組込み分野のための
UML モデルカタログ

組込み分野でモデリングする全ての人に捧げるモデル集

本書は、組込み分野のさまざまなモデルを集めたカタログです。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
組込み分科会

目　次

UML モデルカタログの歩き方

ここでは、みなさんがこのカタログを有効活用できるように、カタログの全体構成とお勧めの使い方を具体的にご紹介します。

UML モデルカタログとは

昨今の組込み分野における開発規模の増大・複雑化・短納期化に対応する手法として、UML によるモデリングが成果を上げています。UML モデルを利用することで、大規模システムであっても、全体をふかんし、アーキテクチャレベルでシステムを理解したうえで各種の問題に対する対策を打てるので、生産性・品質を向上させることができます。

しかし、実情は UML モデリングを開発に取り入れてみたものの、モデリングの有識者がいないがために、その効果を十分に得ることができず、従来の方法に戻してしまう現場が多々あります。

そのような状況を打開するために、初心者には UML モデリングに取り組む第一歩として、ベテランの方にはより良いモデリングのヒントとして、「カタログ」(本書)を提供することにしました。

本書が開発の現場で UML モデリングを利用する際の参考になれば幸いです。

本書の適用範囲は、主に、製品開発におけるソフトウェア設計工程です。「組込みソフトウェア向け　開発プロセスガイド[1]」（以下 ESPR）では、右図で示す範囲です。

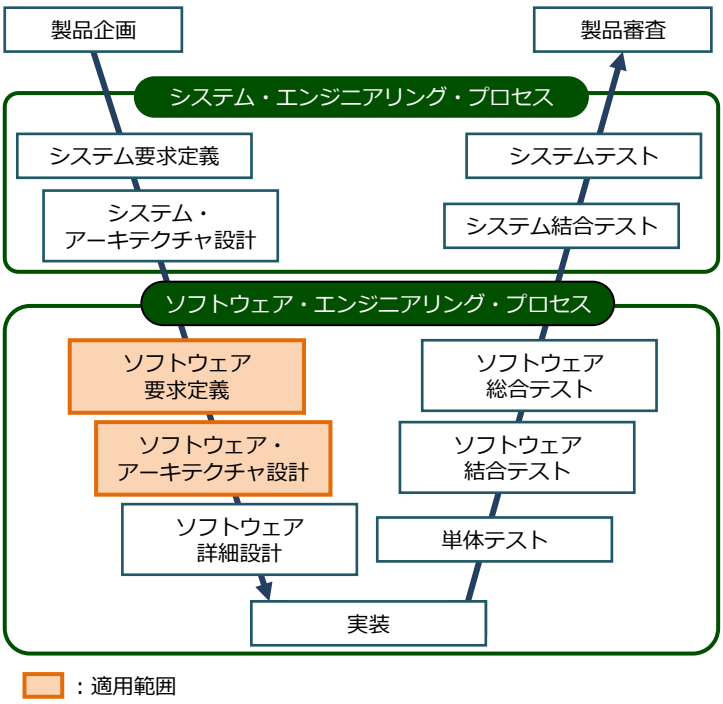


図 1 組込みソフトウェア開発プロセスと本書の適用範囲

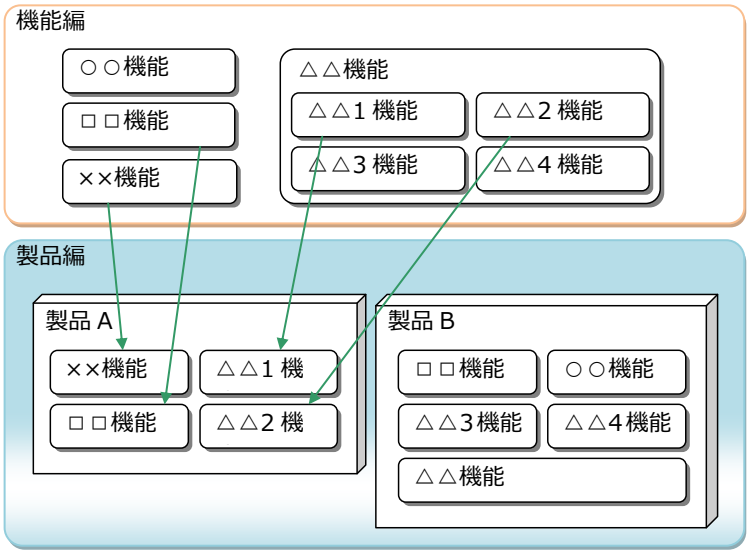
特に本書に関係の深い工程についての定義を以下に示します。

工程名	定義
ソフトウェア 要求定義	当該製品を実現するためにソフトウェアとして実現が必要となる要求を明確にします。
ソフトウェア・ アーキテクチャ 設計	開発する組込みソフトウェアのアーキテクチャ（＝動作[振る舞い]と構造）を決定します。 ※後述する「PIM」設計モデルにあたります。

[1] 書名　：組込みソフトウェア向け開発プロセスガイド
編著者　：独立行政法人 情報処理推進機構 ソフトウェア・エンジニアリング・センター
出版社　：株式会社翔泳社、発行年：2008 年
URL　：http://sec.ipa.go.jp/publish/index.html

カタログの構成

カタログは、「製品編」と「機能編」に分かれています。
「製品編」は、印刷機など製品全体をモデリングして掲載しています。
「機能編」は、多くの製品に共通して搭載されている機能からいくつかをピックアップして掲載しています。
「機能編」に掲載している機能をいくつか組み合わせることで、「製品編」に掲載している製品ができあがる場合もあります。

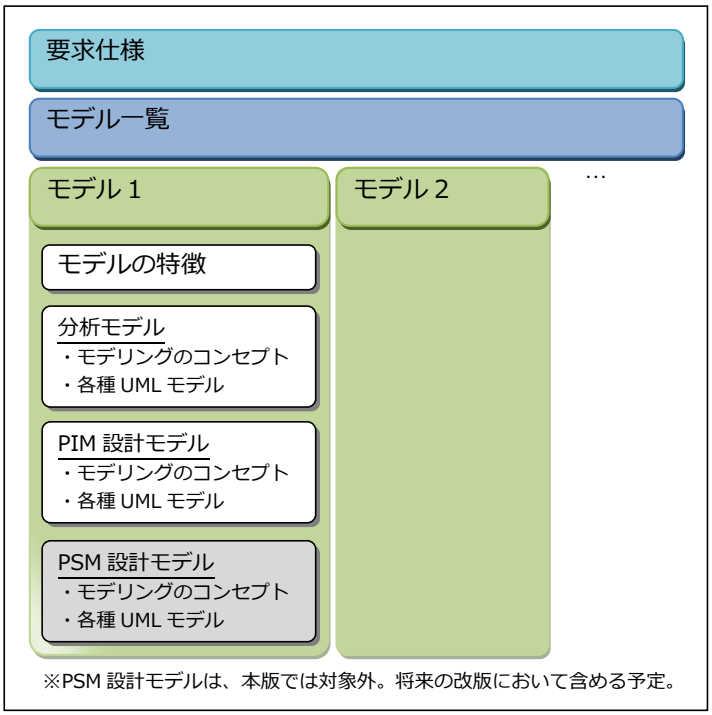


本カタログに掲載しているモデルでは、まず、モデリングの対象となる製品や機能の要求仕様が定義されています。この要求仕様をもとにモデリングを行います。

本カタログでは、1 つの要求仕様に対して、複数のモデルを掲載しています。モデリングは大きく 2 つの考え方があり、1 つは要求仕様を何かに例えて本質を見極める「メタファ」を使った方法、もう 1 つは要求仕様の整理整頓の仕方や重点を置くポイントに「着眼点」を持ったモデリング方法です。

要求仕様の後に、これら複数のモデルのインデックスとなるモデル一覧を用意しています。

各モデルでは、分析モデル、PIM 設計モデル、PSM 設計モデルといった具合に、段階的に詳細化を行っており、それぞれクラス図、オブジェクト図、コミュニケーション図、ステートマシン図などの UML モデルを掲載して、モデルの解説を行っています。



セクション	記載内容	ESPR におけるフェーズ
要求仕様	ユースケース図・ユースケース記述・シナリオなどを使用し、製品または機能の要求仕様を定義します。	ソフトウェア要求定義
分析モデル	モデリングのコンセプトをもとに作成したモデルです。	
PIM 設計モデル	OS やミドルウェアや言語などのプラットフォームに依存しないモデルです(Platform-Independent Model)。	ソフトウェア・アーキテクチャ設計
PSM 設計モデル	プラットフォームに特化したモデル(Platform-Specific Model)です。 ただし、今回はモデルカタログの対象外です。	

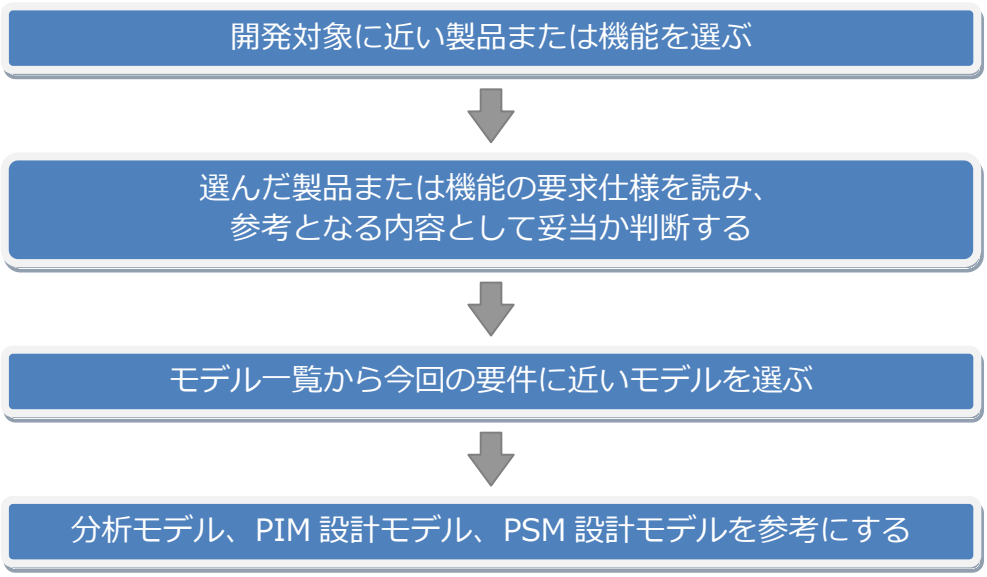
カタログの使い方

本書では、次のような場面で利用されることを想定しています。

- ◆ 新しい製品を開発することになり、製品全体のモデリングの参考として・・・
- ◆ 既存製品に新しい機能を追加することになり、その機能のサンプルとして・・・
- ◆ 既存製品の既存機能を改善するために、比較サンプルとして・・・
- ◆ 組込みシステムにおけるモデリング教育の参考資料として・・・

利用例 1

新しい製品の開発や既存製品への機能追加のサンプルとして利用する場合、下記のような流れで利用します。



例えば、「コピー機」を開発する場合、製品編から「コピー機」のモデルを探しますが、見当たりません。このような場合、製品概要やキーワードからコピー機と「似た」製品を探します。その結果、「孔版印刷機」を選ぶことにします。

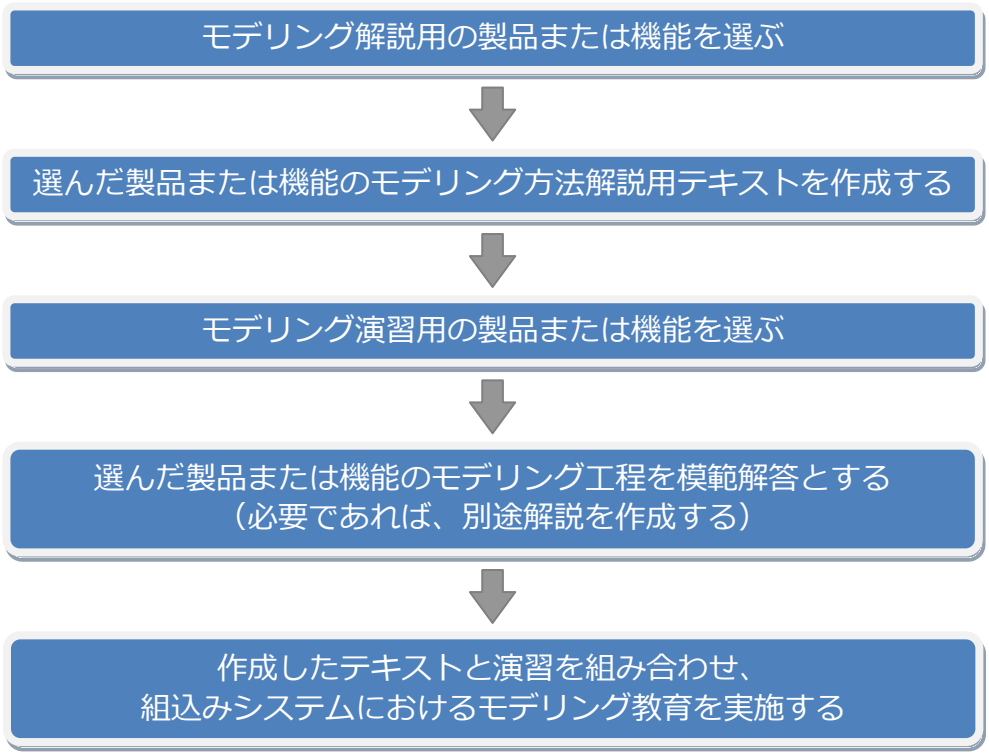
「孔版印刷機」と「コピー機」には、「読み取り」、「書き込み」、「自己診断」等の共通した機能があり、これらがどのようにモデリングされるかを参考に開発ができます。

さらに、「自己診断」については、機能編にその詳細が記述されていますので、機能編の「自己診断」で掲載されているモデルも参考にするとよいでしょう。また、「コピー機」に必要な機能が、参考とする「孔版印刷機」にない場合、機能編を探してみると、その機能が掲載されているかもしれません。製品を構成する機能の多くは、同様の製品に搭載されていることが多く、本書においても機能編でより詳細なモデリングを行っている場合がありますので、機能編も合わせて参考にしてください。

各モデルは、全てを取り入れる必要はありません。必要な部分を取り入れて、開発しようとする製品に合わせて、うまくカスタマイズしてください。

利用例 2

組込みシステムにおけるモデリング教育の参考資料としては、モデリング技法の解説や、演習問題の題材として、以下のような利用ができます。



例えば、組込みソフトウェアのモデリング技法を演習付きで習得できる講座を作るとすると、以下のような流れになります。

まずモデリング解説のためのテキストを作成します。

その際に活用するモデルとして、「自己診断」のモデルのうち「メタファを使って分析したモデル」を選びます。ここでは、モデルの作成手順や考えが詳細に記述してあるものから選択すると良いでしょう。モデリングを解説するテキストでは「自己診断」の「メタファを使って分析したモデル」に記述された本書のモデルが、どのような考えに基づいてされたのか、行間を埋めていくイメージで作成します。例えば、本書のモデルでは、概念モデルとしてのクラス図に続いてシーケンス図が記述されています。テキストでは、概念モデルとしてのクラス図を作成した後には、クラス図には表現しきれない振る舞いを、シーケンス図で表現することや、クラス図のクラスとシーケンス図のクラスの対応から、具体的にどのようにクラス図からシーケンス図が作成されたのかを解説します。

次に、演習を作成します。演習の課題としては「認証」を選びます。ここでは、モデリング例が多いものを選択すると良いでしょう。

必要であれば、モデリング解説用のテキスト作成時と同様に、「認証」に対するモデリング例から模範解答を選び、解説を作成します。以上を準備として、講義を実施します。

モデルカタログ

ここからはモデルカタログになります。

ラインナップ

製品編

製品名	製品概要	キーワード※
孔版印刷機	低コストに大量の印刷物を作成することを実現する印刷機	孔版印刷 読み取り 書き込み 紙搬送 機器管理 ユーザインタフェース 自己診断 認証

※下線は機能編に記載があります。

機能編

機能名	製品概要	キーワード
認証	ユーザを識別し、ユーザ毎に適切なサービスを提供したり、記録を取ったりする機能	ユーザ認証 サービス実行承認 サービス利用状況監視
自己診断	組込みシステムにおいては、システムを構成するデバイスのチェックを行い、結果をレポートする機能。チェックする項目、チェック項目の実行順序、実行する手段(自動、手動)などをモデリングしています。	装置診断

