組込み分野のための
UML モデル解説書

部品編 C002

プログラム更新機能

本書は、UML モデルカタログに含まれる「プログラム更新機能」のモデルの詳細を記述したものです。モデリングの初心者には教科書や参考書として、モデリングのベテランの方々にはモデルのヒントとして、ぜひともお手元に置いて活用してください。

UMTP 組込み
モデリング部会

2016.5.15 更新

目　次

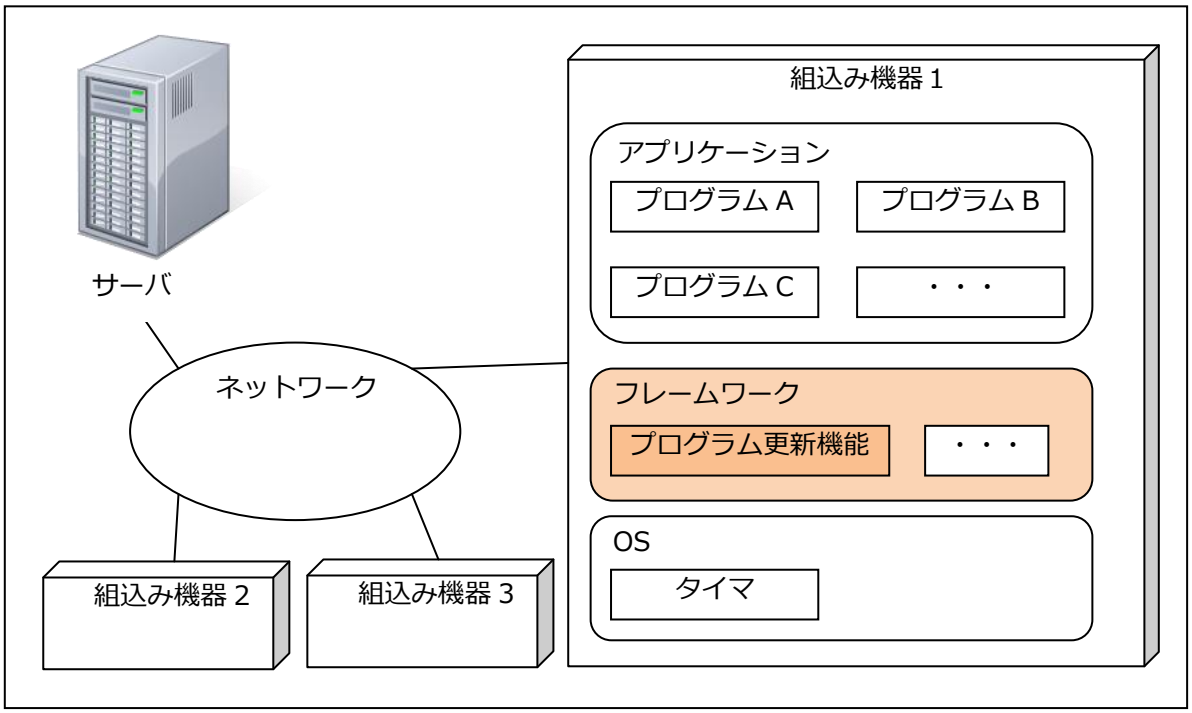
はじめに

プログラム更新機能は、ネットワークを通じてサーバに管理されている組込み機器の複数のプログラムを、依存関係を解決し動的に更新する機能です。今では、自動販売機、スマートメータ、ゲーム機器、携帯電話など、ありとあらゆる製品に搭載されています。

要求仕様

プログラム更新機能の基本的な構成と要求事項

プログラム更新機能を持つ組込み機器のソフトウェアは、OS、フレームワーク、アプリケーションの 3 つのレイヤから構成されているものとします。プログラム更新機能は、フレームワークの一つの機能として存在し、OS が提供する機能を使用しながらフレームワーク上で動作するプログラムを更新します。



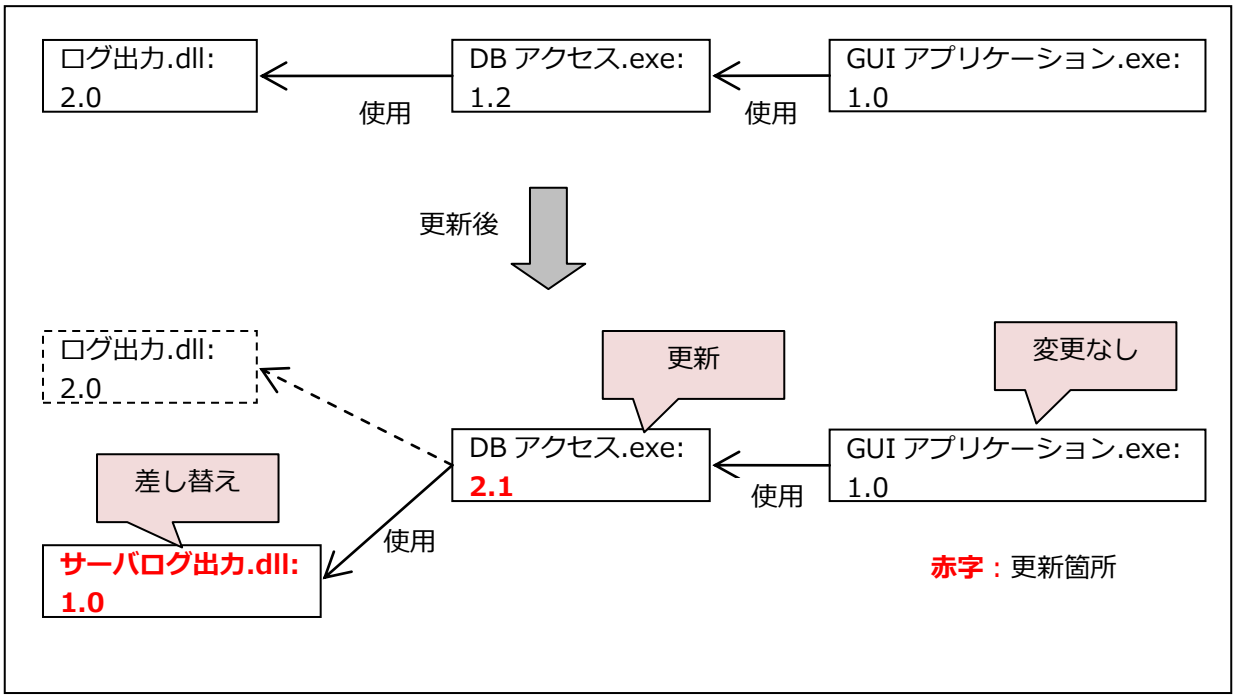
このモデルでは、プログラム更新機能に対する重要な要求事項として、以下のものがあるとします。

- データの通信量を少なく抑えるために、プログラムの差分を圧縮したデータをダウンロードすること。
- プログラムもしくはダウンロードしたデータの正当性を保証するために、署名を用いること。
- 動作中のプロセスをできるだけ止めずにプログラムの更新が可能であること。
- プログラムの更新に失敗した場合に、更新前のプログラムで動作すること。
- プログラム更新の結果をサーバで検知できること。

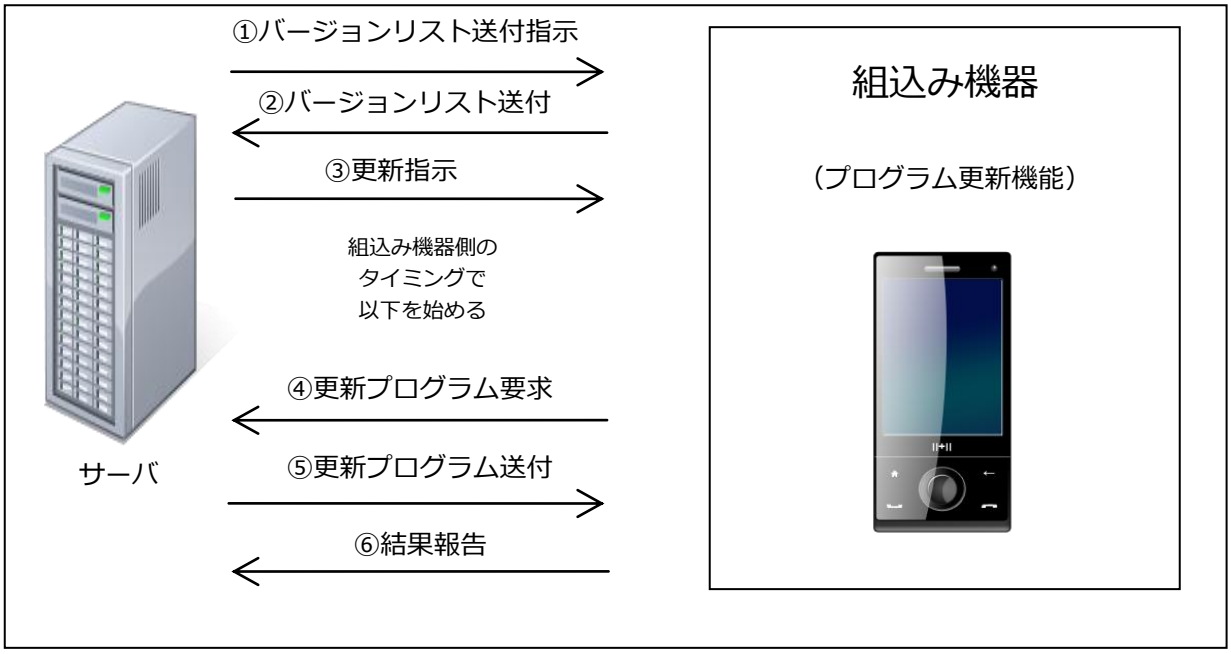
プログラム更新の流れ

ログ出力プログラムを、サーバログ出力プログラムに差し替え（削除と追加）、DB アクセスプログラムを更新する場合を例としてプログラム更新の流れを説明します。

以下の図のように更新を行うとします。



更新の流れは以下の図の通りとなります。



①バージョンリスト送付指示：プログラム更新のトリガーはサーバ側からの場合と組込み機器側の場合の 2 通りの方法が考えられます。双方にメリット・デメリットがありますが、今回は制御が容易であるサーバからのトリガーで行います。

②バージョンリスト送付：サーバに送付するバージョンリストの例です。

プログラム名	更新前バージョン
GUI アプリケーション	1.0
DB アクセス	1.2
ログ出力	2.0

③更新指示：送付されたバージョンリストを元に更新が必要かどうかをサーバ側で判断し、必要な場合にのみ更新を指示します。

④更新プログラム要求：更新プログラムのダウンロードやインストールを行うと組込み機器の CPU やメモリに負荷がかかります。また、更新中は対象プログラムと関係するプログラムを動作させられません。よって、これらを行っても問題ない時間になってから、更新プログラムを要求します。

⑤更新プログラム送付：送付されるダウンロードデータの例です。詳細な形式については付録を参照してください。

プログラム名	更新後バージョン	更新内容	データ
DB アクセス	2.1	変更	XXXX・・・（差分）
ログ出力	(Version なし)	削除	(なし)
サーバログ出力	1.0	追加	XXXX・・・（全体）

⑥結果報告：報告される結果が NG になるケースと、その後の対応は以下となります。

エラーケース	チェックの方法	その後の対応
ダウンロードデータが不正	署名を確認	ダウンロードデータを削除
更新プログラムが動作しない	自己診断を実施	更新後プログラムを削除

データ通信上のエラーは、通信プロトコル層において解決されるものとします。

その他仕様・制約事項

本モデルで検討するプログラム更新機能には、以下の前提条件があるものとします。

【依存関係】

- ◆ 本モデルでは、OS の更新のように組込み機器全体に影響するような更新は想定していません。
- ◆ プログラムの使用関係が双方向になったり、循環したりする関係を禁止とします。これを認めると、プログラムの停止順を決めることができなくなるためです。

