependence Graph) や，関数呼び出しに着目した依存グラフである，FCDG[5](Function Call Dependence Graph) などが提案されている．今後の課題として，これらのビューに存在するソフトウェア要素間の関係を推論規則によって定義し，CASE ツール実装の見通しを向上させることを目指している．

参 考 文 献

1) 福安 直樹，山本 晋一郎，阿草 清滋: 細粒度ソフトウェア・リポジトリに基づいた CASE ツール・プラットフォーム Sapid, 情報処理学会論文誌, Vol.39, No.6, pp.1990–1998 (1998).
2) Y. Hachisu, S. Yamamoto, K. Agusa: A CASE Tool Platform for an Object Oriented Language, IEICE Trans. on Information and Systems, Vol.E82-D, No.5, pp.997-984 (1999).
3) 吉田 一，山本 晋一郎，阿草 清滋: XML を用いた汎用的な細粒度ソフトウェアリポジトリの実装, 情報処理学会論文誌, Vol.44, No.6 (June 2003).
4) 蜂巣 吉成，山本 晋一郎, 阿草 清滋: オブジェクト指向言語のための細粒度システム依存グラフ, 情報処理学会論文誌, Vol.40, No.4, pp.1851–1860 (1999).
5) 三浦 良，山本 晋一郎, 阿草 清滋: プログラミングナビゲーションのための関数呼び出し依存グラフ, 日本ソフトウェア科学会コンピュータソフトウェア別冊, ソフトウェア発展, pp.19–29 (2000).
6) MySQL: The World's Most Popular Open Source Database. http://www.mysql.com/
7) Apache Xindice. http://xml.apache.org/xindice/
8) M. L. Collard, J. I. Maletic, A. Marcus: Supporting Document and Data Views of Source Code, the 2nd ACM Symposium on Document Engineering (DocEng '02), McLean, VA, pp.34–41 (Nov. 2002).
9) M. Boshernitsan, S. L. Graham: Designing an XMLBased Exchange Format for Harmonia, the 7th Working Conference on Reverse Engineering (WCRE'00), Brisbane, Australia, pp.287–289 (Nov. 2000).
10) G. J. Badros: JavaML: A Markup Language for Java Source Code, the 9th International World Wide Web Conference (2000).
11) Y. Zou, K. Kontogiannis: Towards a portable XML-based source code representation, XML Technologies and Software Engineering (XSE) (2001).
12) GCC-XML Home Page. http://public.kitware.com/GCC_XML/HTML/Index.html
13) 山中 祐介，大畑 文明，井上 克郎: プログラム解析情報の XML データベース化, 日本ソフトウェア科学会コンピュータソフトウェア, Vol.19, No.1, pp.39–43 (2002).
14) 田中 哲，一杉 裕志: ソースコード理解支援ツール — JavaMarkup, 日本ソフトウェア科学会 SPA2002 (2002).
15) K. Gondow, H. Kawashima: Towards ANSI C Program Slicing Using XML, Electronic Notes in Theoretical Computer Science, Vol.65, No.3, Elsevier Science Publishers (2002).
16) Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation (Feb. 2004).
17) T. Berners-Lee: Relational Databases on the Semantic Web, W3C Design Issues (Nov. 2001).
18) XML Linking Language (XLink) Version 1.0, W3C Recommendation (June 2001).
19) XML Pointer Language (XPointer) Version 1.0, W3C Last Call Working Draft (June 2001).
20) T. Berners-Lee, R. Fielding, L. Masinter: Uniform Resource Identifiers (URI): Generic Syntax (RFC 2396) (1998).
21) openRDF.org ...home of Sesame . http://www.openrdf.org/
22) J. Broekstra, A. Kampman, F. Harmelen: Sesame: A Generic Architecture for Storing and Querying RDF, International Semantic Web Conference (2002).
23) The SeRQL query language, rev. 1.1 (Nov. 2004). http://www.openrdf.org/doc/users/ch06.html