製品編

孔版印刷機

ご注意：

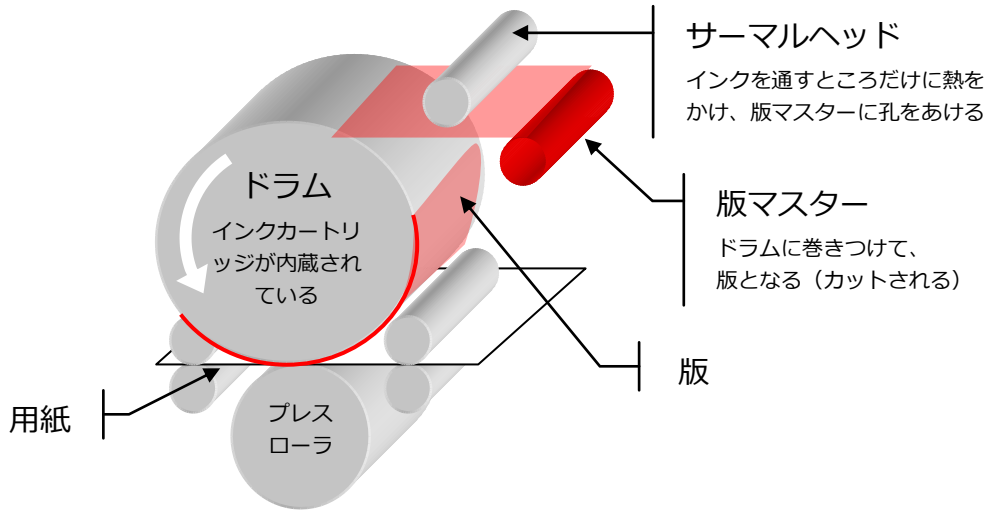
孔版印刷機の要求仕様は、理想科学様よりご提供いただきましたが、本カタログ内のモデルは、理想科学様内で使われている製品版のモデルとは一切関係ございません。また、理想科学様メンバは、本孔版印刷機のモデリング作業には、一切関わっておりません。

要求仕様

孔版印刷機とは、「孔版印刷」という技術を使った印刷機で、低コストに大量の印刷物を作成することを実現する機械です。

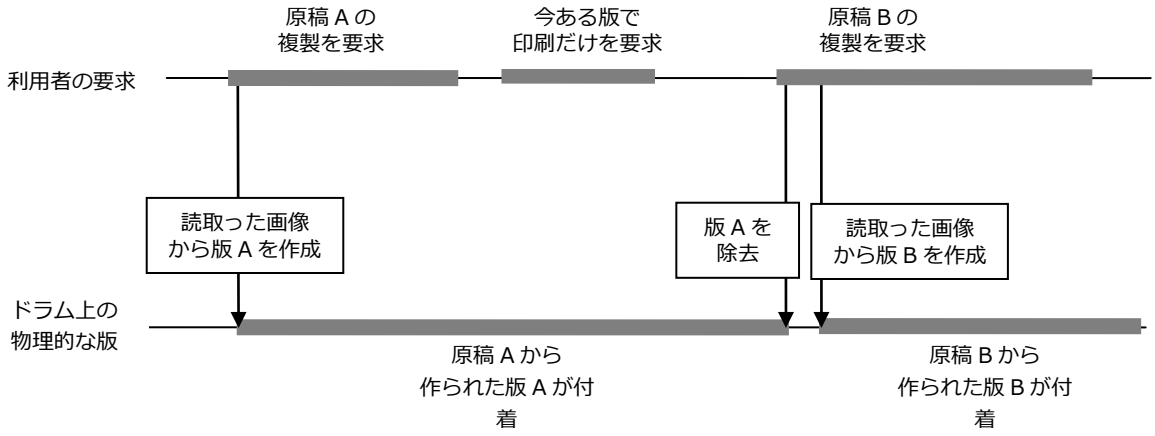
孔版印刷機での基本的な印刷方法

孔版印刷とは、『版』と言われるものに孔（あな）をあけ、孔があいた所にインクを通過させて、画像を用紙に転写する方法です。一度『版』を作ってしまえば、『版』が巻き付いたドラムに用紙を押し当てて通過させることで転写が完了することから、レーザープリンタ・インクジェットプリンタ・PPC 方式の複写機などの比較にならないほどのスピードで印刷することができます。



利用者の要求と版のライフサイクル

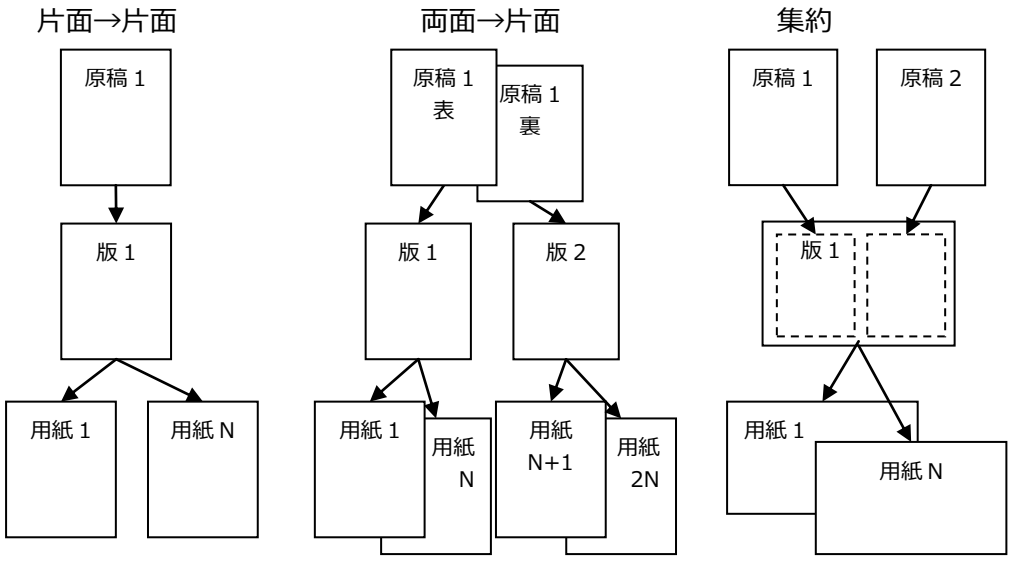
孔版印刷機では、利用者が要求する複写や印刷の単位と、その際に作成した版のライフサイクルは一致しません。例えば、一度複写するために版を作成すると、そのまま版を残しておき、別のタイミングでその版を使って印刷を行うことができます。版を除去するタイミングは、次に原稿から複製を作るとき、もしくは、利用者が版の除去を指定した時となります。



さまざまな印刷設定

孔版印刷機の印刷機能としては、片面・両面・集約・分割・色分割（エリア指定・自動認識）などがあります。それらの機能は、ドラムが 1 つで実現できるものもあれば、ドラムを複数必要とする機能もあります。

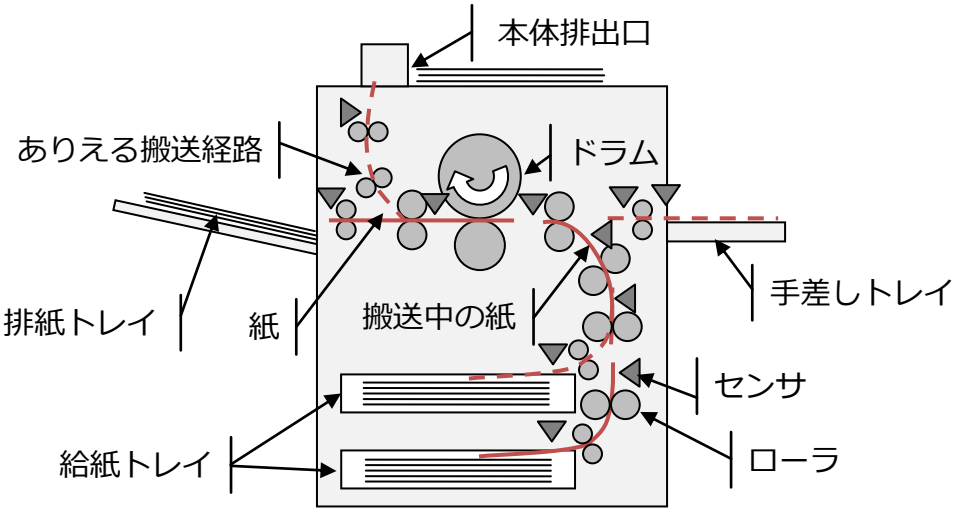
本カタログモデルでは、ドラムが 1 つで実現できる機能にフォーカスしてモデルを作成することにします。



紙搬送

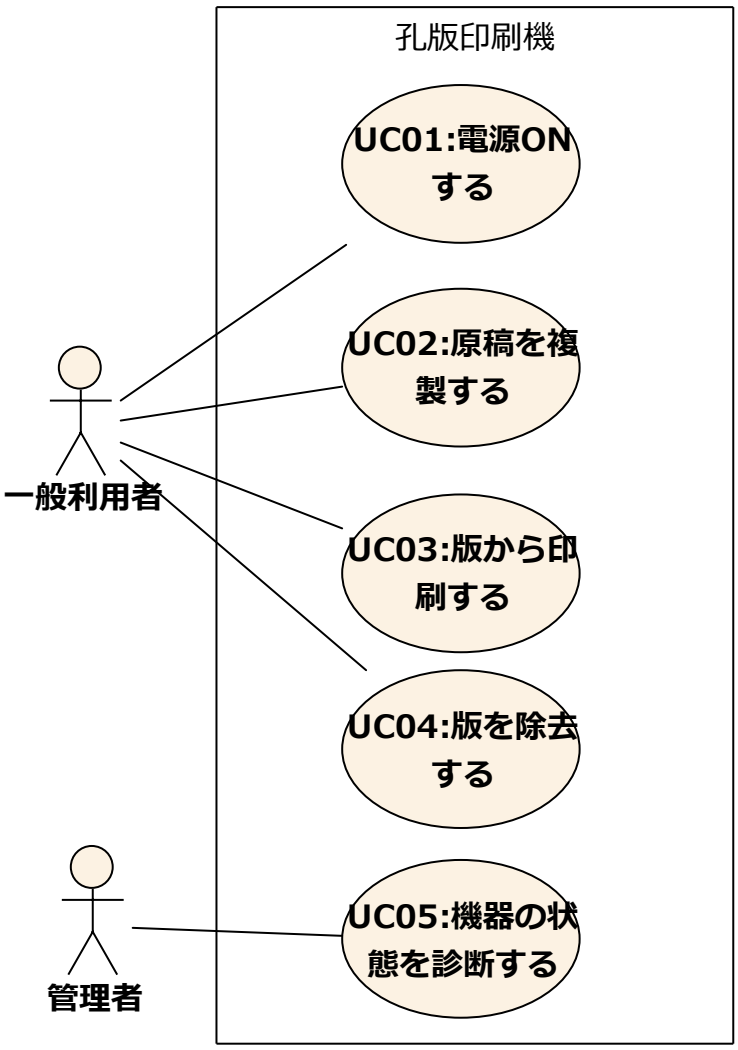
孔版印刷機における紙搬送は、通常のプリンタと殆ど同じです。

ここでは、印刷する用紙を給紙トレイから給紙し、排紙するまでについて説明します。用紙は、ある決められた印刷経路を通って行きます。印刷経路は、給紙する位置をスタート、排紙する位置をゴールとし、その間に印刷する位置を通過します。用紙が通る印刷経路は、その機器のメカ的なレイアウトにより、給紙できる場所、排紙できる場所、どのような道筋を通るかは決まっています。一般的に、一つの機器で複数の経路の選択肢がありえます。しかし、利用者がスタートキーを押した時点で、どこのスタート位置から始まり、どのような道筋を通り、ゴール位置にたどり着くかは決まります。



ユースケース図

本カタログモデルが実現する範囲のユースケースを示します。



ユースケース記述

<UC01：電源 ON する>

■概要

装置の電源を投入し、システムを起動させる

■アクター

一般利用者、管理者

■事前条件

・システムが電源 OFF である

■事後条件

・メイン画面が表示されている
・もし、故障があったならば、故障の表示がされている

■メインフロー

1. アクターは、システムの電源を投入する
2. システムは、アクターに対し、初期化処理中である旨を通知する
3. システムは、アクターに対し、メイン画面を表示する
4. アクターは、メイン画面を確認することで、システムが起動したことを認識する
5.UC を終了する

■代替フロー

3a. システムが初期化中に、故障を発見した場合
3a1. システムは、アクターに対し、メイン画面と故障通知を表示する
3a2. メインフローの 4 に戻る

■課題やＴ.B.D 項目

なし

■備考

なし

<UC02：原稿を複製する>

■概要

システムの読取装置に原稿を投入して、原稿を読み取って印刷物を出力する

■アクター

一般利用者、管理者

■事前条件

・メイン画面が表示されている

■事後条件

・メイン画面が表示されている

・版が作成されている

・印刷物が出力されている

■メインフロー

1. アクターは、システムの読取装置に、原稿を投入する
2. アクターは、システムに対し、枚数を指定して、原稿の印刷を要求する
3. システムは、アクターに対し、原稿の原稿を読み取り中である旨を通知する
4. システムは、アクターに対し、版を作成中である旨を通知する
5. システムは、アクターに対し、印刷中である旨を通知する
6. システムは、印刷された用紙を出力する
7. システムは、アクターに対し、メイン画面を表示する
8. アクターは、システムの読取装置から、原稿を取り除く
9. UC を終了する

■課題やＴ.B.D 項目

なし

■備考

なし

＜UC03：版から印刷する＞

■概要

すでに作成されている版から印刷する

■アクター

利用者

■事前条件

・メイン画面が表示されている

・版が作成済みである

■事後条件

・印刷された用紙が出力されている

・メイン画面が表示されている

■メインフロー

1. アクターは、システムに対し、枚数を指定して、版からの印刷を要求する
2. システムは、アクターに対し、印刷中である旨を通知する
3. システムは、印刷された用紙を出力する
4. システムは、アクターに対し、メイン画面を表示する
5. UC を終了する

■課題やＴ.B.D 項目

なし

■備考

なし

＜UC04：版を除去する＞

■概要

既に作成されている版をドラムから取り除き、排出ボックスに排出する

■アクター

利用者

■事前条件

・版が作られている

・メイン画面が表示されている

■事後条件

・版が排出ボックスに排出されており、ドラムには版がない

・メイン画面が表示されている

■メインフロー

1. アクターは、システムに対し、版の除去を要求する
2. システムは、排出ボックスが、閉じられている事を確認する
3. システムは、アクターに対し、版を除去中である旨を通知する
4. システムは、アクターに対し、メイン画面を表示する
5. UC を終了する

■課題やＴ.B.D 項目

なし

■備考

なし

＜UC05：機器の状態を診断する＞

■概要

孔版印刷機の自己診断機能で、故障がないかを確認する

■アクター

管理者

■事前条件

メイン画面が表示されている

■事後条件

メイン画面が表示されている

■メインフロー

1. アクターは、システムに対し、管理画面への移行を要求する
2. アクターは、システムに対し、故障診断を要求する
3. システムは、各ユニットの故障診断を行い、アクターに対して、診断結果を通知する
4. アクターは、システムに対し、診断結果表示のクリアを要求する
5. システムは、アクターに対し、管理画面を表示する
6. アクターは、システムに対し、メイン画面への移行を表示する
6. UC を終了する