【通信】

- ◆ サーバと組込み機器間の通信に関する機能は、プログラム更新機能には直接関係しないため、本モデルの範囲外とします[1]。

【セキュリティ／正当性の確認】

- ◆ 更新後、本当に正しく動作するかどうかをカタログの機能編で公開している自己診断機能を使って検証します。このため、更新のためにプログラムが停止されてから再開されるまでに必要な時間が長くなりますが、これは制限事項とします。
- ◆ セキュリティに関して、考慮される事案は様々です[2]が、すべてを網羅することはできません。本モデルではプログラムに添付された署名と別モデルの自己診断を利用したチェックを行うこととします。
- ◆ プログラムの正当性の検証は、プログラムのハッシュ値と署名者の公開鍵で復号した署名に含まれるハッシュ値を比較することで行います。署名者の公開鍵は、証明書内に含まれています。具体的なアルゴリズムについては、ライブラリ[3]を使用して実現できるものとします。

[1] サーバと組込み機器間の通信に関する機能というのは、具体的には、データ破損、パケットの欠落、パケットの到着順番の逆転などの問題が発生したときのチェック機能や再送機能のことです。これについては、TCP/IP 上で通信を行うことで解決されるため、本モデルの対象外としています。

[2] 組込み機器がネットワークに接続される機器となりますから、そこからの攻撃、すなわち、侵入、踏み台、DoS 攻撃などに備える必要があります。他にも、プログラム間のアクセス権、証明書等で使用する暗号化アルゴリズム（RSA など）の更新も考える必要があります。(http://www.ipa.go.jp/security/event/2011/sympo4/documents/IPA_sympo4_2011_A1.pdf 参照)

[3] 具体的には以下を想定しています。
Java(J2ME CDC/Android)：java.security や java.security.cert パッケージ
C 言語：OpenSSL ライブラリ
Qt(C++)：QtNetwork モジュール内の QSslKey クラスや QSslCertificate クラス

用語定義

すでにいくつかの用語を使用していますが、あらためて用語の定義を行います。

用語	用語の意味
プログラム更新機能	プログラムを更新する機能。
プログラム	更新対象のプログラム。実行ファイル(exe)，動的ライブラリ(dll)[4]だけではなく、設定ファイル、リソースファイル等も想定している。
プロセス	プログラムの動作中のインスタンスで、プログラムの実行状態を保持する。本モデルでは、実行ファイルから生成されるものとする。
バージョンリスト	プログラムのバージョン情報のリスト。プログラム更新機能はサーバ側のプログラムのバージョンと組込み機器側のプログラムのバージョンを比較することで、更新の必要性を判断する。
ダウンロードデータ	更新プログラム要求に応じて、サーバから送付されてくる、更新用のデータ。
署名	ダウンロードしてきたプログラムの正当性を証明するもの。
証明書	署名を検証するために使用する証明書で、公開鍵を含むものとする。
サーバ	更新プログラムを提供するサーバ。

[4] dll とはダイナミックリンクライブラリのことです。

ユースケース

本モデルが実現する範囲のユースケースを示します。

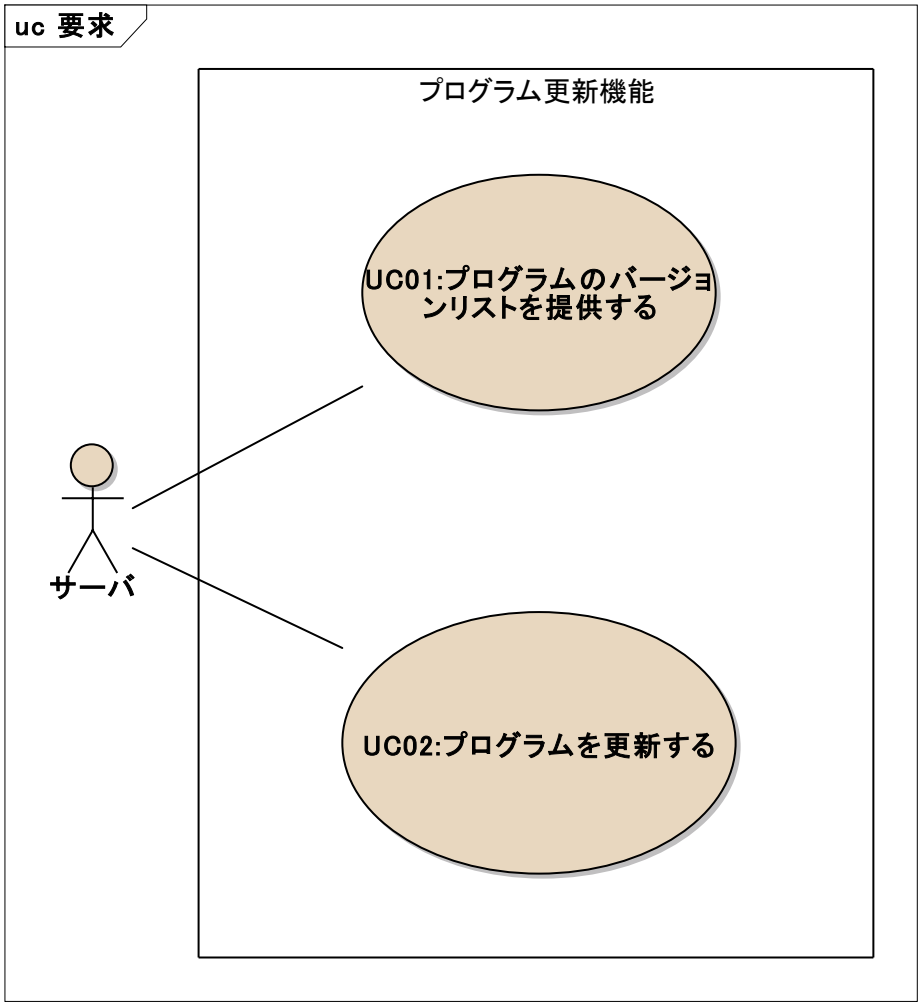


図1

ユースケース一覧

No.	ユースケース名	概要／備考
UC01	プログラムのバージョンリストを提供する	プログラムのバージョンリストをサーバに送る。
UC02	プログラムを更新する	プログラムをダウンロードして更新する。

ユースケース記述

<UC01：プログラムのバージョンリストを提供する>

■概要

サーバにプログラムのバージョンリストを送る。

■アクター

サーバ

■事前条件

・サーバとの通信が確立している。

■事後条件

・サーバがプログラムのバージョンリストを受け取っている。

■メインフロー

1. アクターはシステムにプログラムのバージョンリストの送付を指示する。
2. システムはアクターに現在のプログラムのバージョンリストを送る。
3. UC を終了する。

■代替フロー

なし

■例外フロー

なし

■備考

なし

<UC02：プログラムを更新する>

■概要

プログラムをダウンロードして更新する。

■アクター

サーバ

■事前条件

・サーバはプログラムのバージョンリストから、更新が必要だと判断している。

■事後条件

・更新対象のプログラムが更新されている。

■メインフロー

1. アクターはシステムにプログラムの更新を指示する。
2. システムは更新予定時刻になると、アクターに更新後プログラムを要求する。
3. アクターはシステムに圧縮された差分データ（以後、ダウンロードデータ）を送る。
4. システムはアクターからダウンロードデータを受け取る。
5. システムは署名を使用して、ダウンロードデータの正当性を検証し、差分データに展開する。
6. システムは更新前プログラムと差分データをマージして、更新後プログラムを復元する。
7. システムは更新の影響を受けるプロセスを停止する。
8. システムは更新後プログラムを診断する。
9. システムは現在のプログラムを更新後プログラムに変更する。
10. システムは更新前プログラムを削除する。
11. システムはアクターに更新成功を通知する。
12. UC を終了する。

■代替フロー

なし

■例外フロー

5a.ダウンロードデータが不正だった場合
5a1. システムはダウンロードデータを削除する。
5a2. システムはアクターに更新失敗を通知する。
5a3. UC を終了する。

8a. 診断結果が NG の場合
8a1. システムは更新後プログラムを削除する。
8a2. システムはアクターに更新失敗を通知する。
8a3. UC を終了する。

■備考

更新予定時刻は、真夜中といった組込み機器があまり使われない時間帯に設定されるものとする。

モデル一覧

着目点	コンセプト	ポイント
エンティティに着目したモデル	更新対象のプログラムとその組み合わせというシステム内に存在するエンティティに着目したモデルです。	プログラム間の使用関係を使って、停止するプロセスを特定します。

エンティティに着目したモデル

モデリングのコンセプト

動作中のプロセスをできるだけ止めずにプログラムの更新を行うために、プログラム間の使用関係を管理し、更新対象のプログラムと関連しているプログラムを実行しているプロセスのみを停止します。

使用関係

要求仕様の例について、更新前のプログラム間の使用関係とプロセスとの関係を表すと以下のようになります。

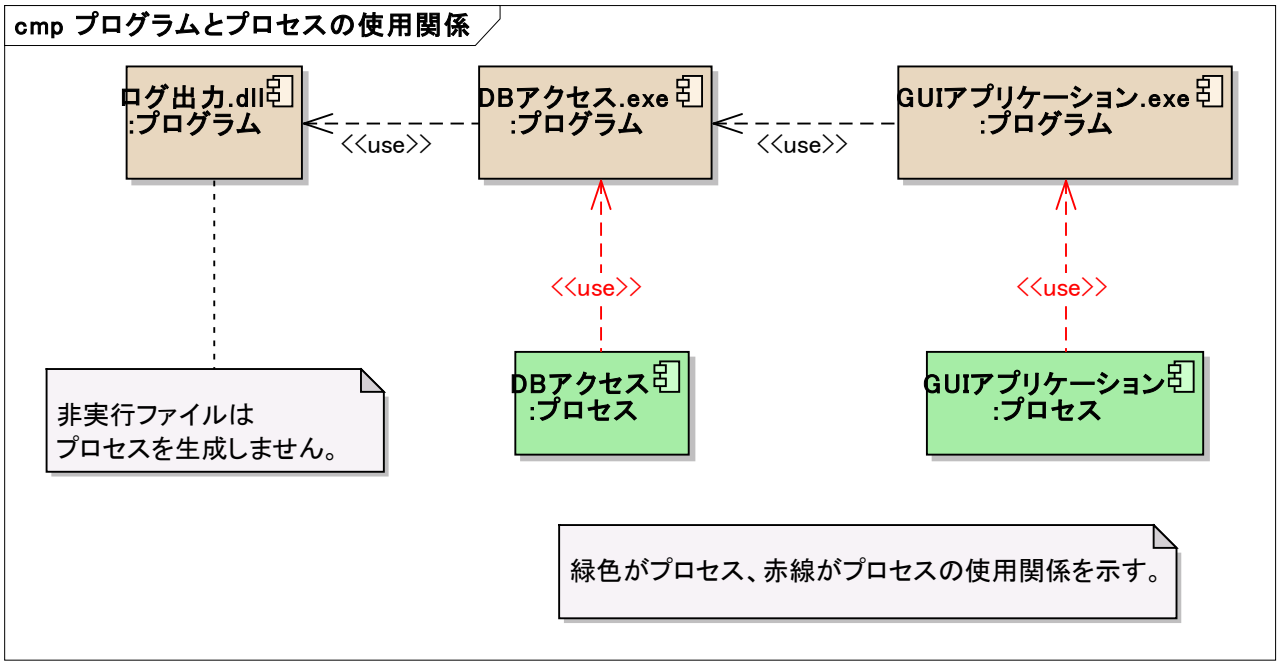


図2

ログ出力.dll をサーバログ出力.dll に入れ替え、DB アクセスプログラムを更新する場合、更新対象を使用しているプロセスは DB アクセスプロセスになります。この DB アクセスプロセスを停止するため、それを使用している GUI アプリケーションプロセスも停止対象になります。

停止する際には、プログラム間の使用関係をたどり、使用元のプログラムに対応するプロセスから順番に停止します。上図の場合は、GUI アプリケーションプロセス、DB アクセスプロセスの順で停止を行うようにします。

プログラム更新機能の動作フロー

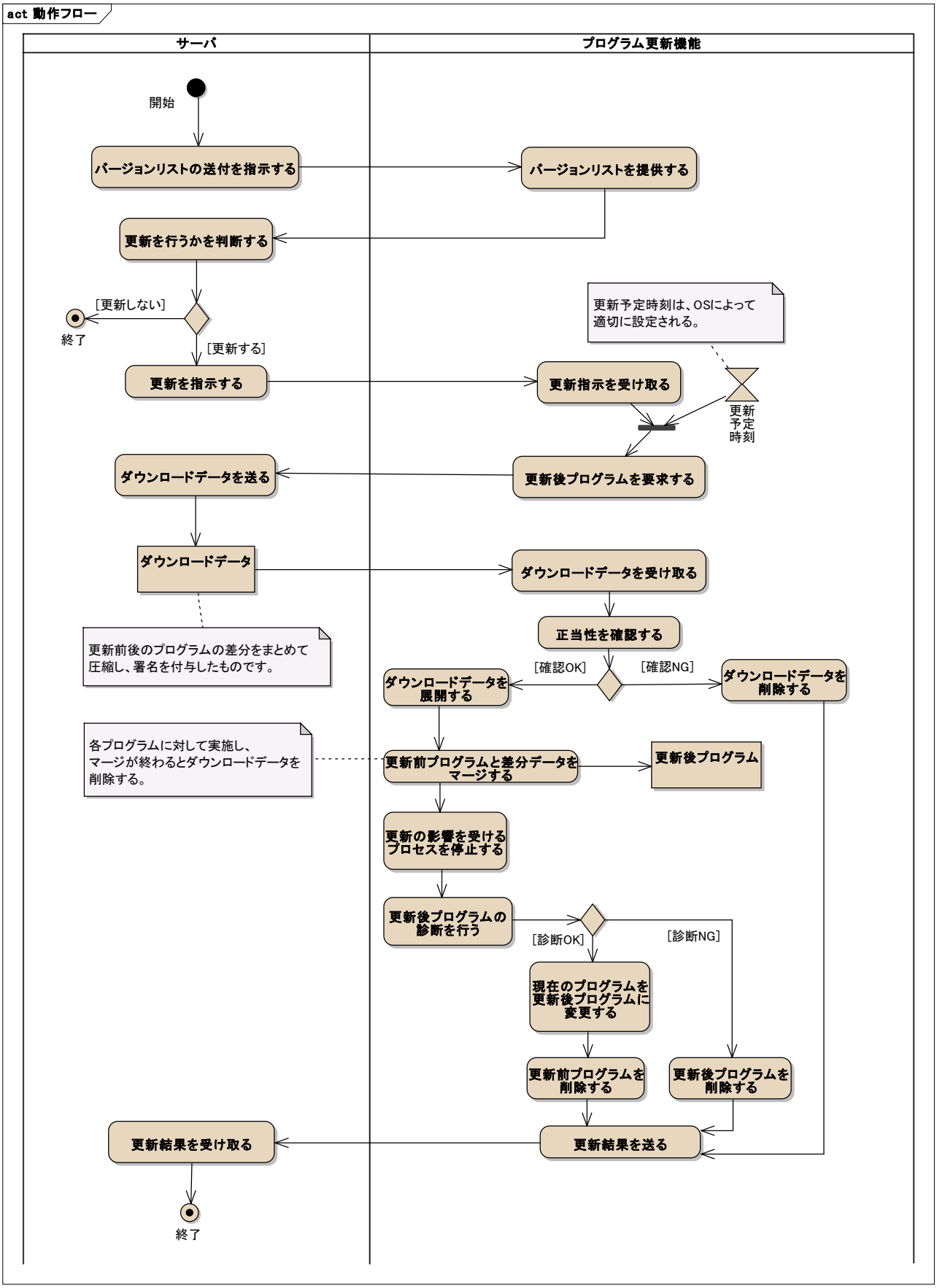


図3

分析モデル

静的モデル

クラス構成

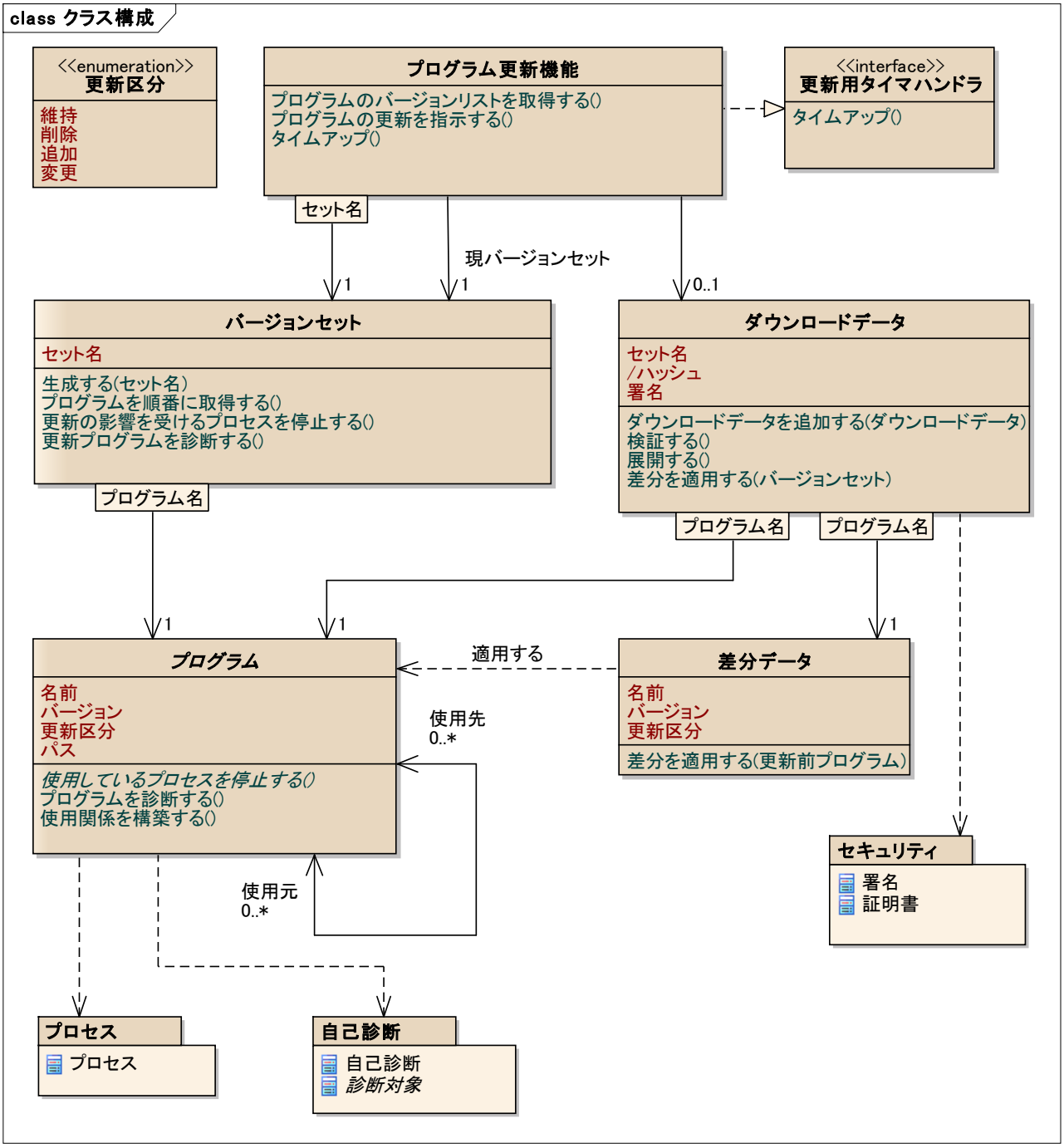


図4

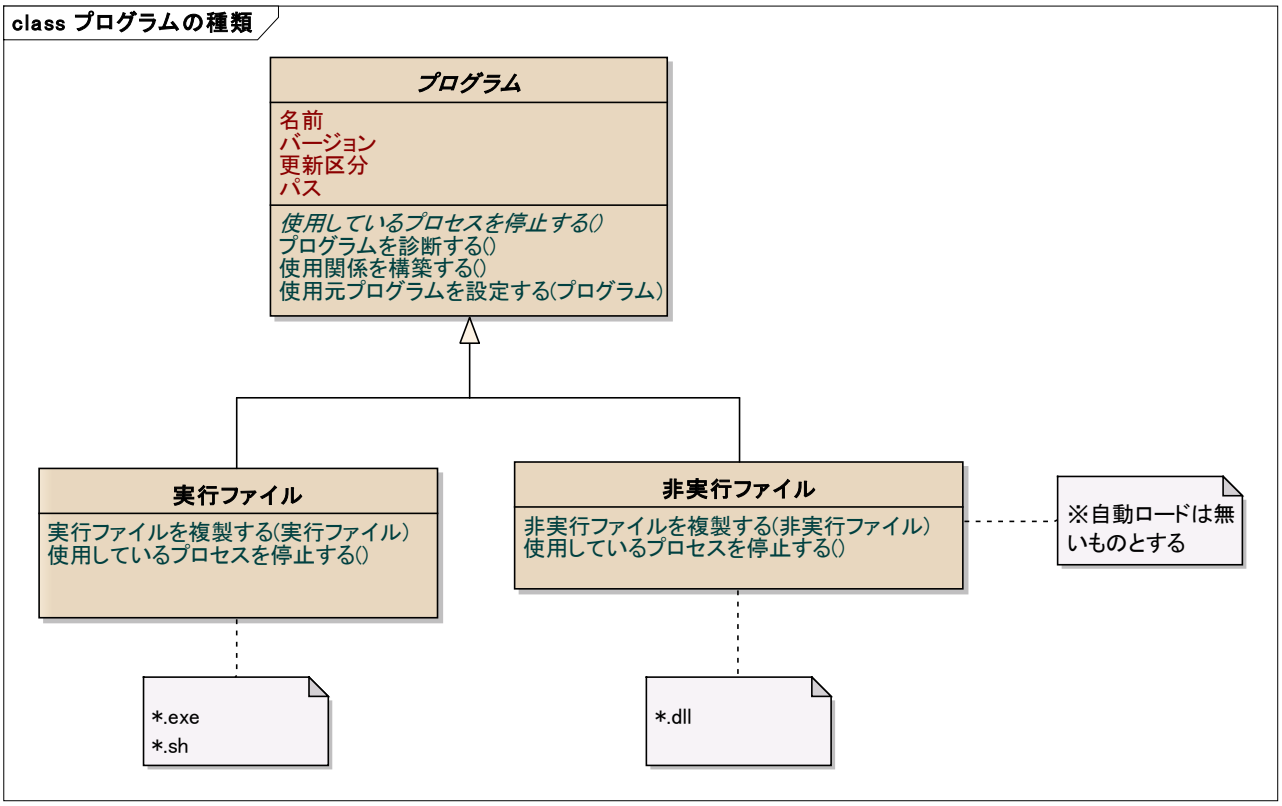


図5

クラス名	説明
プログラム更新機能	プログラム更新機能の中心となるクラスです。1 つのフレームワークに 1 つのオブジェクトが存在します。
バージョンセット	プログラム更新機能が管理するプログラムの集合です。
プログラム	プログラムの特定のバージョンに対応しているクラスです。 exe や dll の親クラスになります。
実行ファイル	プロセスを生成するプログラムです。
非実行ファイル	プロセスを生成しないプログラムです。 動的ライブラリなどが対応します。
ダウンロードデータ	サーバからダウンロードしたデータです。 プログラムの差分を持っています。
差分データ	サーバからダウンロードしたプログラムの差分です。 現バージョンセットのプログラムとマージします。