■課題やＴ.B.D 項目

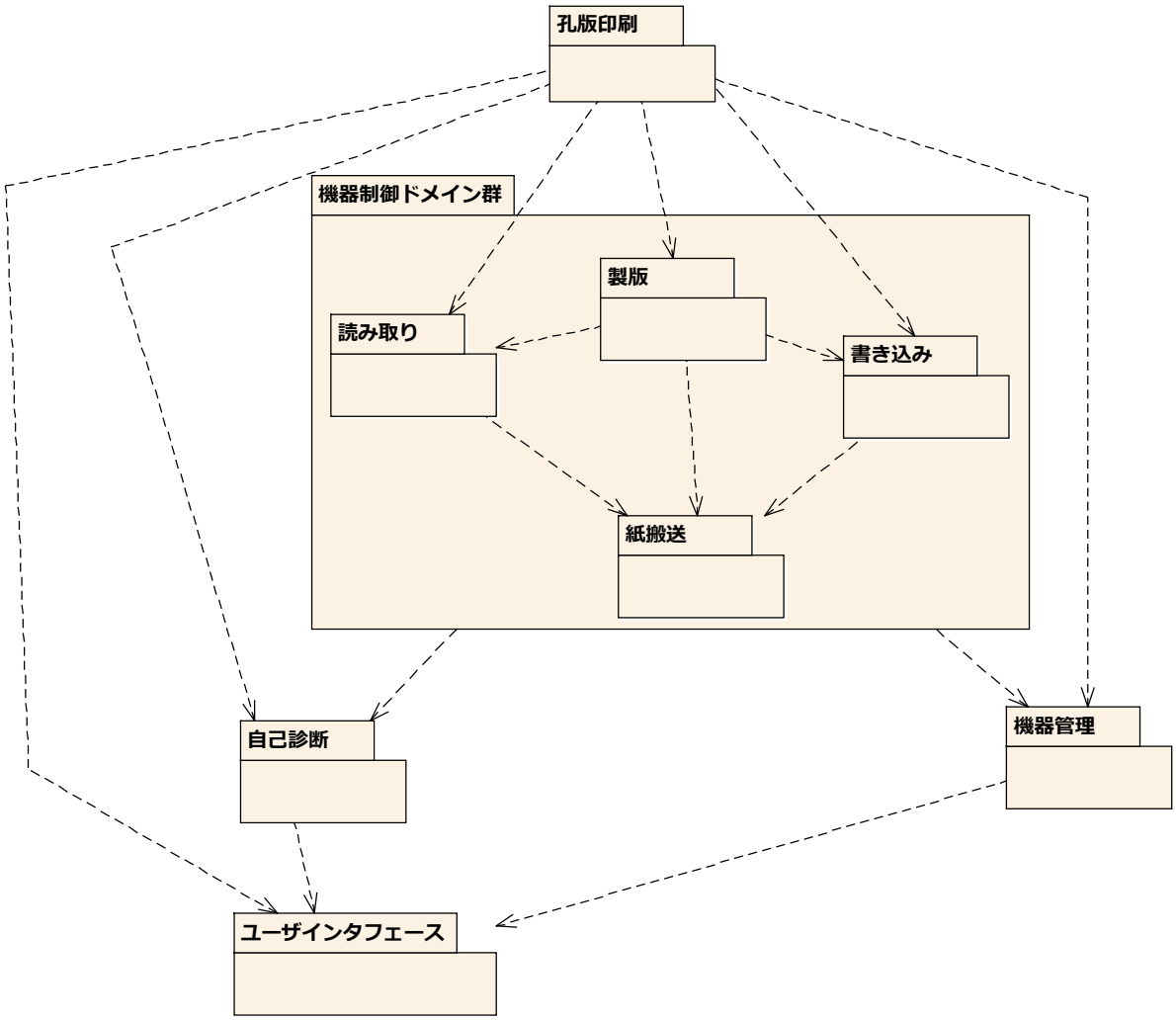
なし

■備考

どのユニットの故障診断を行うのか？

ドメインチャート

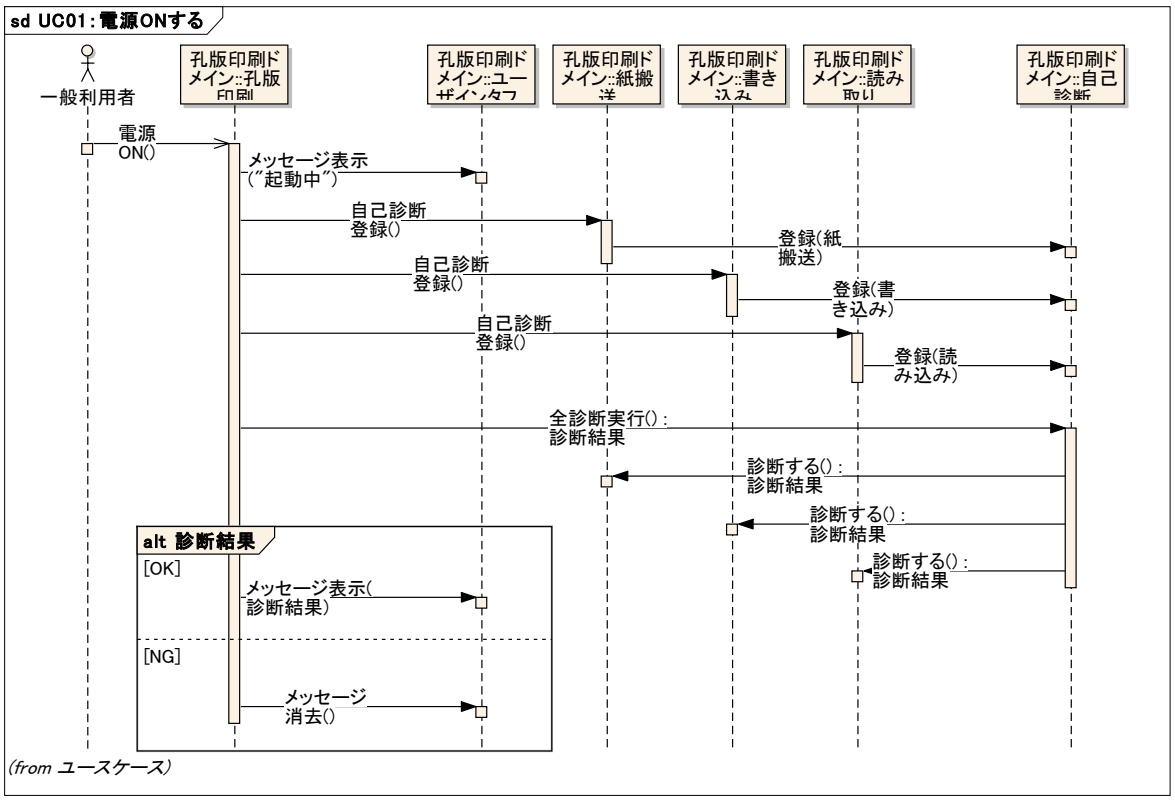
孔版印刷機のシステム全体を、次のようにドメイン分割しました。以降に記載しているモデルは、このドメイン分割を前提としています。



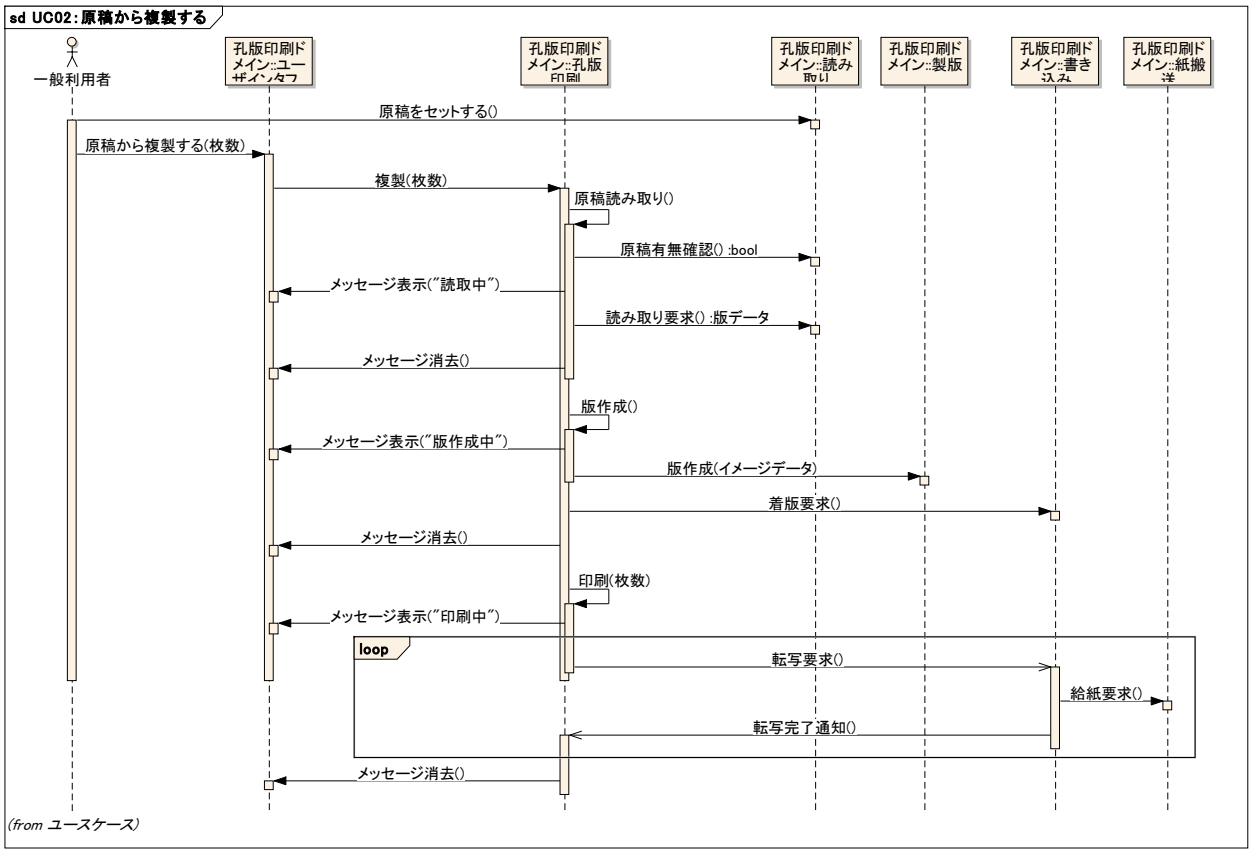
- ◆ 孔版印刷　　：「原稿の複製・印刷物」を作成するという孔版印刷機の目的を実現する
- ◆ 機器制御ドメイン群：機器制御をおこなうドメインのグループ
- ◆ 読み取り　　：原稿の読み取りを、装置についているメカ機構を利用して実現する
- ◆ 製版　　：マスターから、サーマルヘッドを使って孔をあけ、版を作成してドラムに巻きつける
- ◆ 書き込み　　：製版と用紙への転写を、装置についているメカ機構を利用して実現する
- ◆ 紙搬送　　：原稿および印刷対象の用紙の搬送を、装置についているメカ機構を利用して実現する
- ◆ 機器管理　　：機器全体の稼働状況を把握する
- ◆ ユーザインタフェース：利用者とのコミュニケーションを実現する
- ◆ 自己診断　　：機器の機能・装置・部品などの状態を診断して、診断結果を報告する