パッケージ名	説明
セキュリティ	署名および証明書の検証で使用するパッケージです。
自己診断	装置が正しく動作しているかどうかを調べる機能。 自己診断モデルを参照
プロセス	プログラムが実行されているプロセスを管理するパッケージです。プロセスの起動・停止を行います。

更新前のオブジェクト構成

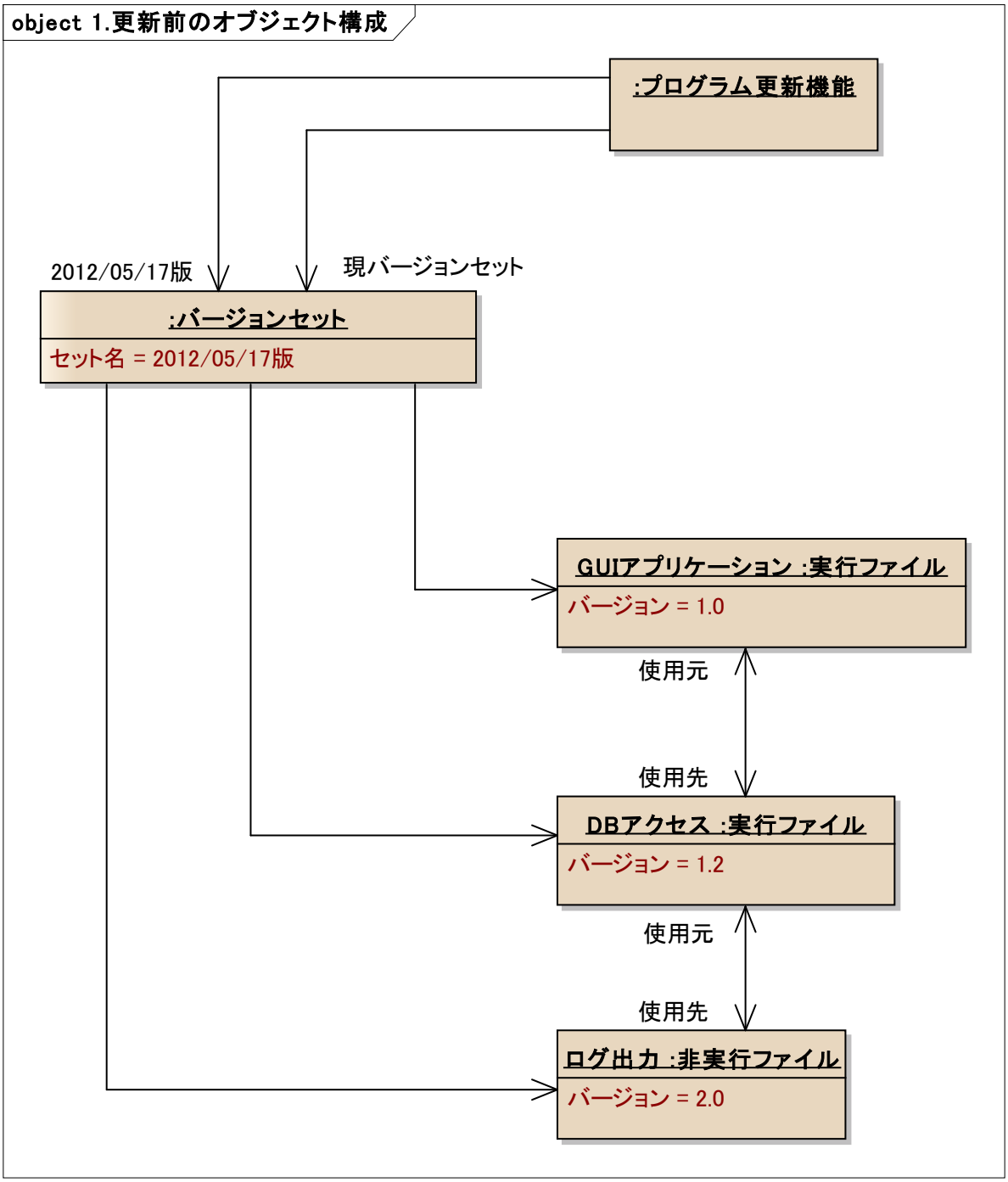


図6

更新途中のオブジェクト構成

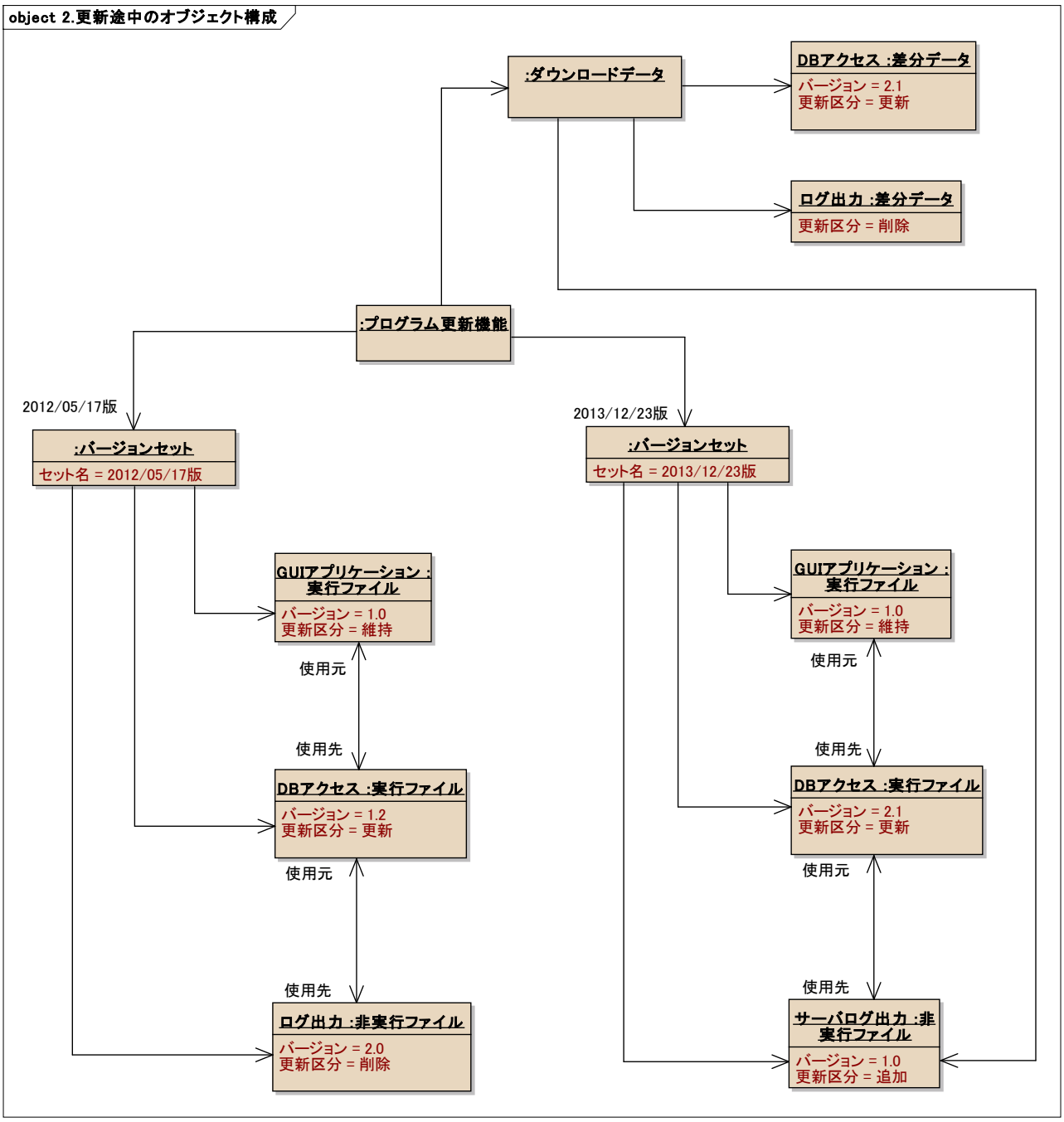


図7

更新後のオブジェクト構成

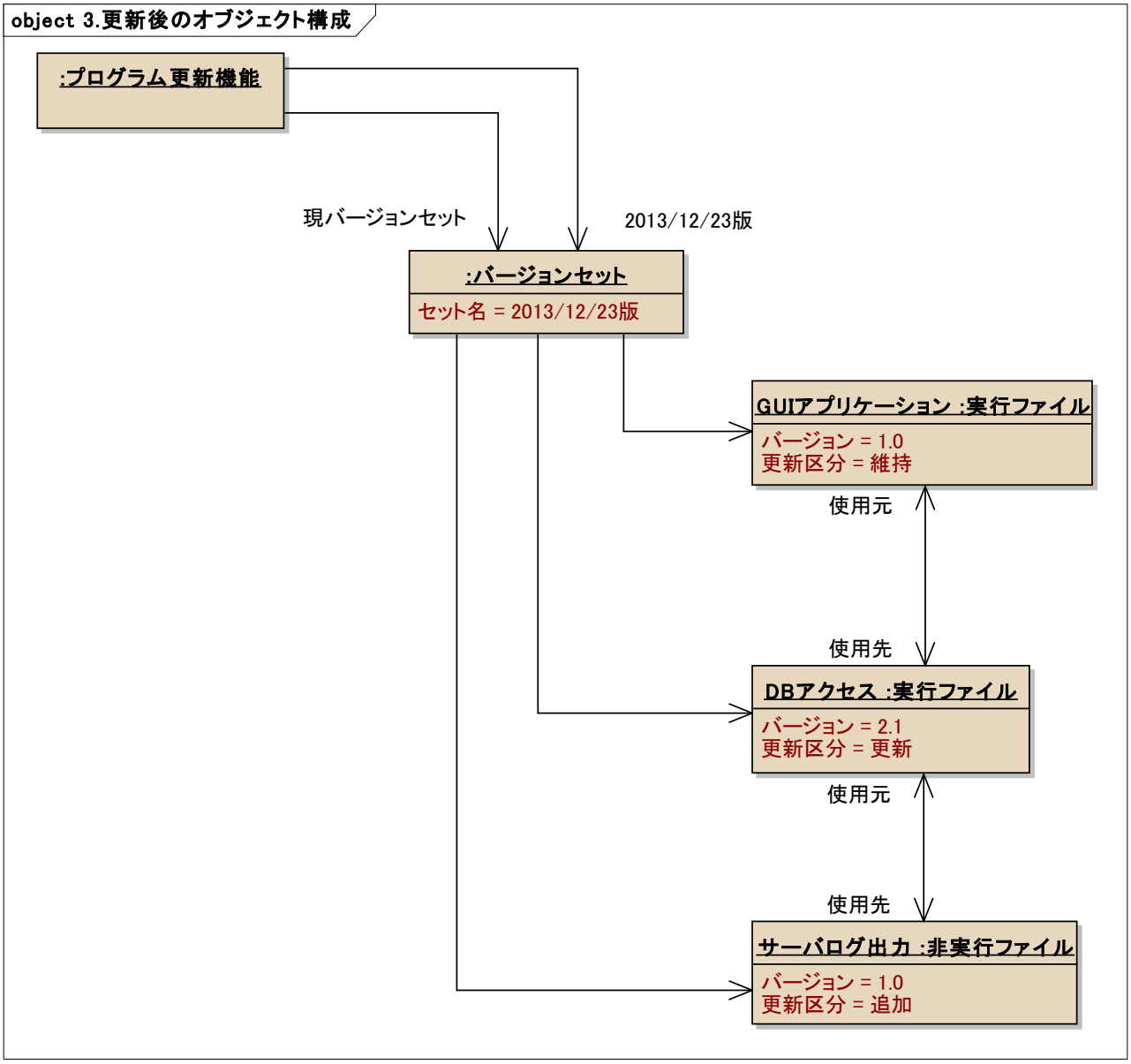


図8

動的モデル

プログラム更新の相互作用