ドメイン間シーケンス図

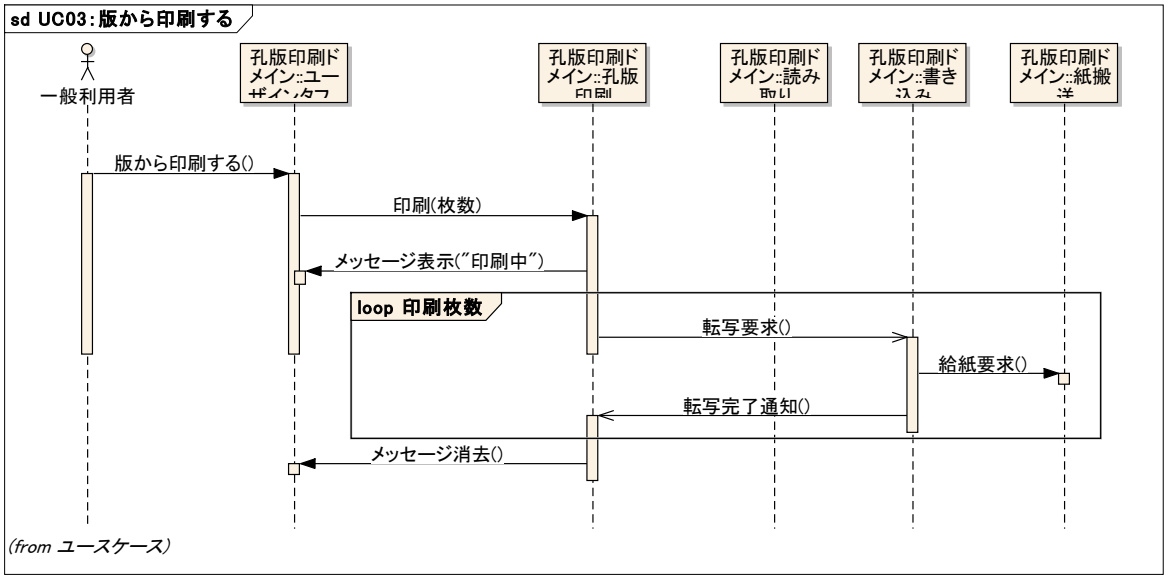
電源 ON のシーケンス



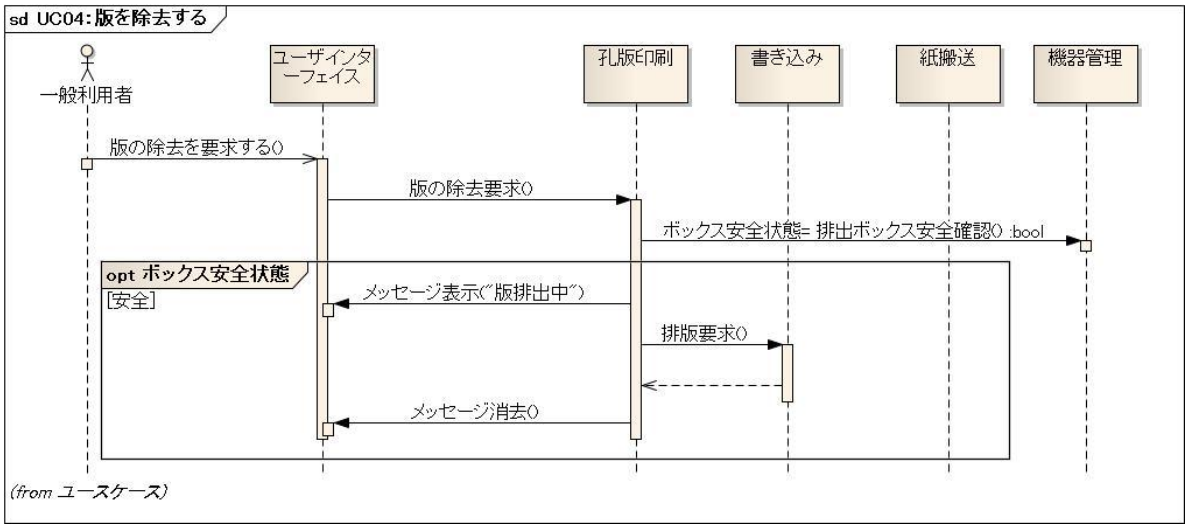
原稿から複製するシーケンス



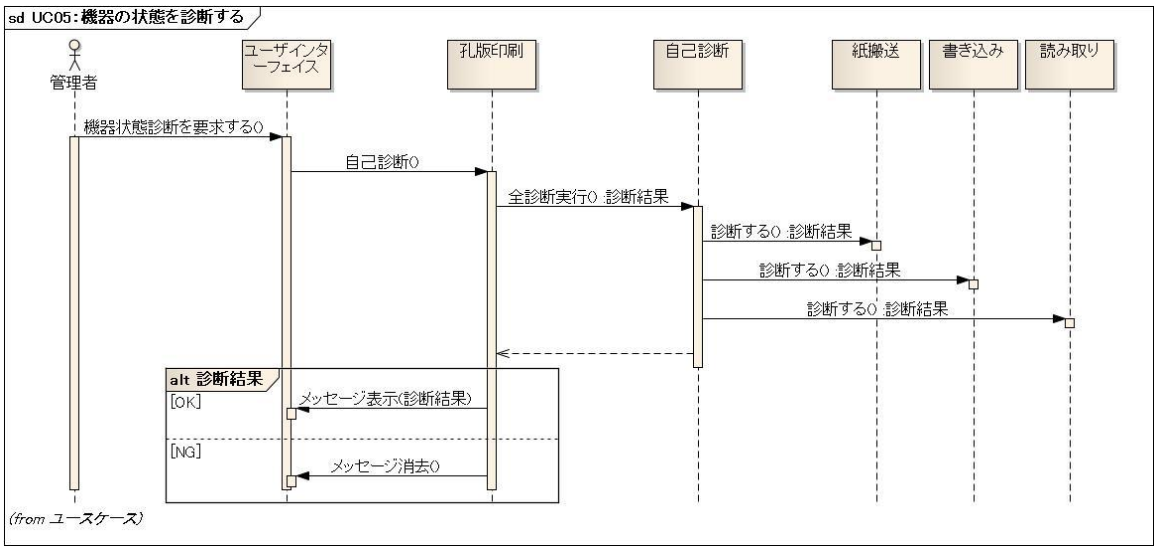
版から印刷するシーケンス



版を除去するシーケンス



機器の状態を診断するシーケンス



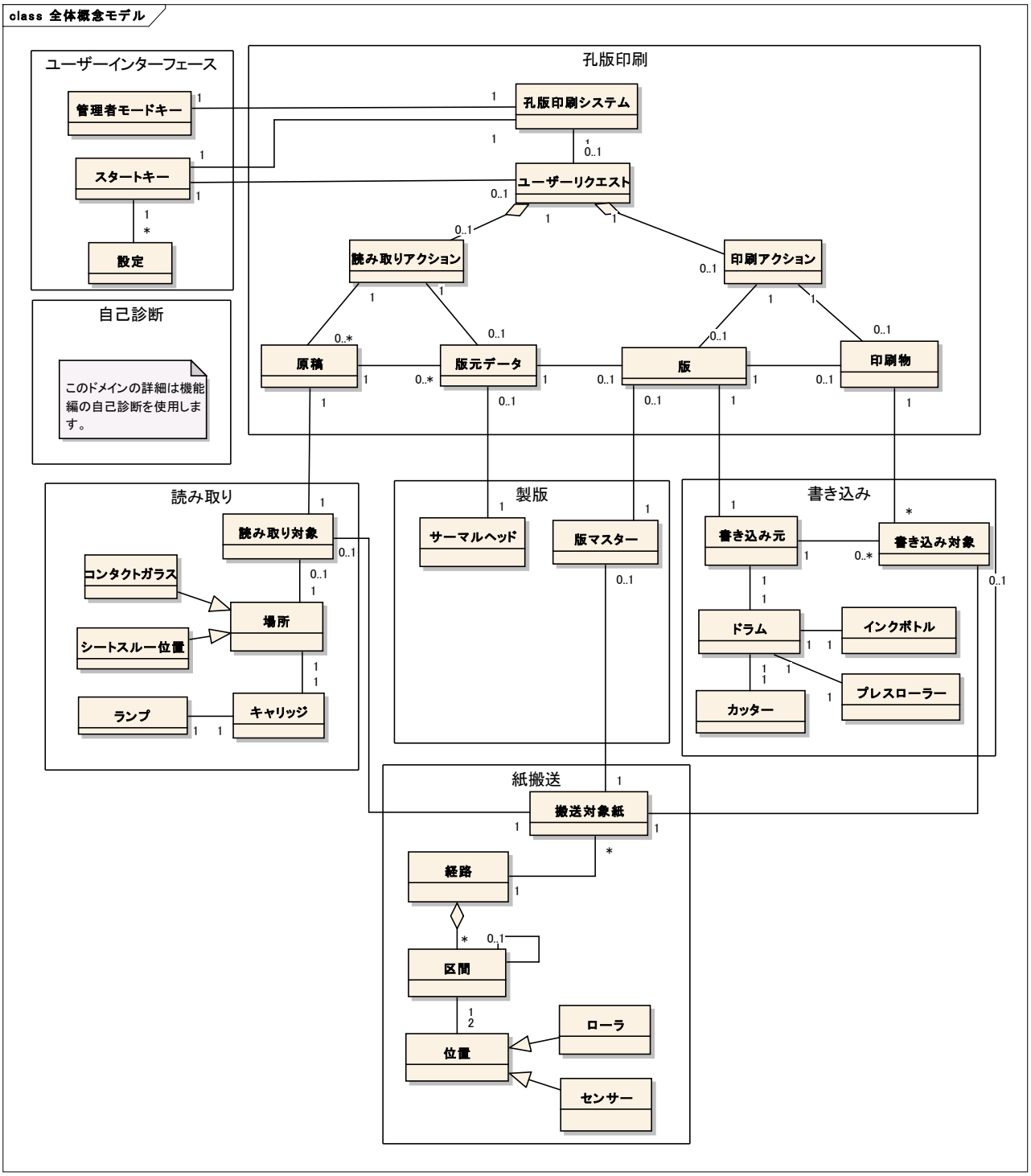
モデル一覧

モデル名	概要	ポイント
エンティティに着目して分析したモデル	孔版印刷機の特徴は、原稿から用紙に複写されるまでの特徴として、「版」が存在することです。それらの対応関係の構造を中心に、エンティティに着目して分析したモデルです。	いろいろな印刷設定が増えた場合に対応しやすいモデルです。

エンティティに着目して分析したモデル

要求仕様で整理された原稿、版、用紙の静的な構造に着目し、これらを孔版印刷機の中心と位置づけます。その構造を中心として孔版印刷機をモデル化した全体概念モデルと、それをドメインに分類したクラス図を次に示します。孔版印刷ドメインは、主にシステム全体の状態や、利用者からの要求を取り扱うことが使命です。それ以外のドメインは、各種デバイスを制御し、孔版印刷を実現するための手段を提供する、という役割分担となっています。

概念モデル



分析モデル

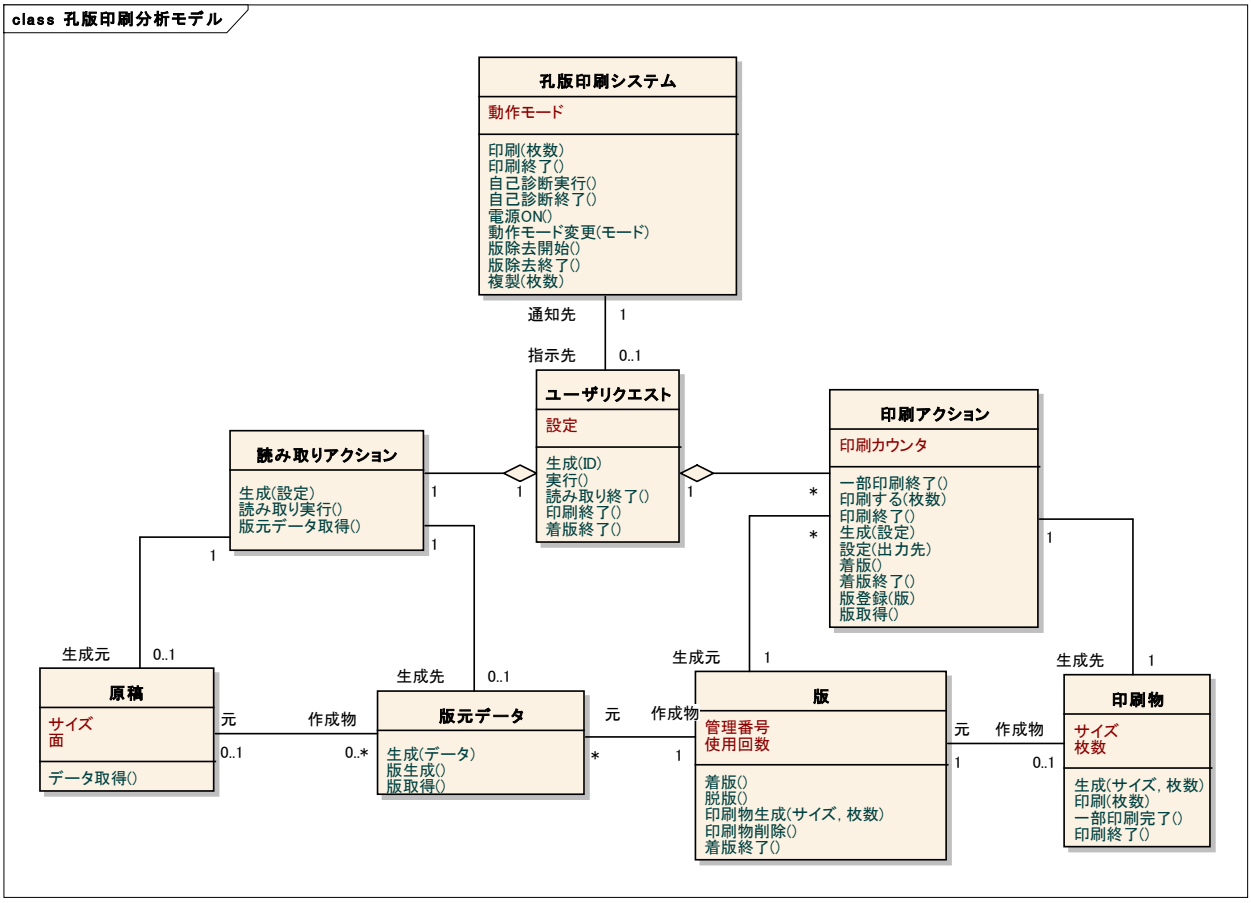
孔版印刷ドメイン

モデリングのコンセプト

孔版印刷ドメインは、システム全体の最上位のドメインであり、利用者からの要求を実現することが使命です。
利用者からの要求の特徴は、要求仕様でまとめたように、原稿から複製を作成する場合と、すでにある版を使って印刷する場合の 2 種類があります。原稿から複製を作成する場合には、原稿を読み取って画像化し、その画像を使って版を作成し、版から印刷物を作成する、といった流れになります。すでにある版を使って印刷する場合には、版から印刷物を作成するだけとなります。

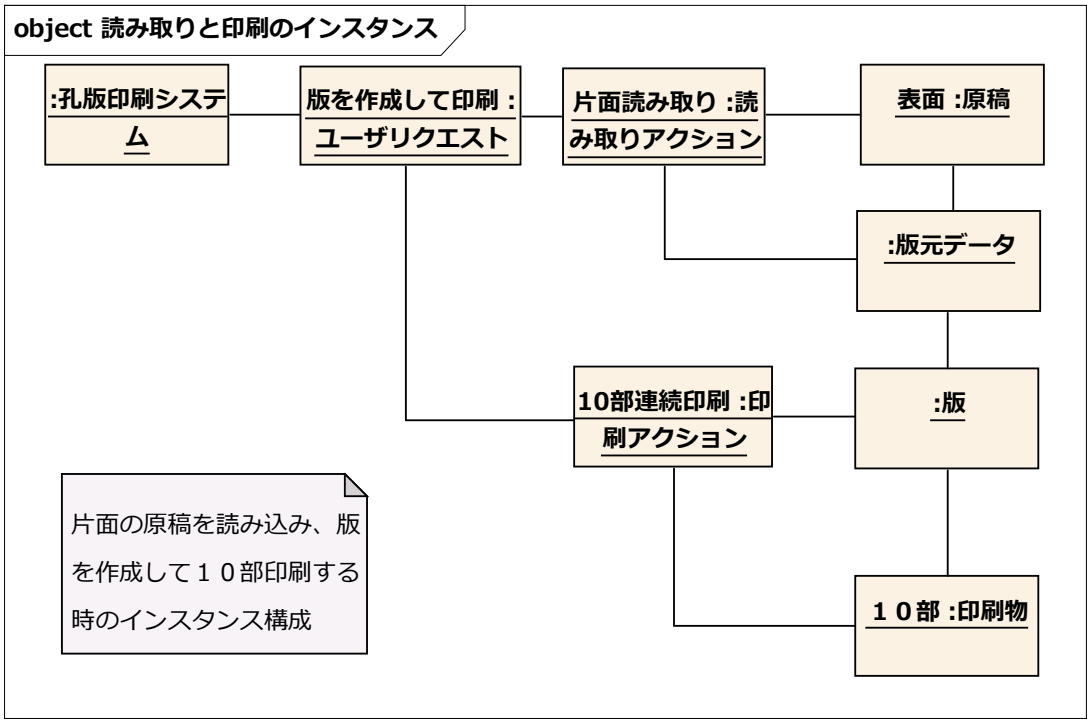
また、このドメインでは、パフォーマンスを出すために、一枚ずつ印刷結果を確認しません。印刷自体のながれは、書き込みドメインに指定した枚数分の印刷の管理をゆだねます。

クラス図

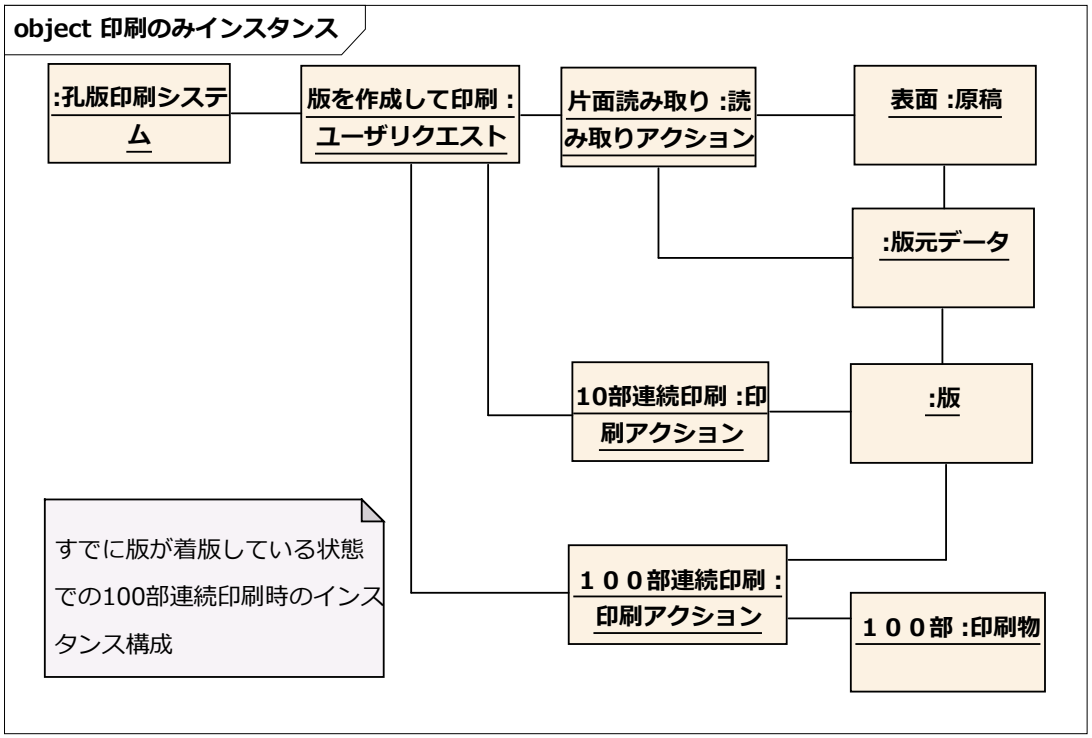


オブジェクト図

原稿から複製を作成する際のオブジェクト構成

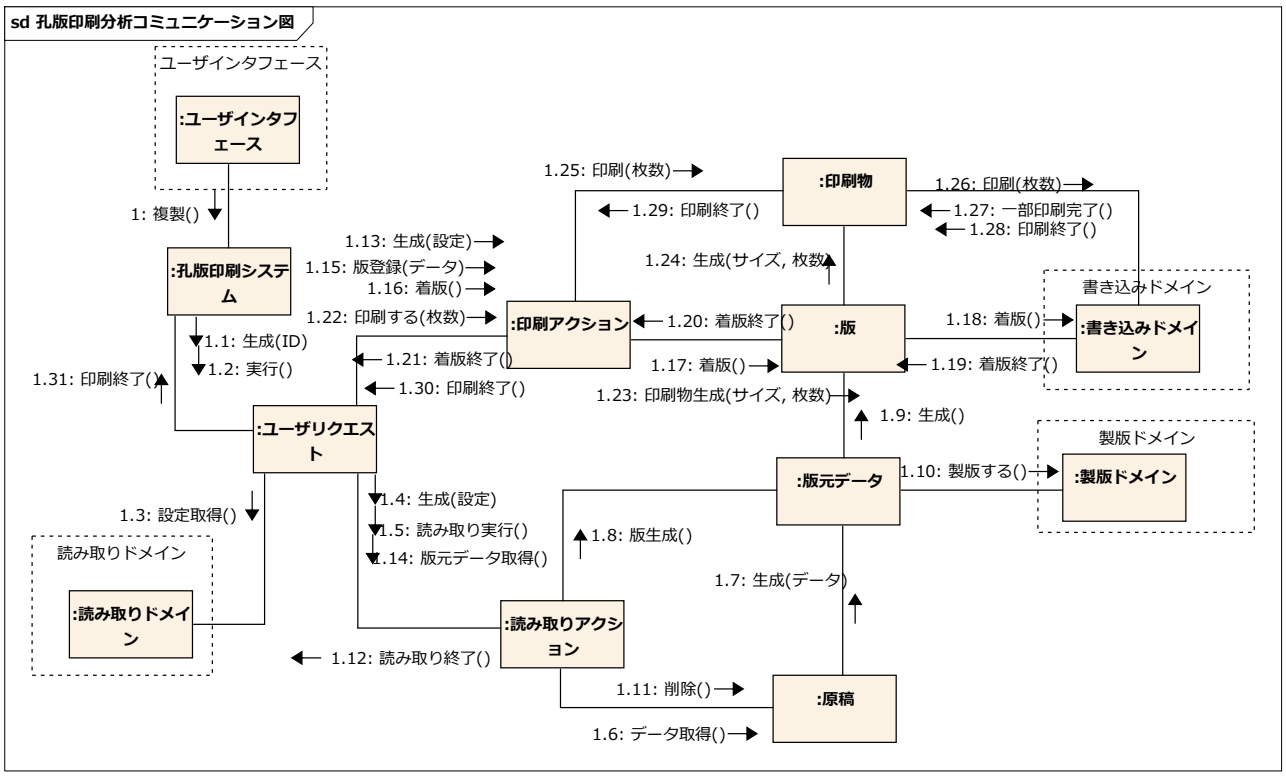


版から印刷する際のオブジェクト構成



コミュニケーション図

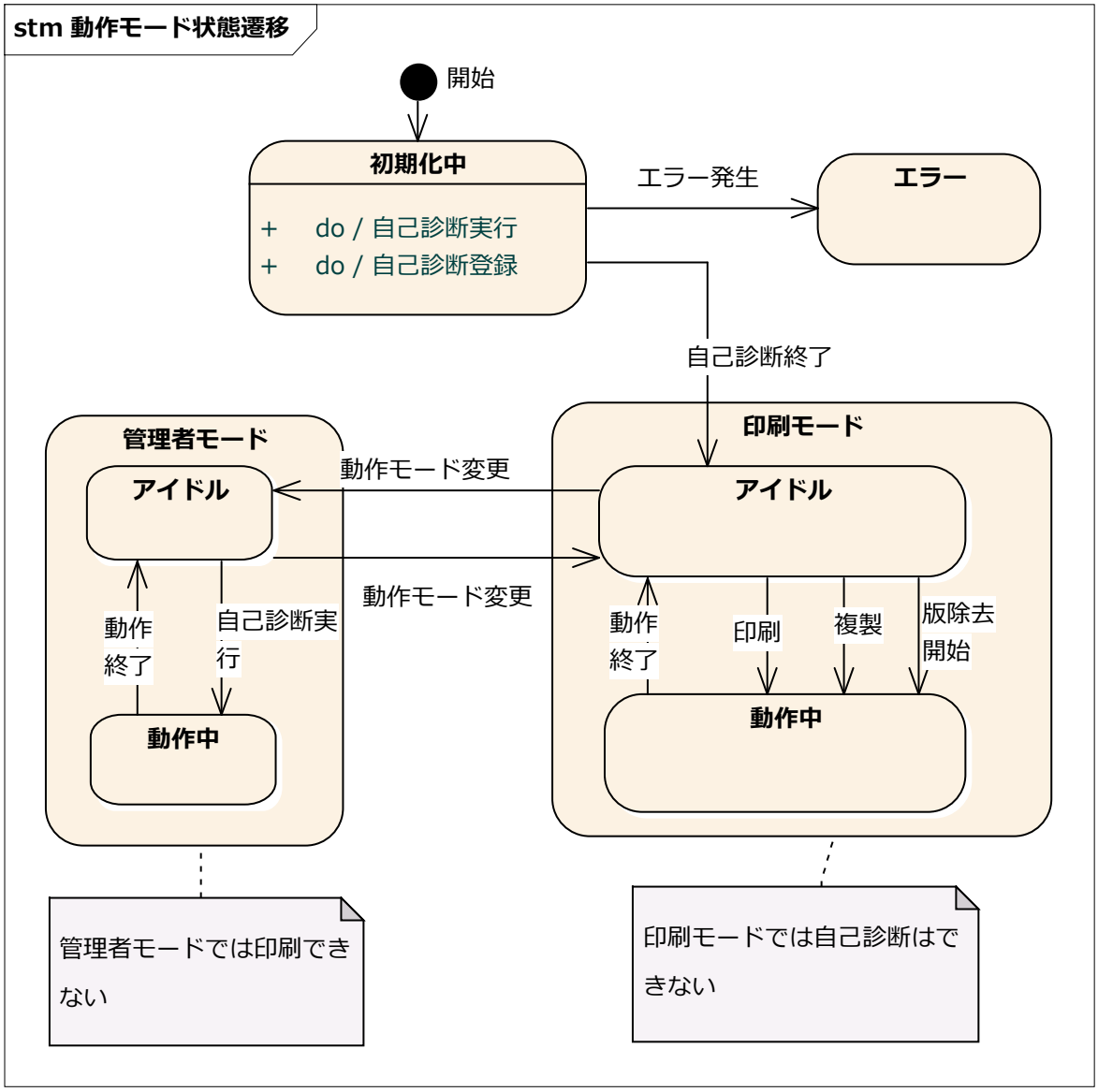
印刷開始指示により、原稿読み取り、製版、印刷を行うコミュニケーション図



ステートマシン図

「孔版印刷システム」クラスのステートマシン図

孔版印刷システムクラスは、機器全体の動作モードレベルの振舞いを責任範囲とします。初期化中に起こったエラーは、自己診断で発見したエラーであり、サービスマン等を呼ばないと解決しないような問題となります（ジャムなどの問題ではない）。



紙搬送ドメイン

紙搬送ドメインは、印刷用の紙搬送だけでなく、原稿やサーマルペーパーの紙搬送も役割として担っています。

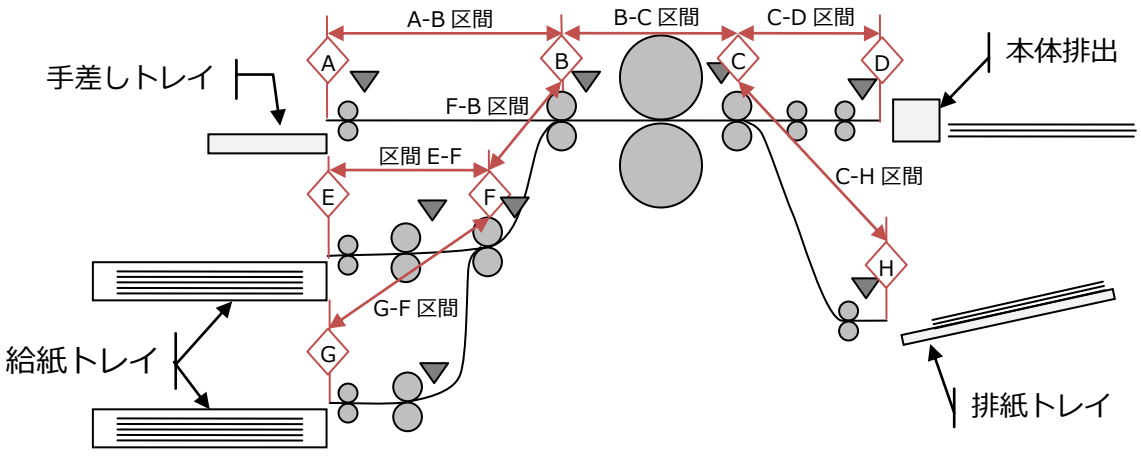
モデリングのコンセプト

原稿・サーマルペーパー・印刷用紙の全ての紙搬送において、対象の紙が搬送を開始する場所から、搬送を終了する場所まで、ある経路を通って搬送されていきます。また、それらが動的に変化しない、という点で共通の特性を持っています。

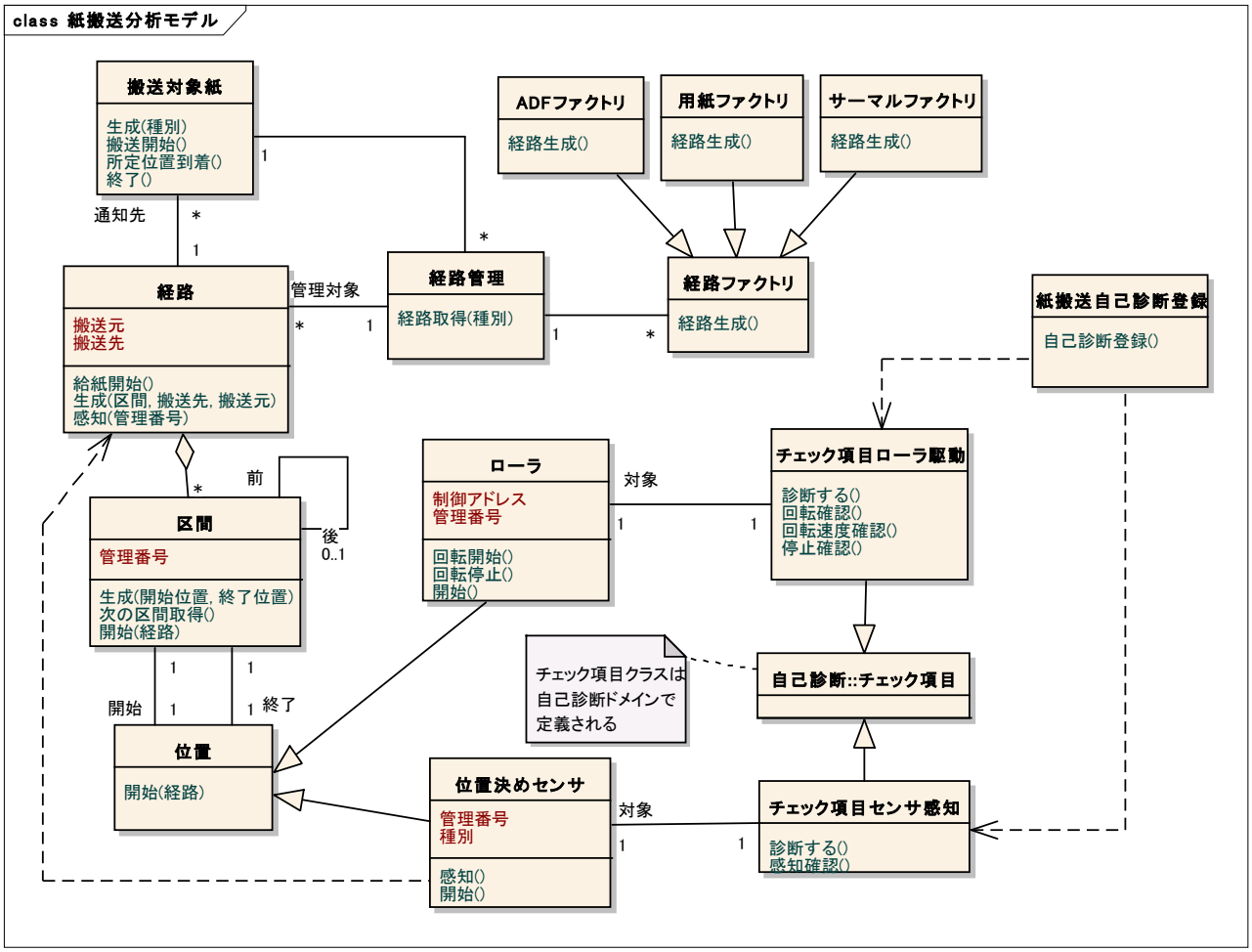
どの搬送においても、搬送経路を意味のある単位に区切り、それらをつなげることで経路を構成することができます。区間には、必ず開始位置と終了位置があり、その位置はローラや位置決めセンサがレイアウトされています。

次の図は、印刷用紙搬送を例に説明したものです。

本モデルでは、紙を搬送するための仕組みを、紙は決められた経路を搬送されるもの、経路は複数の区間で構成されるもの、区間には開始位置と終了位置があるものとしてモデリングします。

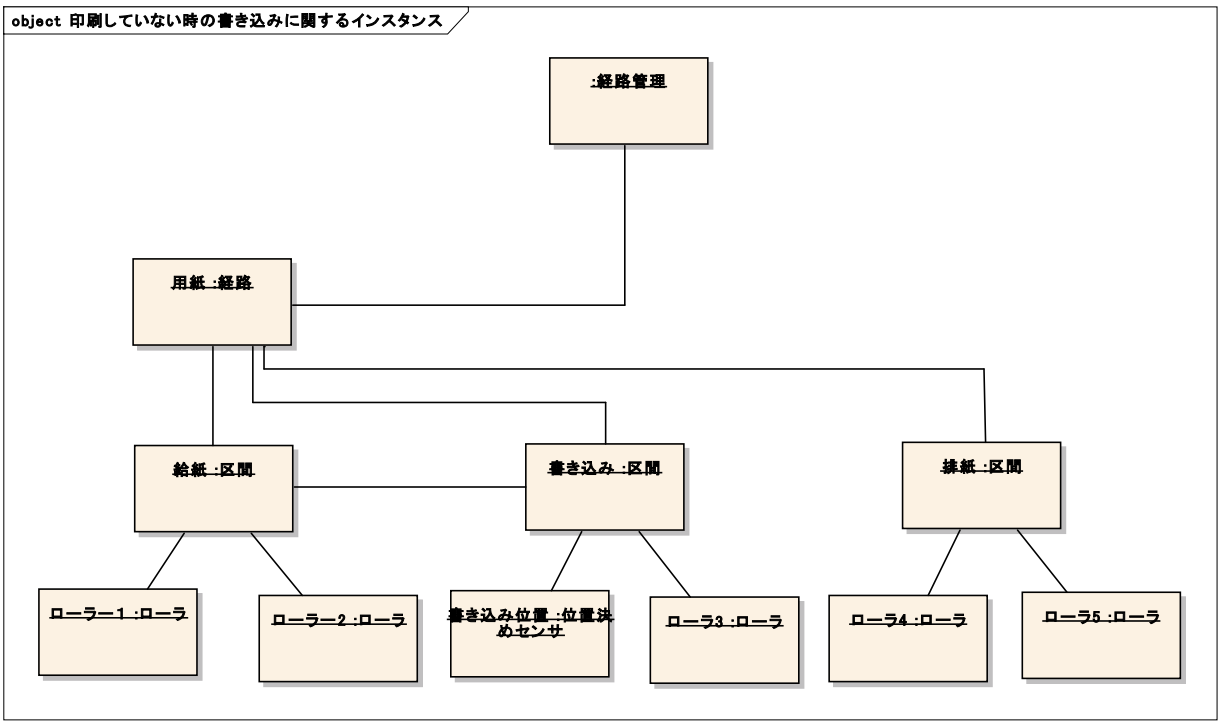


クラス図

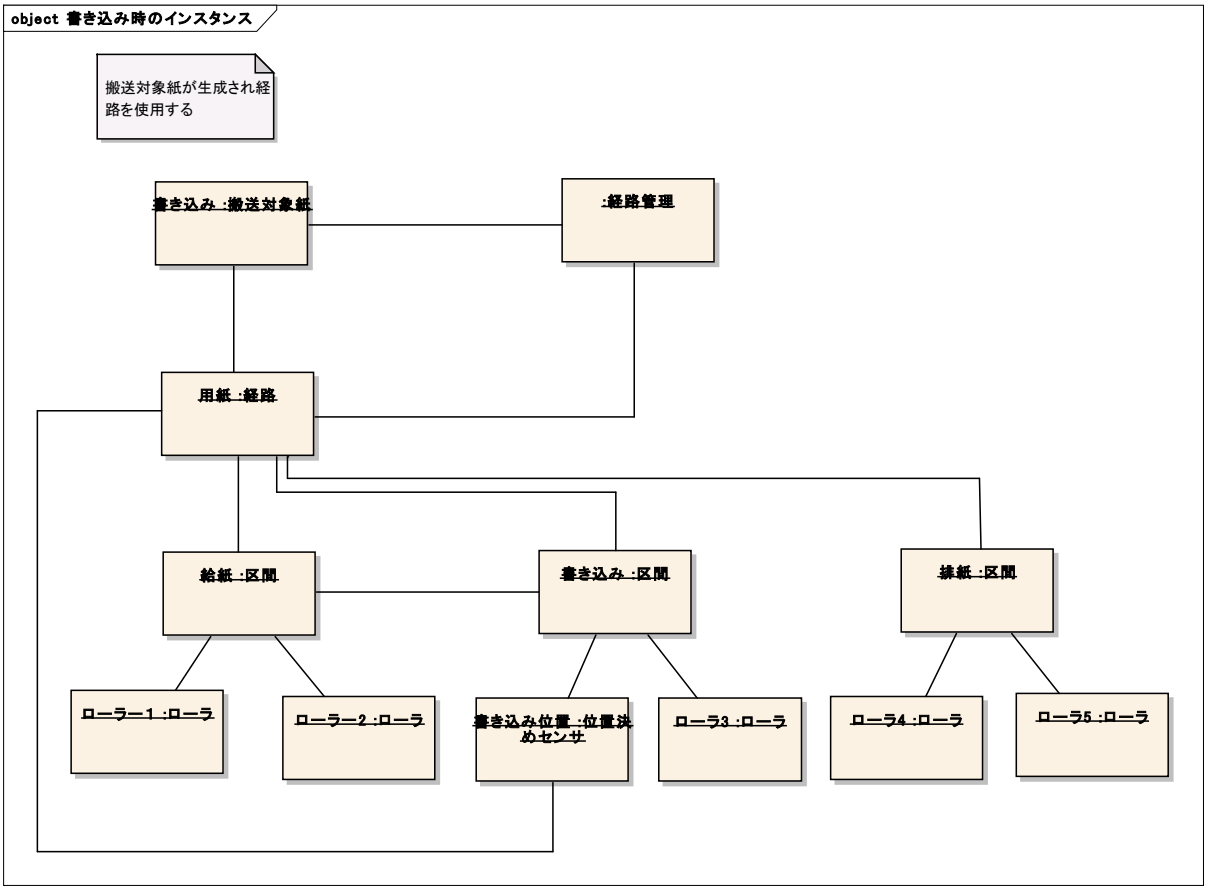

class 紙搬送分析モデル
搬送対象紙
生成(種別)
搬送開始()
所定位置到着()
終了()
ADFファクトリ
経路生成()
用紙ファクトリ
経路生成()
サーマルファクトリ
経路生成()
通知先
経路
搬送元
搬送先
給紙開始()
生成(区間, 搬送先, 搬送元)
感知(管理番号)
管理対象
経路管理
経路取得(種別)
経路ファクトリ
経路生成()
紙搬送自己診断登録
自己診断登録()
区間
管理番号
生成(開始位置, 終了位置)
次の区間取得()
開始(経路)
前
後
0..1
ローラ
制御アドレス
管理番号
回転開始()
回転停止()
開始()
対象
チェック項目ローラ駆動
診断する()
回転確認()
回転速度確認()
停止確認()
開始
終了
位置
開始(経路)
チェック項目クラスは
自己診断ドメインで
定義される
自己診断::チェック項目
位置決めセンサ
管理番号
種別
感知()
開始()
対象
チェック項目センサ感知
診断する()
感知確認()