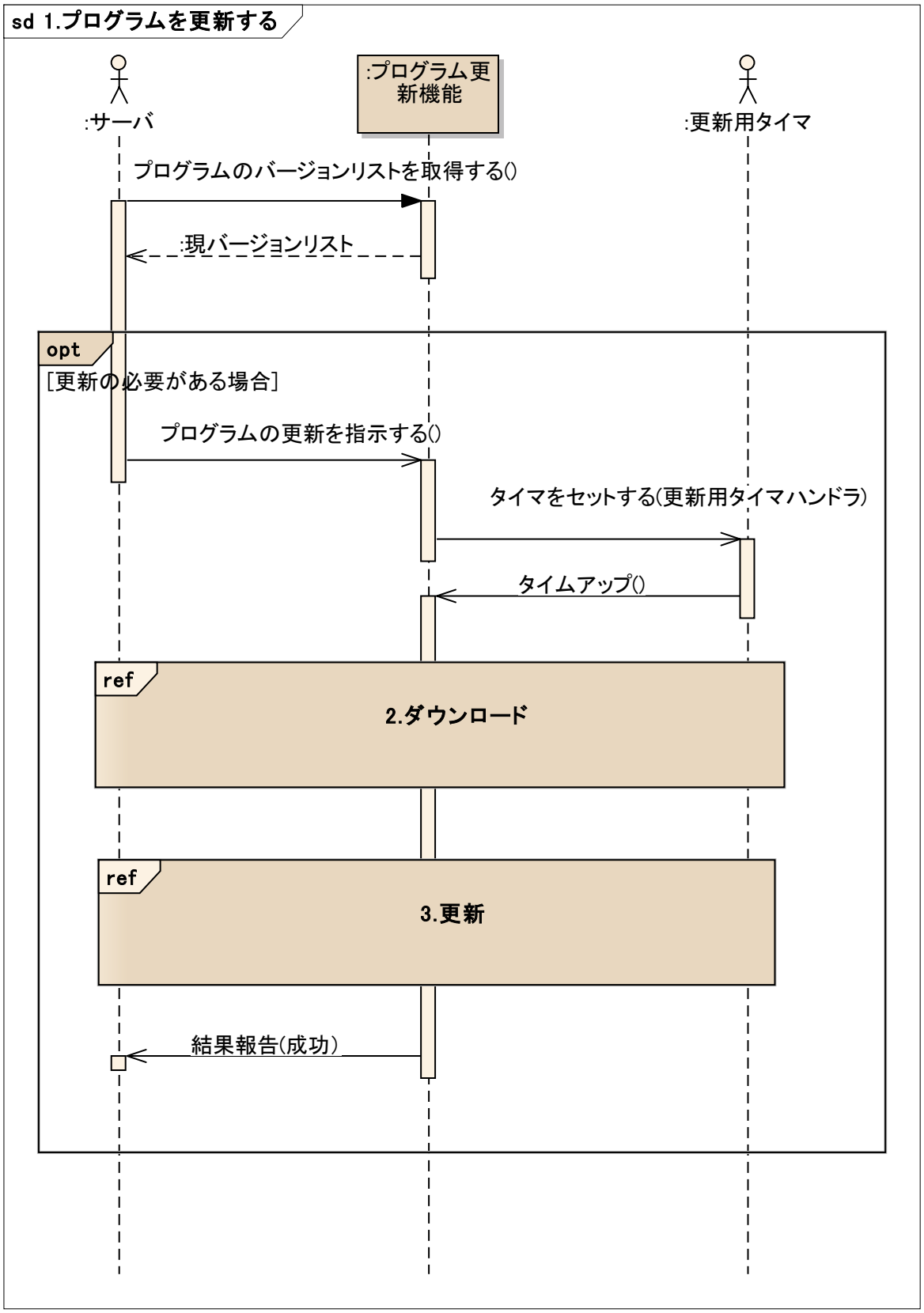


図9

ダウンロード時の相互作用

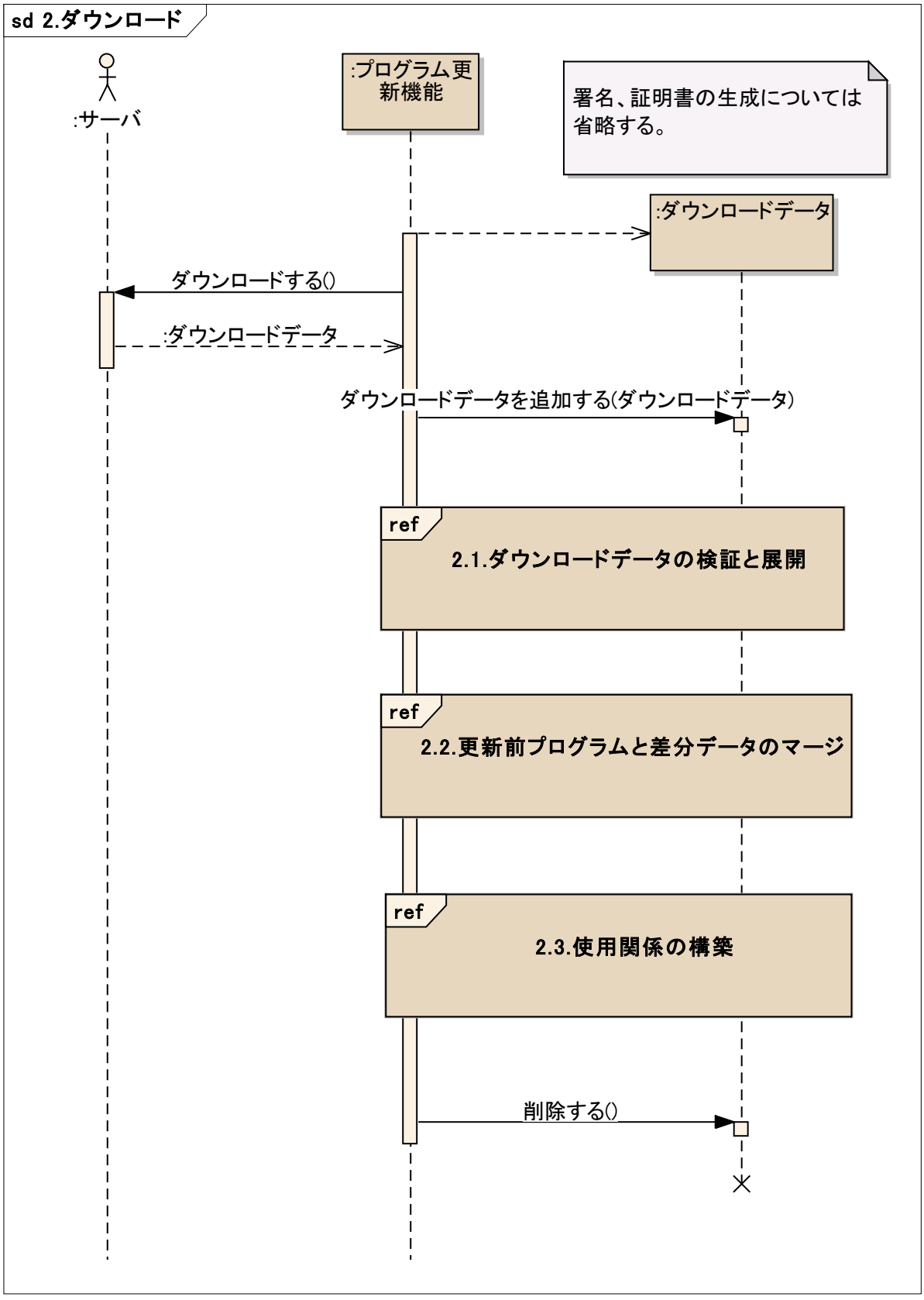


図10

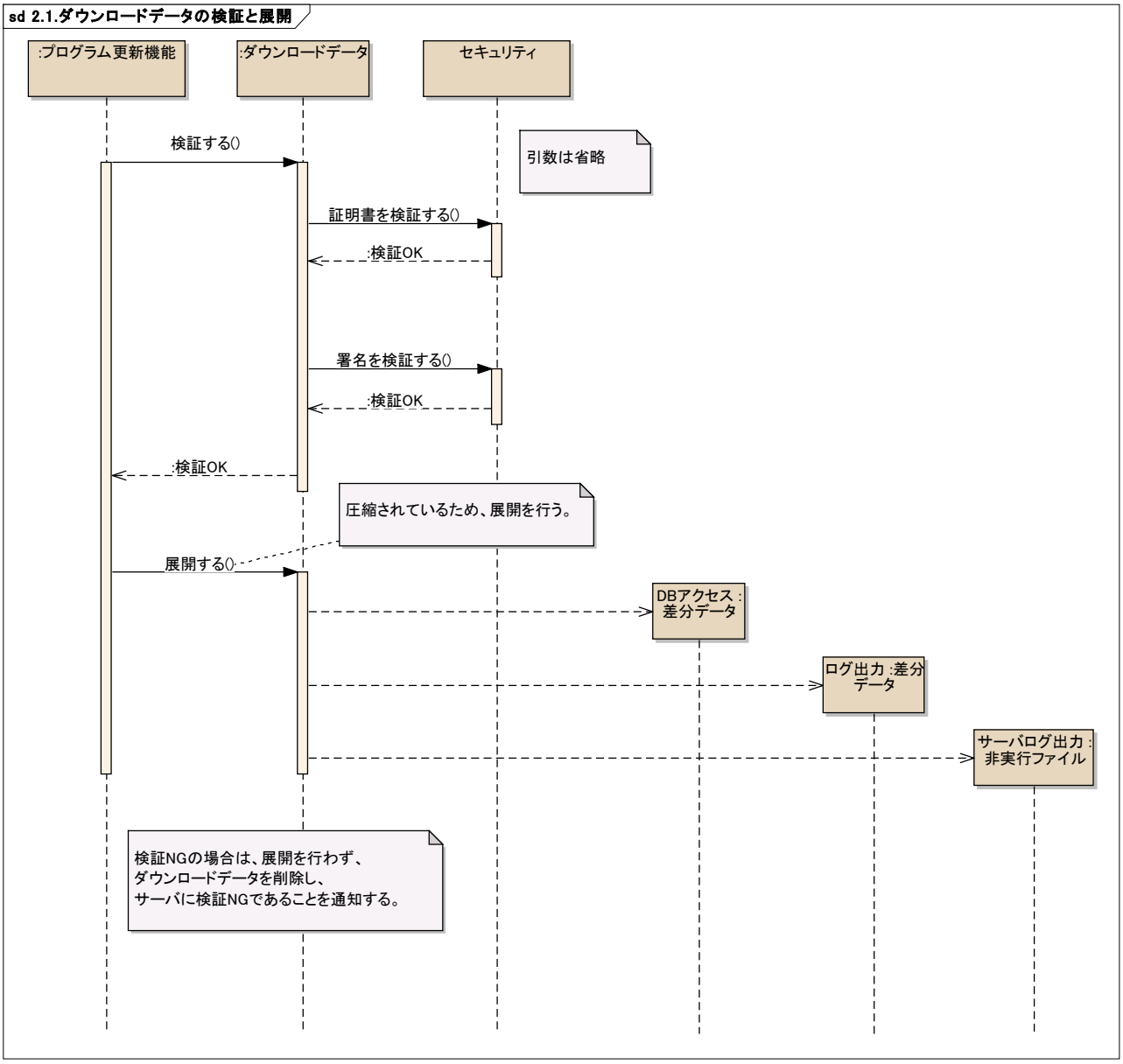

sd 2.1.ダウンロードデータの検証と展開
:プログラム更新機能
:ダウンロードデータ
セキュリティ
検証する()
引数は省略
証明書を検証する()
:検証OK
署名を検証する()
:検証OK
:検証OK
圧縮されているため、展開を行う。
展開する()
DBアクセス : 差分データ
ログ出力 :差分データ
サーバログ出力 : 非実行ファイル
検証NGの場合は、展開を行わず、
ダウンロードデータを削除し、
サーバに検証NGであることを通知する。