オブジェクト図

印刷を行っていない時のオブジェクト構成

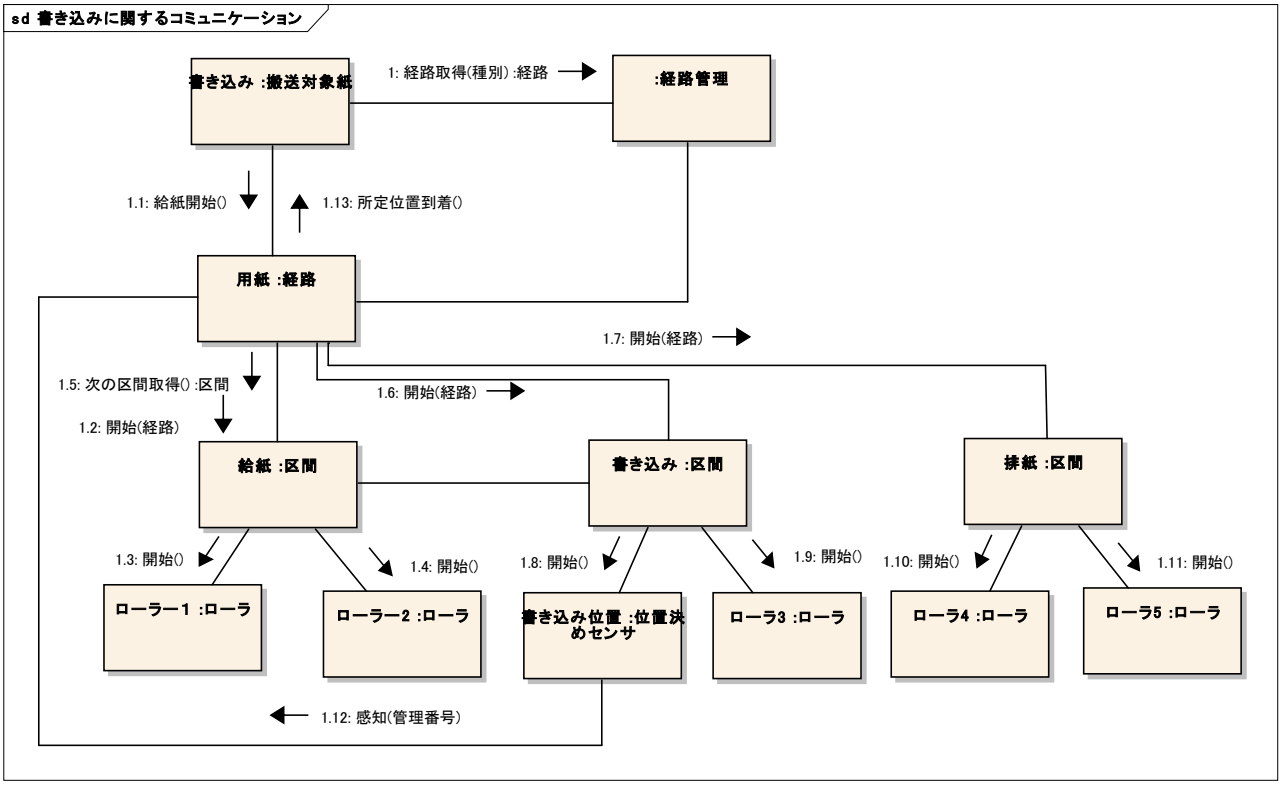


印刷を行っている最中のオブジェクト構成



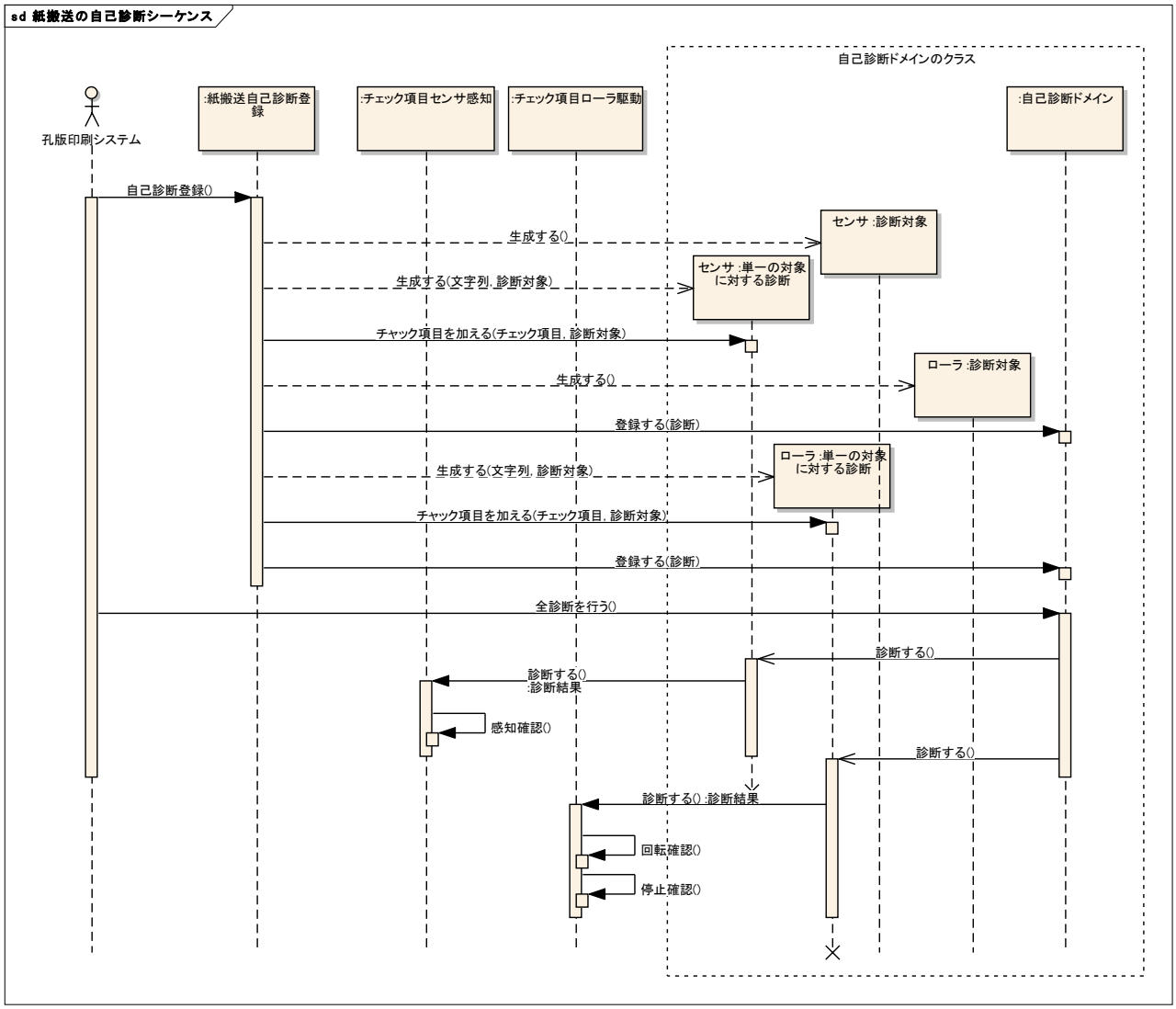
コミュニケーション図

通常の搬送のコラボレーション



シーケンス図

自己診断を登録し実行するシーケンス



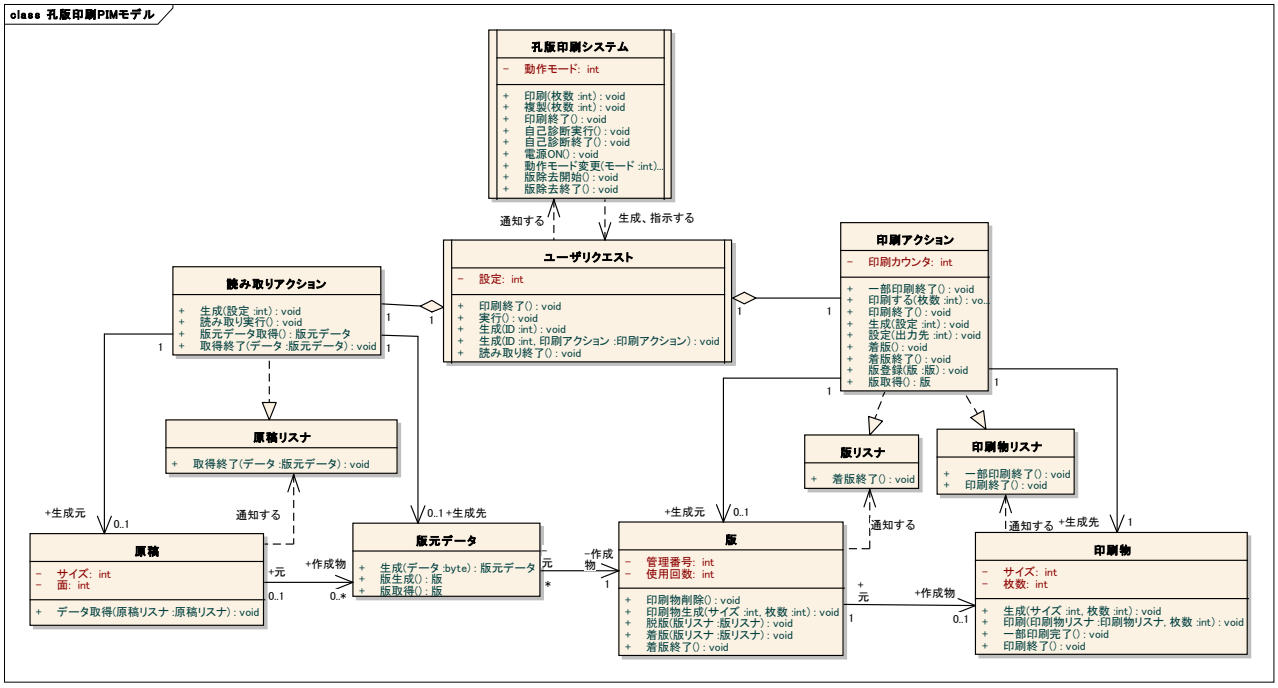
PIM 設計モデル

孔版印刷ドメイン

モデリングのコンセプト

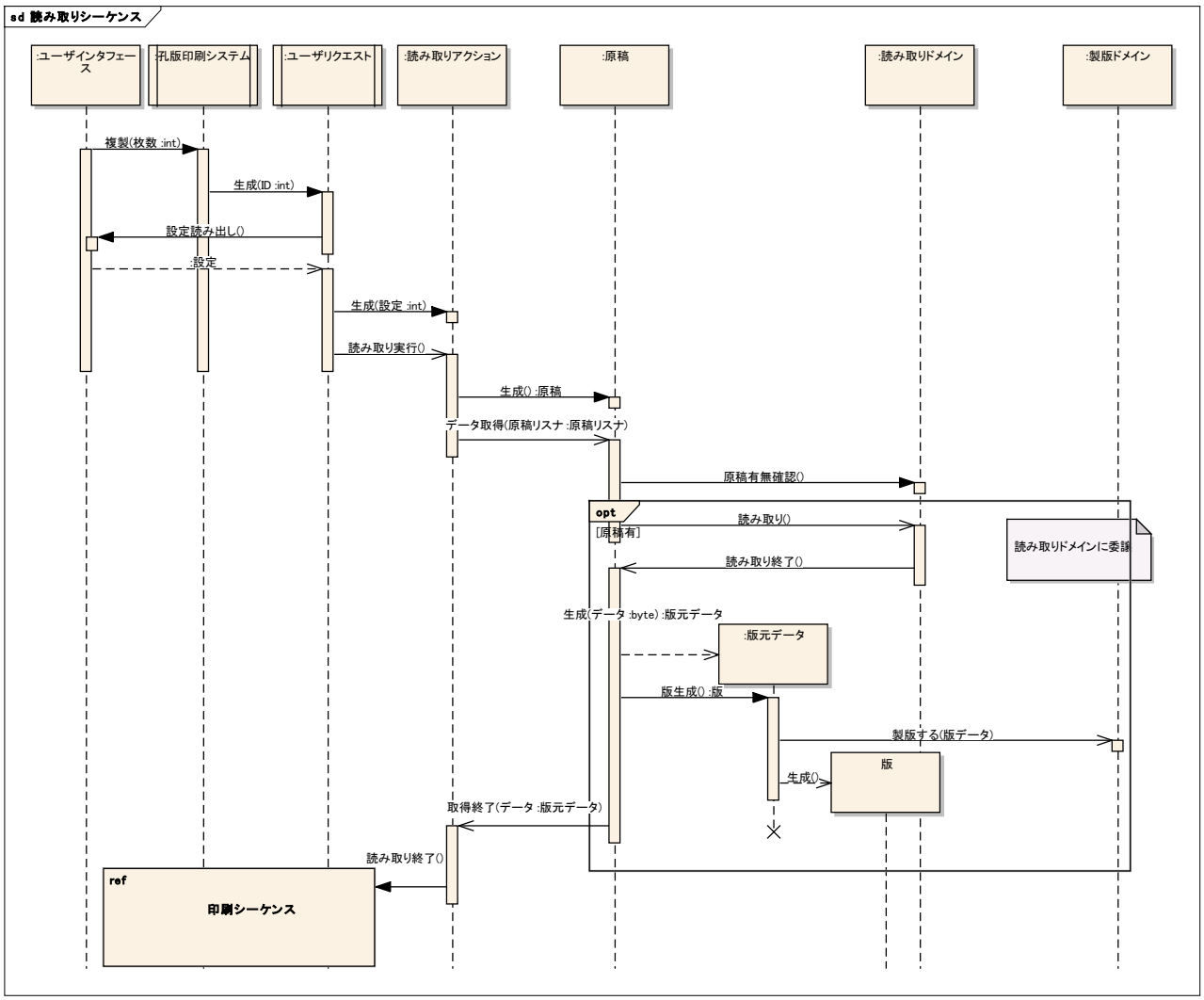
　アクティブオブジェクトは、並列での動作が必要となる「孔版印刷システム」と「ユーザーリクエスト」とします。また、循環参照を避けるために、メッセージ通信には「リスナ」を導入します。