図11

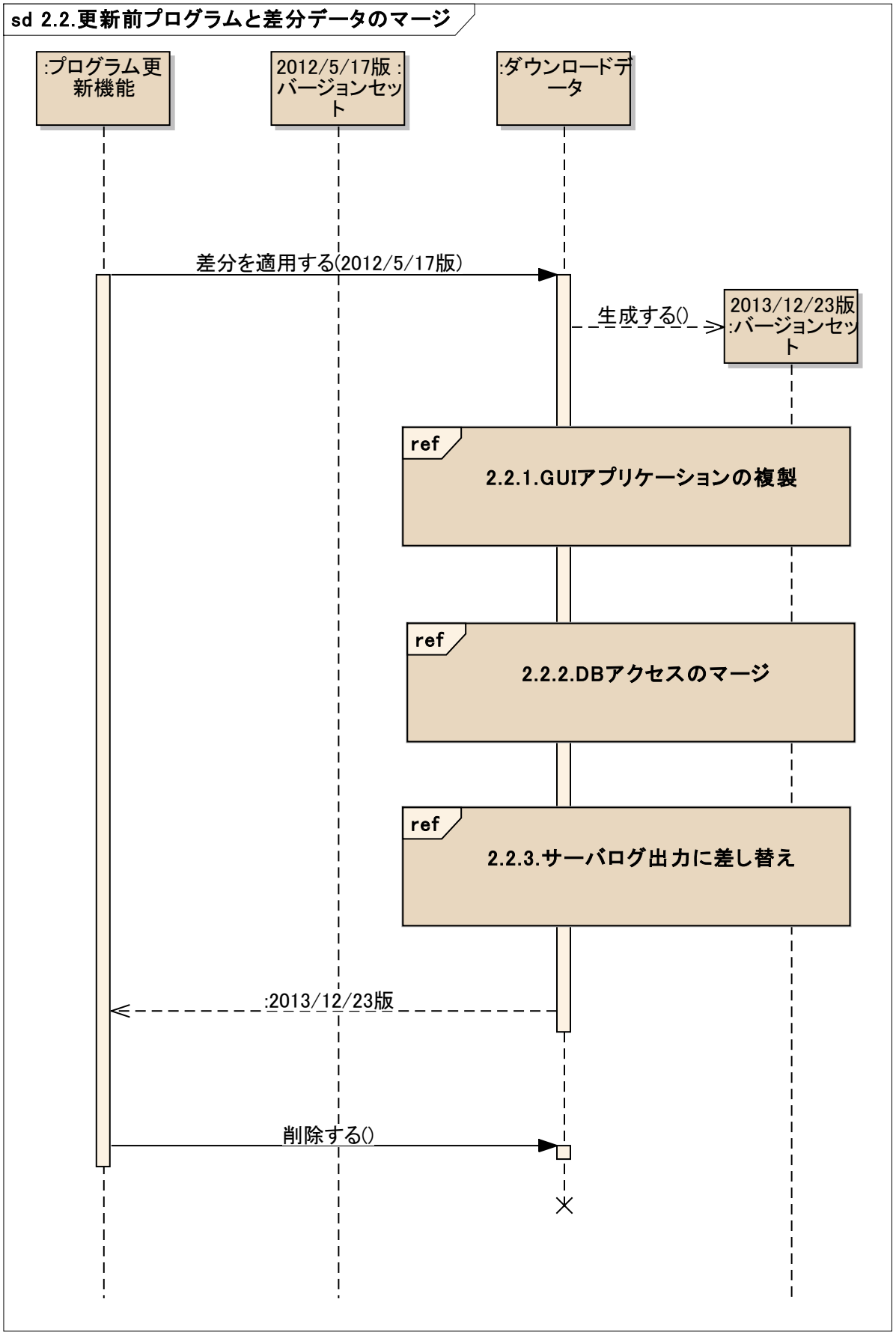

sd 2.2.更新前プログラムと差分データのマージ
:プログラム更新機能
2012/5/17版 :バージョンセット
:ダウンロードデータ
差分を適用する(2012/5/17版)
生成する()
2013/12/23版 :バージョンセット
ref
2.2.1.GUIアプリケーションの複製
ref
2.2.2.DBアクセスのマージ
ref
2.2.3.サーバログ出力に差し替え
:2013/12/23版
削除する()

図12

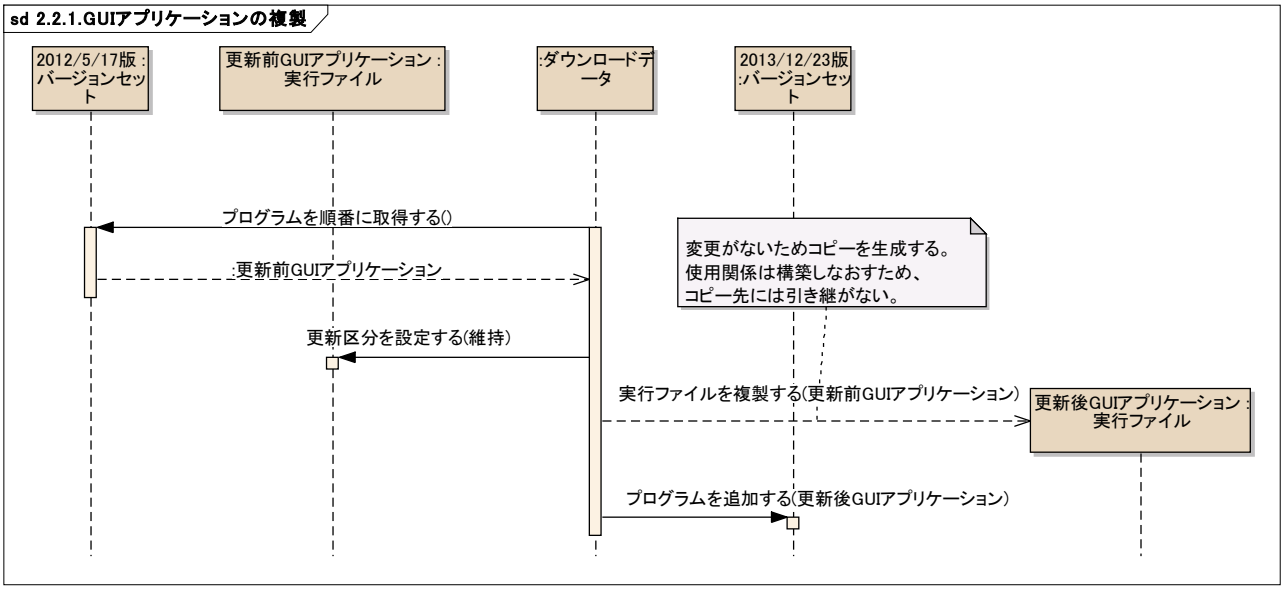

sd 2.2.1.GUIアプリケーションの複製
2012/5/17版 : バージョンセット
更新前GUIアプリケーション : 実行ファイル
:ダウンロードデータ
2013/12/23版 :バージョンセット
プログラムを順番に取得する()
:更新前GUIアプリケーション
変更がないためコピーを生成する。
使用関係は構築しなおすため、
コピー先には引き継がない。
更新区分を設定する(維持)
実行ファイルを複製する(更新前GUIアプリケーション)
更新後GUIアプリケーション : 実行ファイル
プログラムを追加する(更新後GUIアプリケーション)

図13

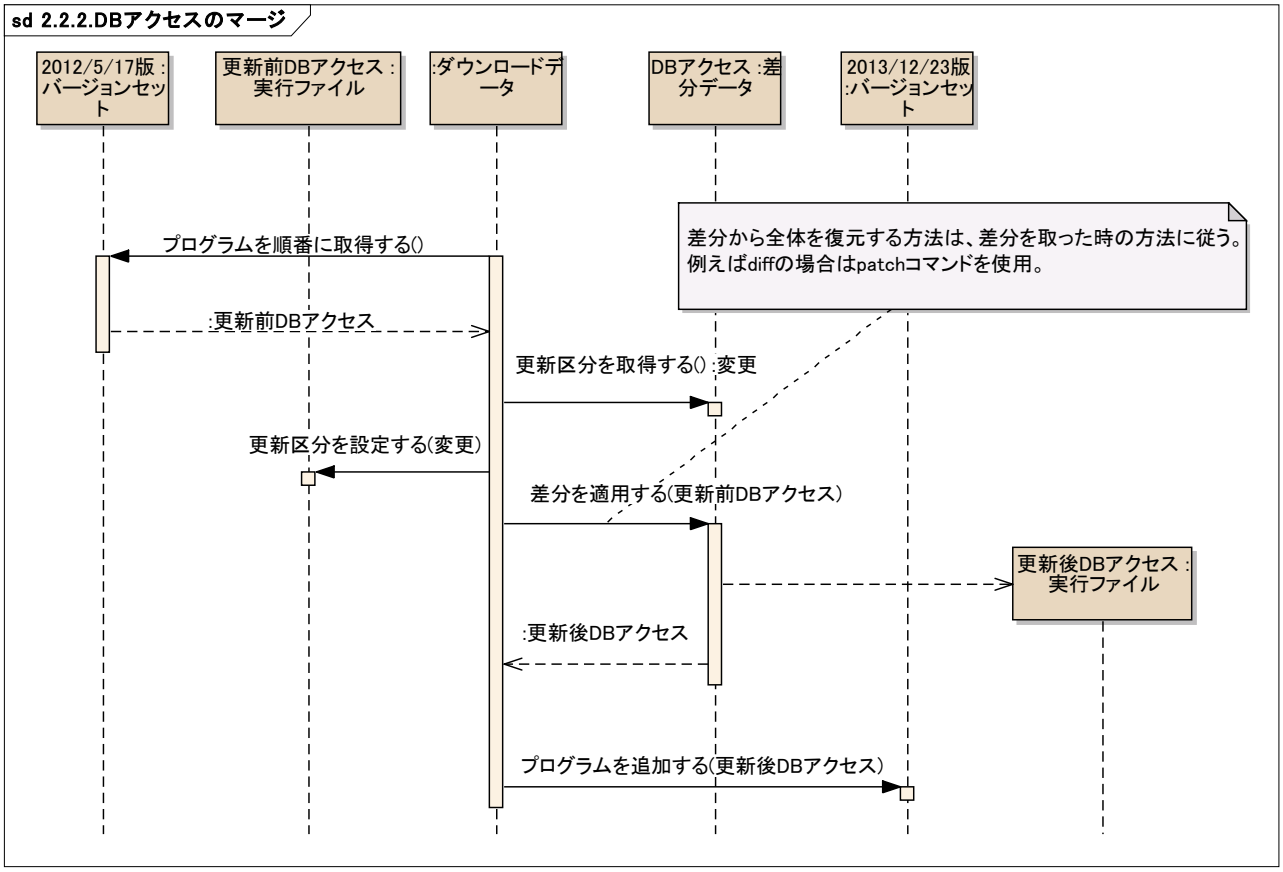

sd 2.2.2.DBアクセスのマージ
2012/5/17版 : バージョンセット
更新前DBアクセス : 実行ファイル
:ダウンロードデータ
DBアクセス :差分データ
2013/12/23版 :バージョンセット
差分から全体を復元する方法は、差分を取った時の方法に従う。
例えばdiffの場合はpatchコマンドを使用。
プログラムを順番に取得する()
:更新前DBアクセス
更新区分を取得する() :変更
更新区分を設定する(変更)
差分を適用する(更新前DBアクセス)
更新後DBアクセス : 実行ファイル
:更新後DBアクセス
プログラムを追加する(更新後DBアクセス)

図14

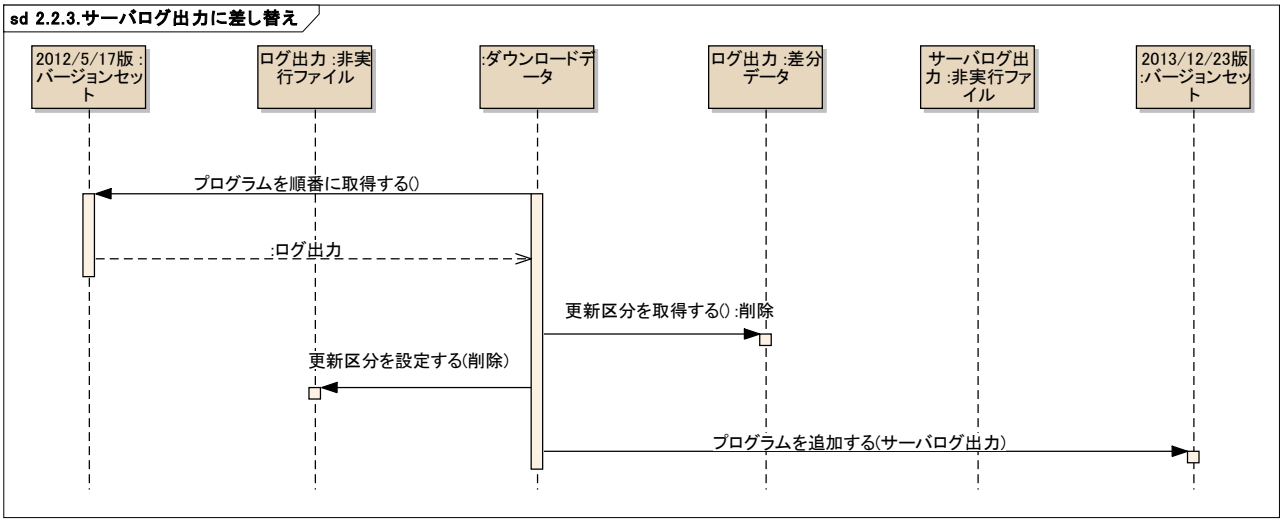


図15

更新前のバージョンセットに更新区分を設定します。これは更新前のプロセスの停止を更新前のプログラムが行うためです。

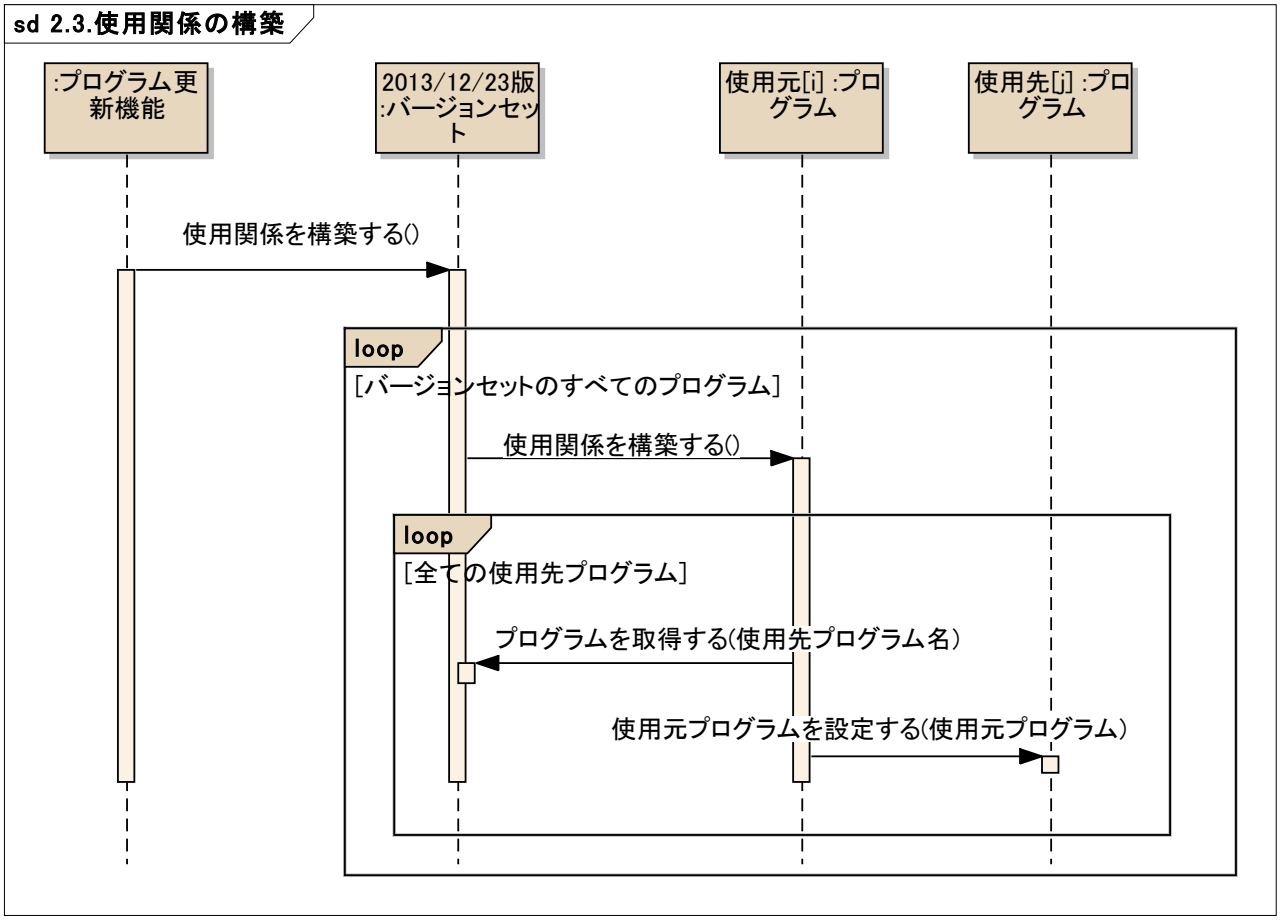


図16

更新時の相互作用

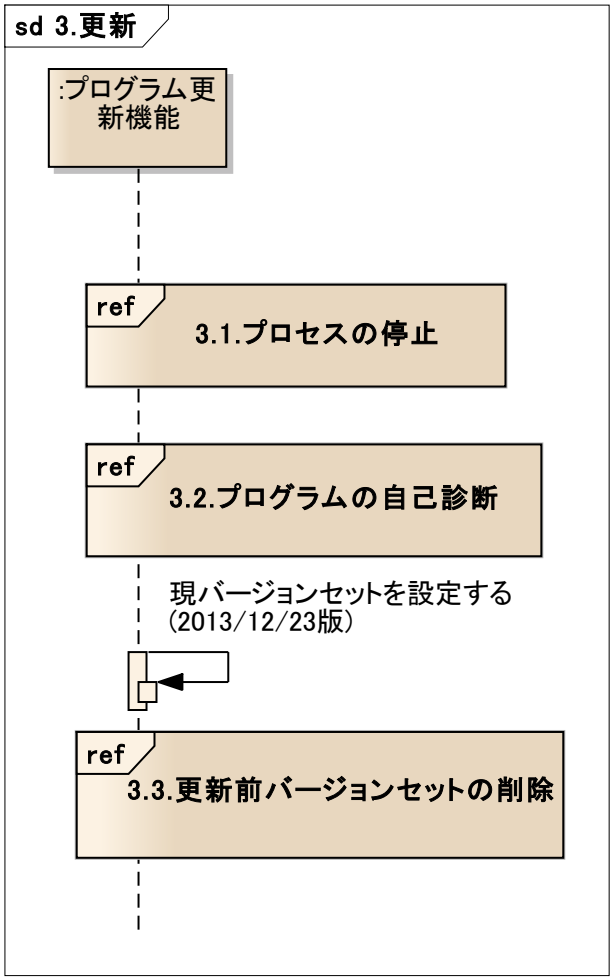
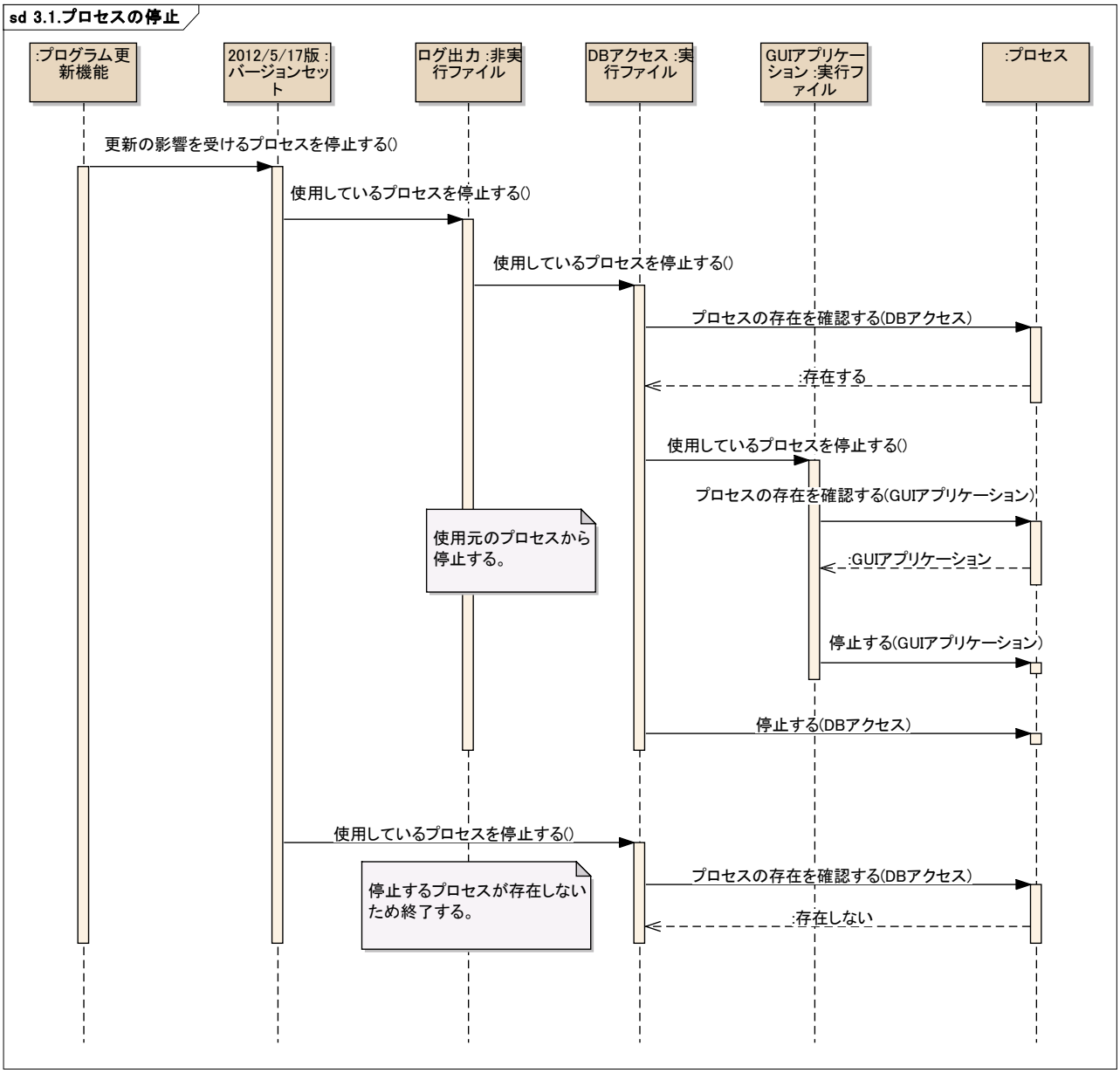