クラス図

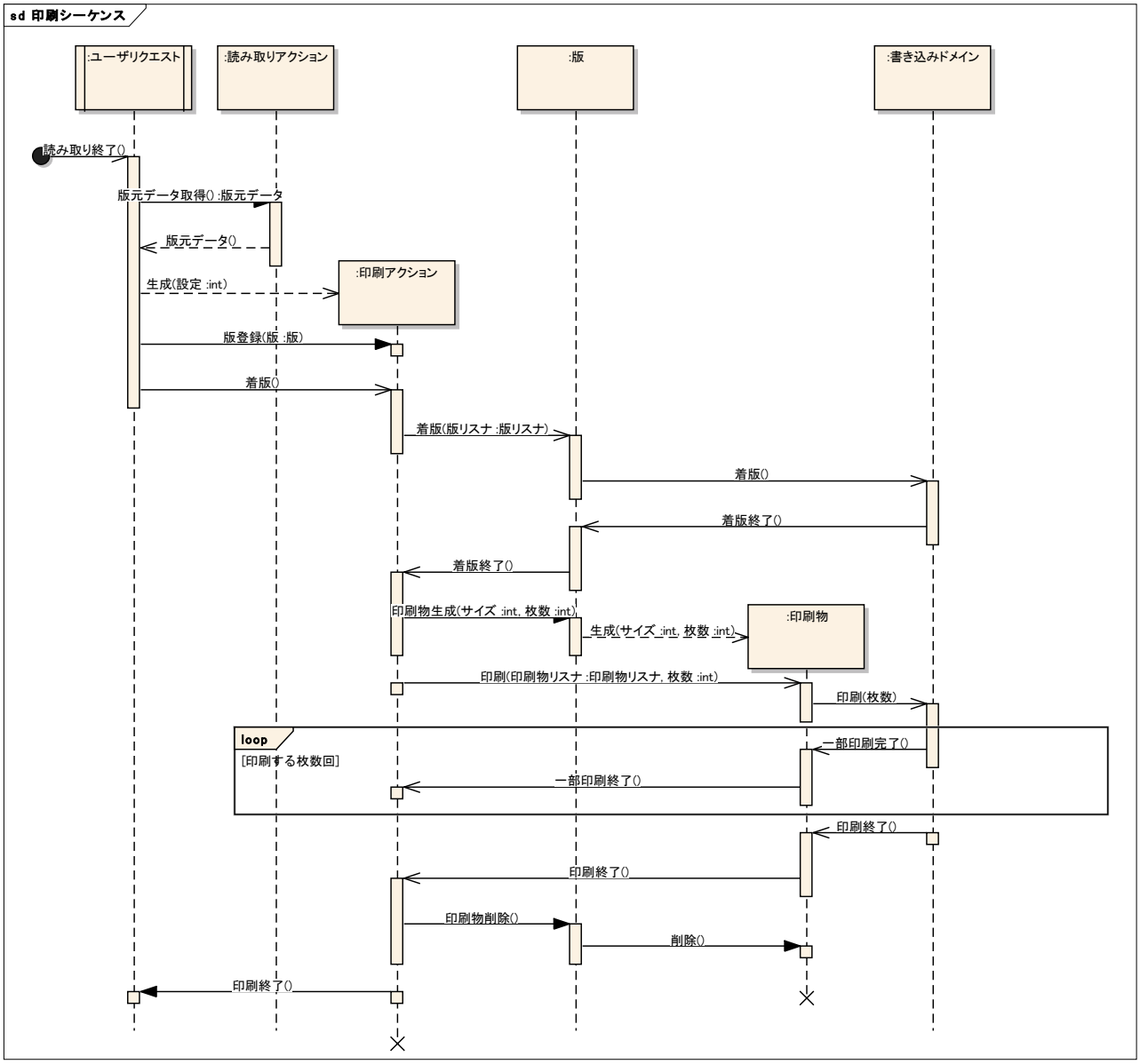


シーケンス図

読み取りのシーケンス図



着版と印刷時のシーケンス図



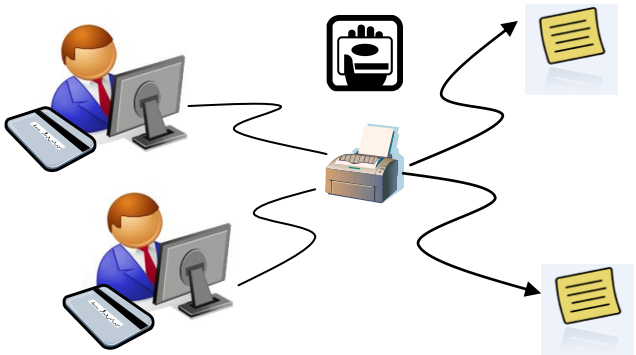
機能編

認証

要求仕様

認証例

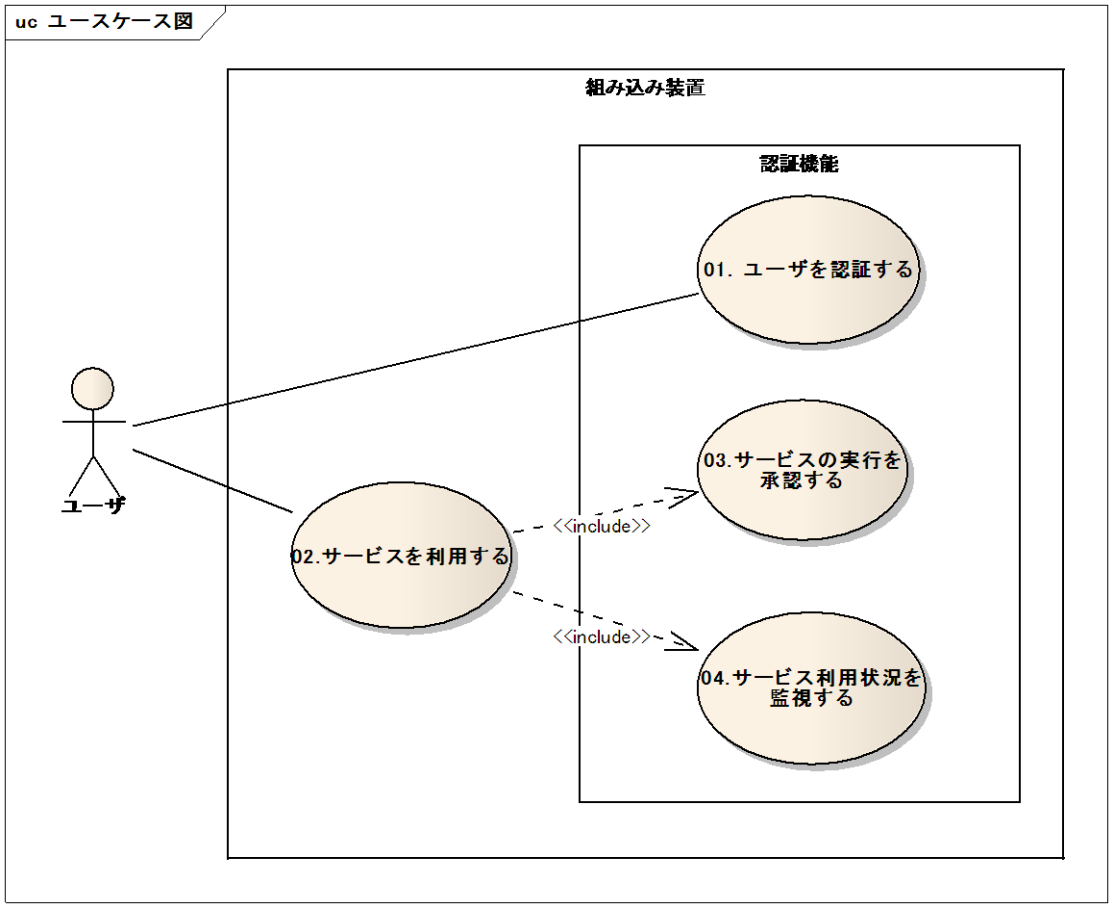
ユーザ A とユーザ B がいます。二人はそれぞれ固有の ID カードを持っています。この 2 人が 2 台の PC からそれぞれの電子データを 1 台のプリンタに印刷指示を出します。ユーザ A がプリンタに ID カードをかざすと、ユーザ A が印刷指示した電子データが呼び出され、プリントが始まります。ユーザ A の電子データの印刷処理終了後、ユーザ B が同様に ID カードをプリンタにかざすと、ユーザ B の電子データが呼び出されプリントが始まります。ユーザ A,B の使用日時や電子データのタイトル、プリント枚数など使用記録がプリンタ保存されます。



このようなユーザごとに適切なサービスを提供できる仕組みを【認証】と呼称します。

この【認証】では、ユーザを識別し、ユーザごとに適切なサービスを提供し、機器の使用記録を取ります。この一連の機能の設計モデルを提供します。

ユースケース図


uc ユースケース図
組み込み装置
認証機能
01. ユーザを認証する
03.サービスの実行を承認する
02.サービスを利用する
<<include>>
<<include>>
04.サービス利用状況を監視する
ユーザ

ユースケース記述

＜UC01 名：ユーザを認証する＞

■概要

ユーザが正当なユーザであるかどうかを判断する

■アクター

ユーザ

■事前条件

なし

■事後条件

ユーザに認証結果が渡されていること

■メインフロー

1 .アクターは、システムに対し、アクセス許可を要求する
2. システムは、アクターが正当なユーザかどうか判断する
3. システムは、アクターに対し、認証結果を渡す
4. UC を終了する

■課題や T．B．D 項目

なし

■備考

なし

＜UC02 名：サービスを利用する＞

■概要

ユーザが組み込み装置で提供されているサービスを利用する

■アクター

ユーザ

■事前条件

ユーザが認証済みである

■事後条件

ユーザに認証結果が渡されている

■メインフロー

1 .アクターは、システムに対し、サービスの利用を要求する
2. システムは、認証機能に対し、「サービスの実行を承認する」ユースケースを実行する
3. システムは、認証機能から承認結果(OK)を受け取る
4. システムは、サービスを実行する
5. システムは、認証機能に対し、「サービス利用状況を監視する」ユースケースを実行する
6. システムは、アクターに対し、サービスの利用を開始した旨を返す
7. UC を終了する

■例外フロー

1 .アクターは、システムに対し、サービスの利用を要求する
2. システムは、認証機能に対し、「サービスの実行を承認する」ユースケースを実行する
3. システムは、認証機能から承認結果(NG)を受け取る
4. システムは、アクターに対し、承認に失敗した旨を返す
5. UC を終了する

■課題や T．B．D 項目

なし

■備考

なし

＜UC03 名：サービスの実行を承認する＞

■概要

ユーザが指定のサービスを実行できる権限があるか確認する

■アクター

なし

■事前条件

ユーザが認証済みである

■事後条件

システムにサービス利用可否が渡されている

■メインフロー

1. システムは、認証機能に対し、指定ユーザが指定サービスを利用できるか確認する
2. 認証機能は、指定ユーザにサービス実行の権限があるかどうかを判断する
3. 認証機能は、システムに対し、サービス利用可否を渡す
4. UC を終了する

■課題や T．B．D 項目

なし

■備考

なし

＜UC04 名：サービス利用状況を監視する＞

■概要

ユーザがサービスを利用している状況を監視し、記録する

■アクター

なし

■事前条件

ユーザのサービス利用が承認されている

■事後条件

サービス利用状況の記録(課計)が存在している

■メインフロー

1.システムは、 認証機能に対し、サービス利用状況を通知する
2. 認証機能は、サービス利用状況を記録する
3. UC を終了する

■課題や T．B．D 項目

なし

■備考

なし

モデル一覧

モデル名	概要	ポイント
機能に着目して分析したモデル	認証機能の設計として、広く知られている AAA モデルの 3 機能に着目したモデルです。	AAA モデルの 3 要素を基本クラスとしたモデルです。なんらかの指針を使った設計例として利用してください。
エンティティに着目して分析したモデル	認証がユーザのサービス使用に対して制限を掛けるという観点から、制限を掛けるエンティティに着目したモデルです。	エンティティを中心に理解しやすいシンプルな作りにしたので、モデリング初心者の方にお勧めです。
状態に着目して分析したモデル	ユーザが認証された・認証されていない、といった状態に着目したモデルです。	状態毎の振舞いの違いを、ステートパターンで実現します。

AAA モデルとは

本章には、AAA モデルの用語が 3 つの設計モデルで共通概念として使用されています。

ここで、AAA モデルの概要を説明します。

AAA モデルとは、認証機能を実現する上で、主だった機能をつぎの 3 要素に分割したモデルを指します。

ユーザを認証する、適切なサービスの提供する、サービス使用の記録を作成する、これらを認証（Authentication）、承認（Authorization）、アカウンティング（Accounting）と分類し、 3 つの頭文字 A をつなげて AAA モデルと呼ばれます。

また AAA モデルは、AAA プロトコルとも言われ、認証、承認、アカウンティングの 3 要素自体と、またその要素間の通信とにおいて秘匿性が設計上考慮されていることを指します。

本章に於いて、AAA のアカウンティングの呼称を、監視と改名しています。これはアカウンティングがユーザ情報（アカウント）と誤解しやすいためです。

機能に着目して分析したモデル

分析モデル

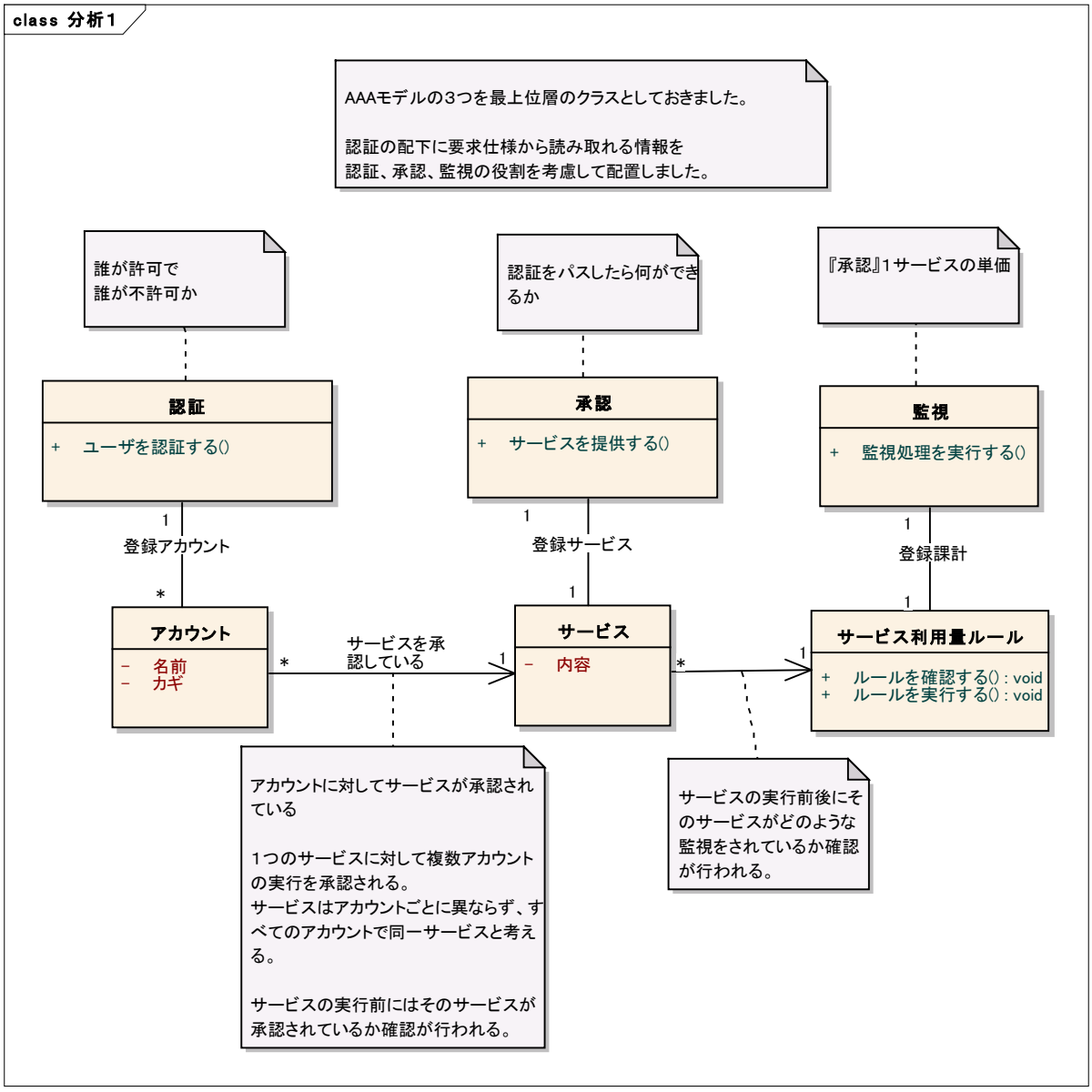
モデリングのコンセプト

本モデルは、広くしられている AAA モデルの構造を模す事を出発点としました。AAA に登場する認証、承認、アカウンティングの３要素が取り扱う情報の責務に着目することを本モデルのコンセプトとして設計を行いました。