図17



プロセスの停止は、更新前の 2012/5/17 版バージョンセットを元に行います。そのため、バージョンセットが持つ、プログラムに更新区分を設定しています。この場合は、「ログ出力：非実行ファイル」の更新区分が削除、「ＤＢアクセス：実行ファイル」の更新区分が変更になっています。

そのため、「ログ出力：非実行ファイル」と「ＤＢアクセス：実行ファイル」を使用しているプロセスが停止対象となります。「ログ出力：非実行ファイル」を使用しているプロセスを停止することで、「ＤＢアクセス：実行ファイル」を使用しているプロセス停止する際に、該当するプロセスが存在しないことになります。

使用元をたどり、使用元から順番に停止していくことで、使用先が存在しないという問題を回避しています。

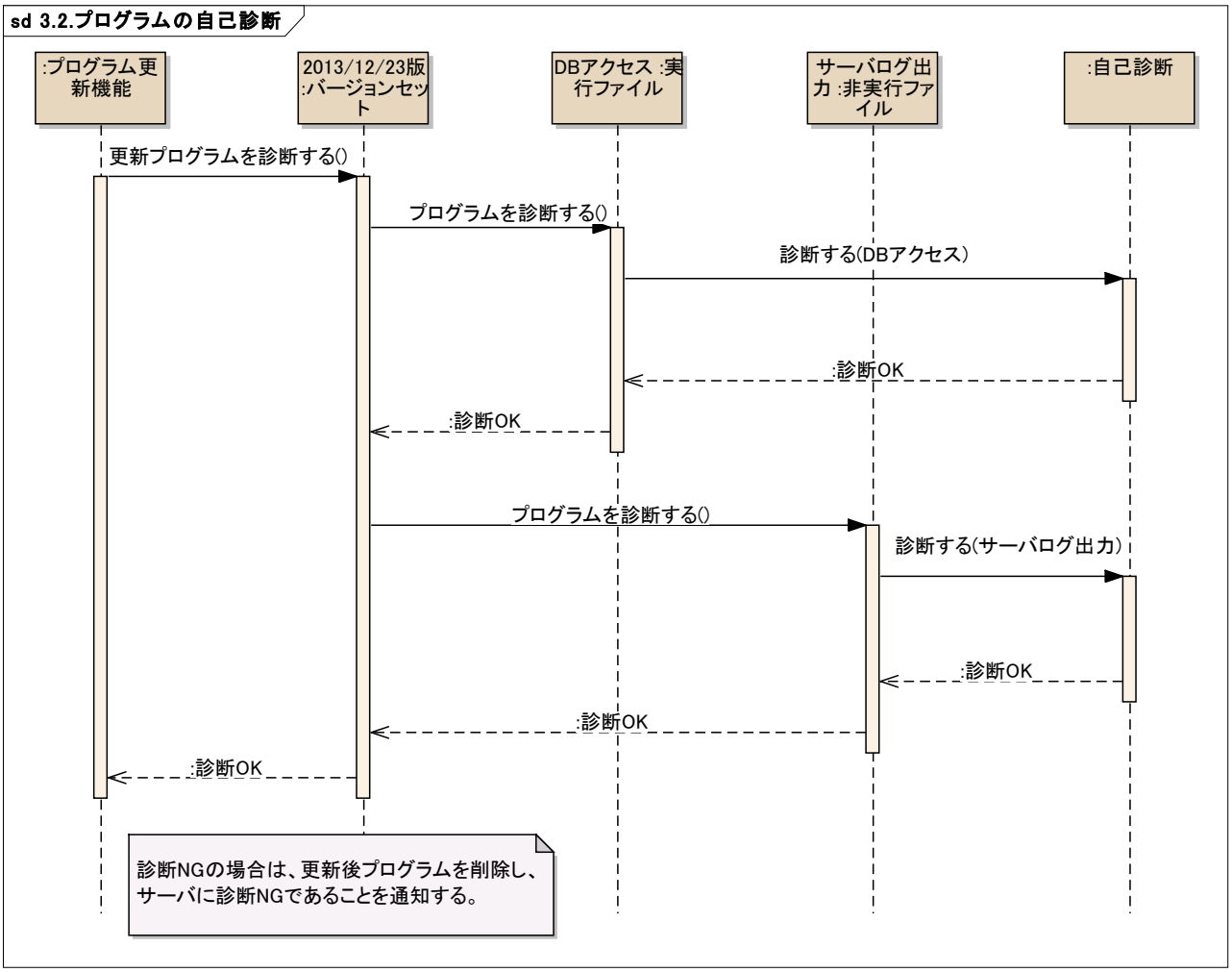

sd 3.2.プログラムの自己診断
:プログラム更新機能
2013/12/23版:バージョンセット
DBアクセス :実行ファイル
サーバログ出力 :非実行ファイル
:自己診断
更新プログラムを診断する()
プログラムを診断する()
診断する(DBアクセス)
:診断OK
:診断OK
プログラムを診断する()
診断する(サーバログ出力)
:診断OK
:診断OK
:診断OK
診断NGの場合は、更新後プログラムを削除し、
サーバに診断NGであることを通知する。

図18

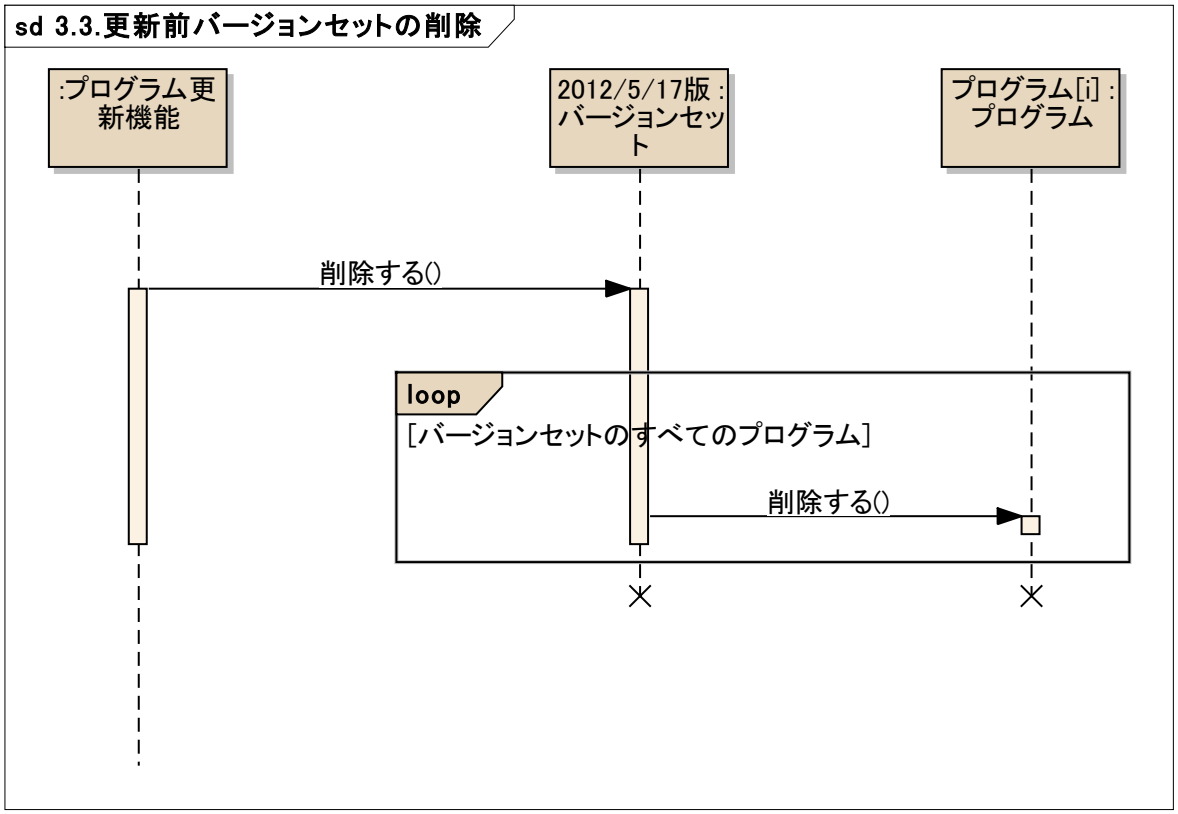

sd 3.3.更新前バージョンセットの削除
:プログラム更新機能
2012/5/17版 : バージョンセット
プログラム[i] : プログラム
削除する()
loop
[バージョンセットのすべてのプログラム]
削除する()

図19

参考文献

◆ 『Java モバイルアプリケーション J2ME で実現するユビキタス・コンピューティング』立川 敬行 著、ソフト・リサーチ・センター、2004

◆ 『IT Text 組込みシステム』阪田 史郎、高田 広章 著、オーム社、2006

◆ OSGi アライアンス “OSGi Alliance | Main / OSGi Alliance” http://www.osgi.org/Main/HomePage

◆ OMA Open Mobile Alliance “Homepage” http://www.openmobilealliance.org/

付録．ダウンロードデータの構成例

```
+contents.txt                    …コンテンツの情報、一覧
+contents                        …コンテンツを格納するディレクトリ
  +DBAccess.2.1-1.2.zip
  +Log.-2.0.zip
  +ServerLog.1.0-.zip
+sig                             …署名データファイル
+cert                            …署名に対する証明書
```

ディレクトリ構成

```
name=2013/12/23 版
U,DBAccesss,2.1,1.2
D,Log,,2.0
A,ServerLog,1.0,
```

Contents.txt の中身

1 行目はバージョンセットの名前

2 行目以降は以下の通り

1 列目：更新区分
（U：更新、D：削除、A：追加）

2 列目：プログラムの名前

3 列目：更新後バージョン(ToVersion)

4 列目：更新前バージョン(FromVersion)

```
+DBAccess.2.1-1.2.patch          …差分（必ずしも差分でなくて良い。）
+DBAccesss.properties            …設定ファイル（使用先のプログラムの名前やその他設定）
```

各プログラムの zip の中身

※　上記は更新の場合の例。追加の場合は「<名前>.<ToVersion>-<FromVersion>.patch」が「<名前>.<ToVersion>.dll」のようになり、削除の場合は存在しない。「<名前>.properties」は追加の場合でも削除の場合でも存在する。

```
#Basic Settings              …基本設定（必須）
Name=DBAccess
Version=2.1
Type=Executable              …プログラムのタイプ（Executable、NonExecutable）
usePrograms=ServerLog    …このプログラムが使用するプログラム(複数あるときはカンマ区切り)
#Other settings              …プログラム特有のその他の設定
DBName=MySQL
DBIP=127.0.0.1
DBEncode=UTF-8
```

DBAccess.properties の中

※　削除の場合は、Version の部分は「Version=」となり、usePrograms の部分も「usePrograms=」となる（すなわち、キーに対する値が無い状態となる）。

プログラムのタイプは、「Executable」、「NonExecutable」のいずれか。

- ◆ Executable：実行ファイルなど。使用する際にプロセスが生成される。
- ◆ NonExecutable：非実行ファイルで動的ライブラリやリソースファイルなど。使用する際にプロセスが生成されない。

```
Signature Algorithm: Sha256withRSAEncryption    …SHA256 ハッシュを RSA 秘密鍵で暗号化
Signature: ZCAKhGvXxAkIqdwAsJQa/E3Tj+SlJzA16sOLAM4rkls= ・・・
```

sig の中身

「contents.txt」と「contents」配下全ての内容（バイナリデータ）からハッシュ値を計算し、証明書に結びついている秘密鍵で暗号化し、BASE64 エンコードしたもの。

```
(省略)
```

cert の中身

署名を検証するための公開鍵、証明書の発行者などの情報をもつ。また、証明書には発行者の署名が付与される。

組込み分野のための UML モデルカタログ
「プログラム更新機能」

初版発行 ２０１４年（平成２６年）１０月１日
発行者　UMTP, Japan
編　著　組込みモデリング部会
印刷

UMTP, Japan
東京都渋谷区代々木 1 丁目 22 番 1 号
http://www.umtp-japan.org/