３要素のほかに要求仕様から読み取れる、ユーザの ID カード、ユーザに提供するサービス、ユーザの使用履歴といった情報を抽出し、それを元に分析作業を行いました。

クラス図

分析図その１



最上段には AAA の３要素、認証、承認、監視を配置しました。

認証が扱う情報は、ユーザを識別するための情報です。認証が扱うユーザを識別する情報をアカウントとし、その下に配置しました。

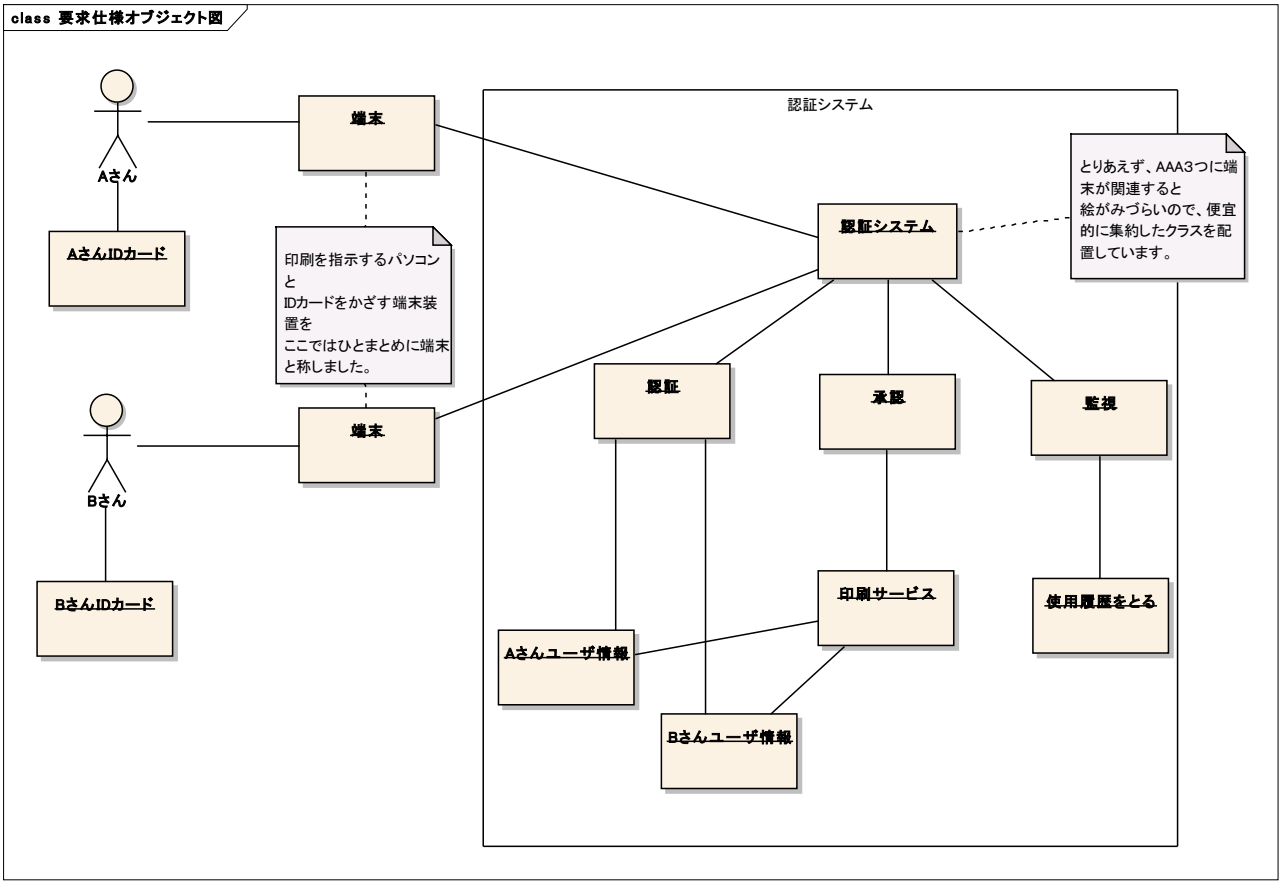
承認が扱う情報は、サービスです。承認が扱うサービスを、その下に配置しました。

監視が扱う情報は、サービス利用（サービス利用後情報）です。

ひとまず分析作業はここまでとして、この後シーケンス検討、PIM モデル作成を通して必要なクラスを抽出します。

オブジェクト図

分析図その１におけるオブジェクト構成

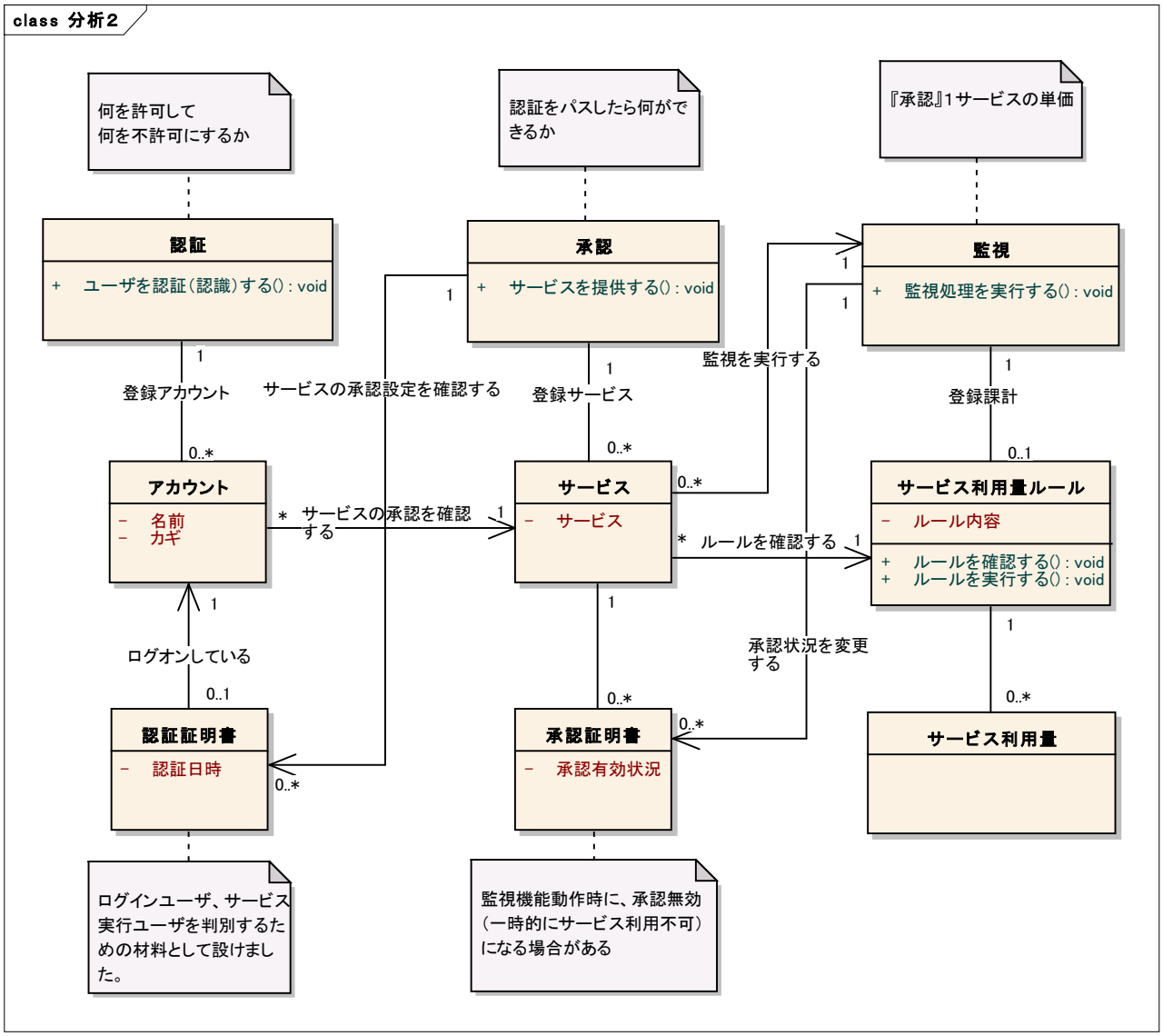


要求仕様の状況をひとまずオブジェクト図にしてみました。クラス図その１の登場人物でひとまず足りている模様です。

クラス図

分析図その 2

分析図その 1 をもとに、さらにクラスを抽出



シーケンスの検討や PIM モデルの検討を通してクラス図は上記のように変化しました。以下の追加された最下段のクラスの責務と説明となります。

認証証明書：ログインアカウントと非ログインアカウントを区別するためにログイン時に生成されます。

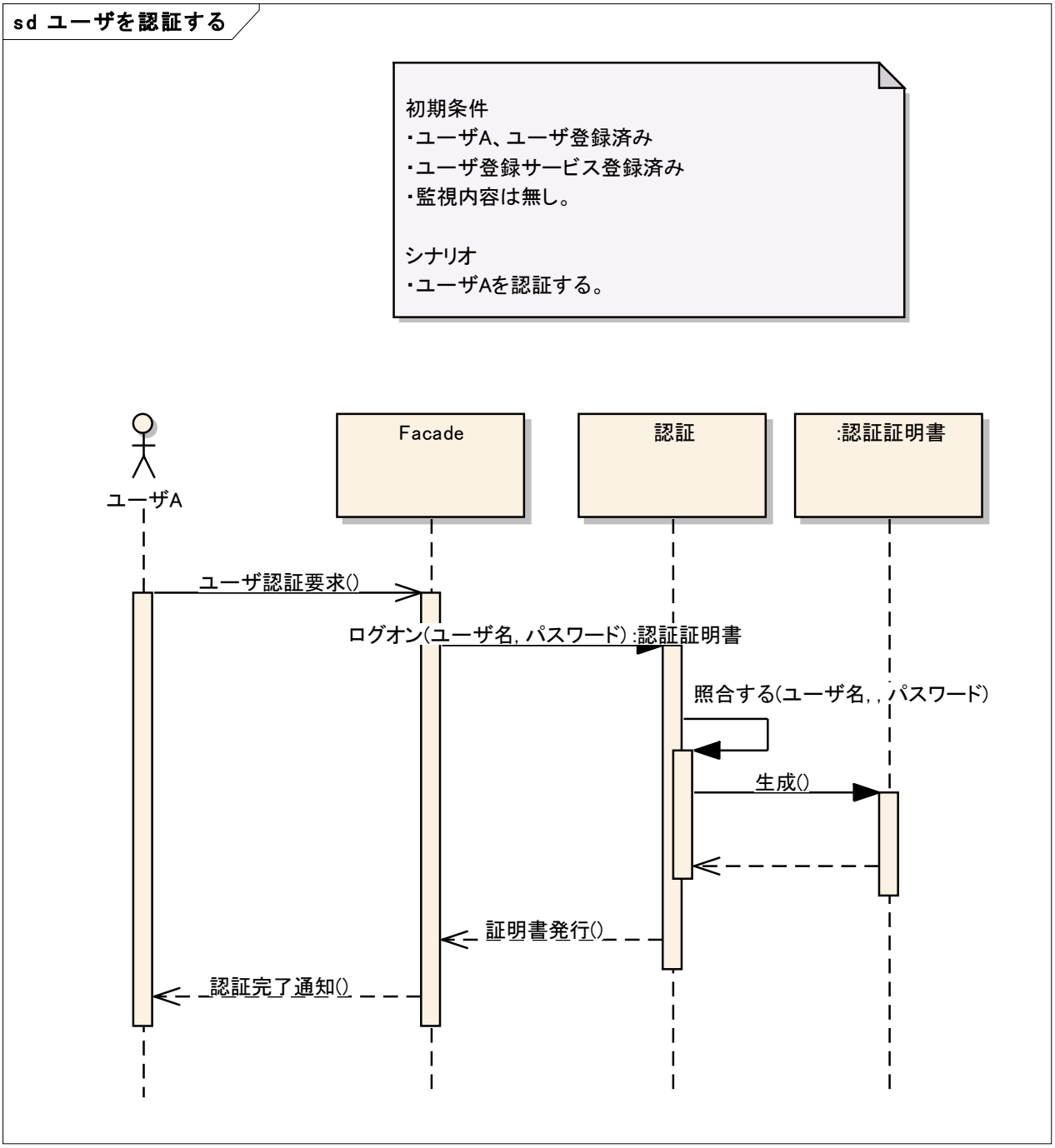
承認証明書：サービス実行前にサービスの実行権が監視からも認められているか確認するクラスです。例えば監視条件を満たさない場合（課金不足など）、サービスの実行権を失う状況を実現します。

サービス利用：監視の使用記録などの実データとなります。

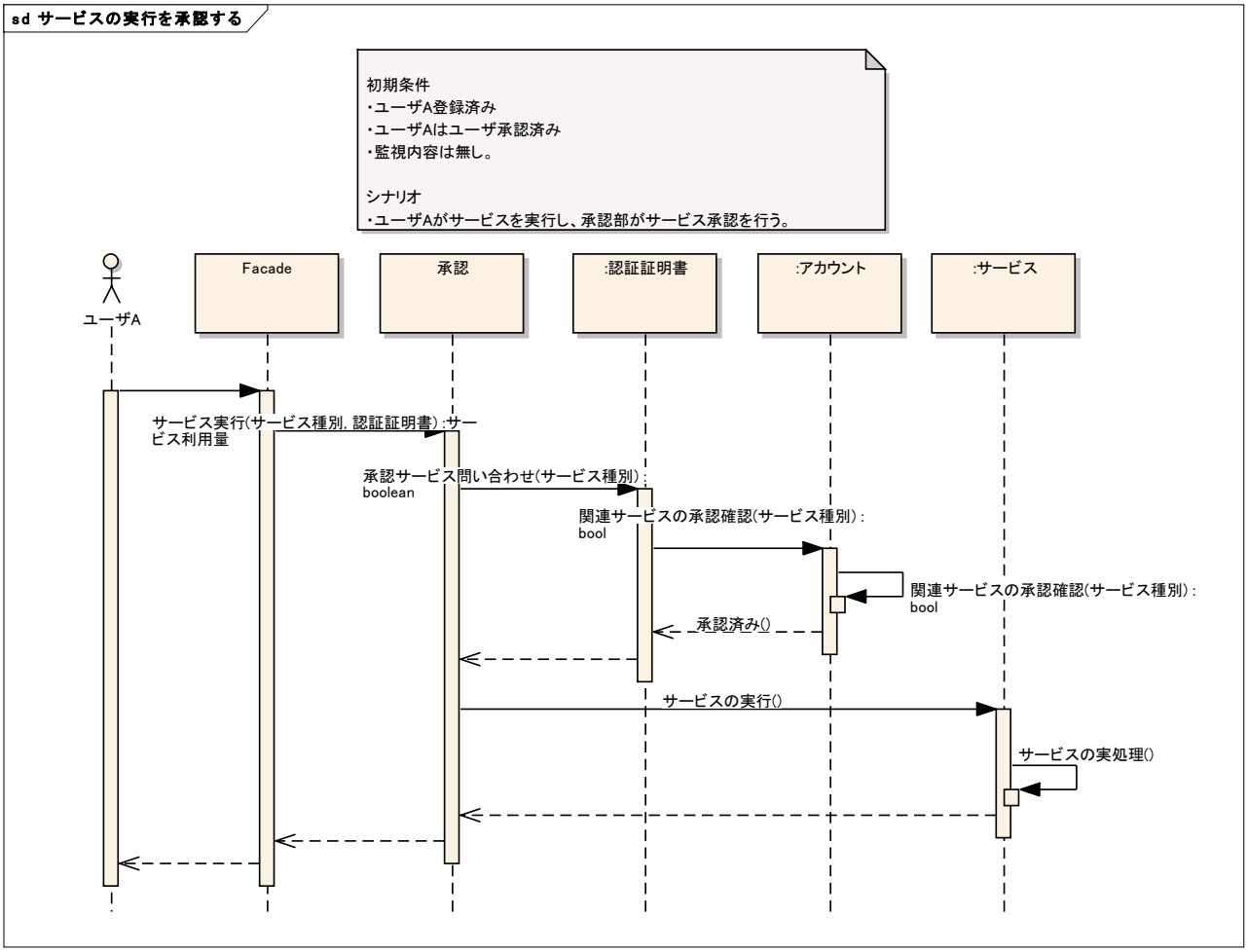
つぎにユースケースに対応したシーケンス図を示します。インタフェースは先の PIM モデルのものを利用しています。

シーケンス図

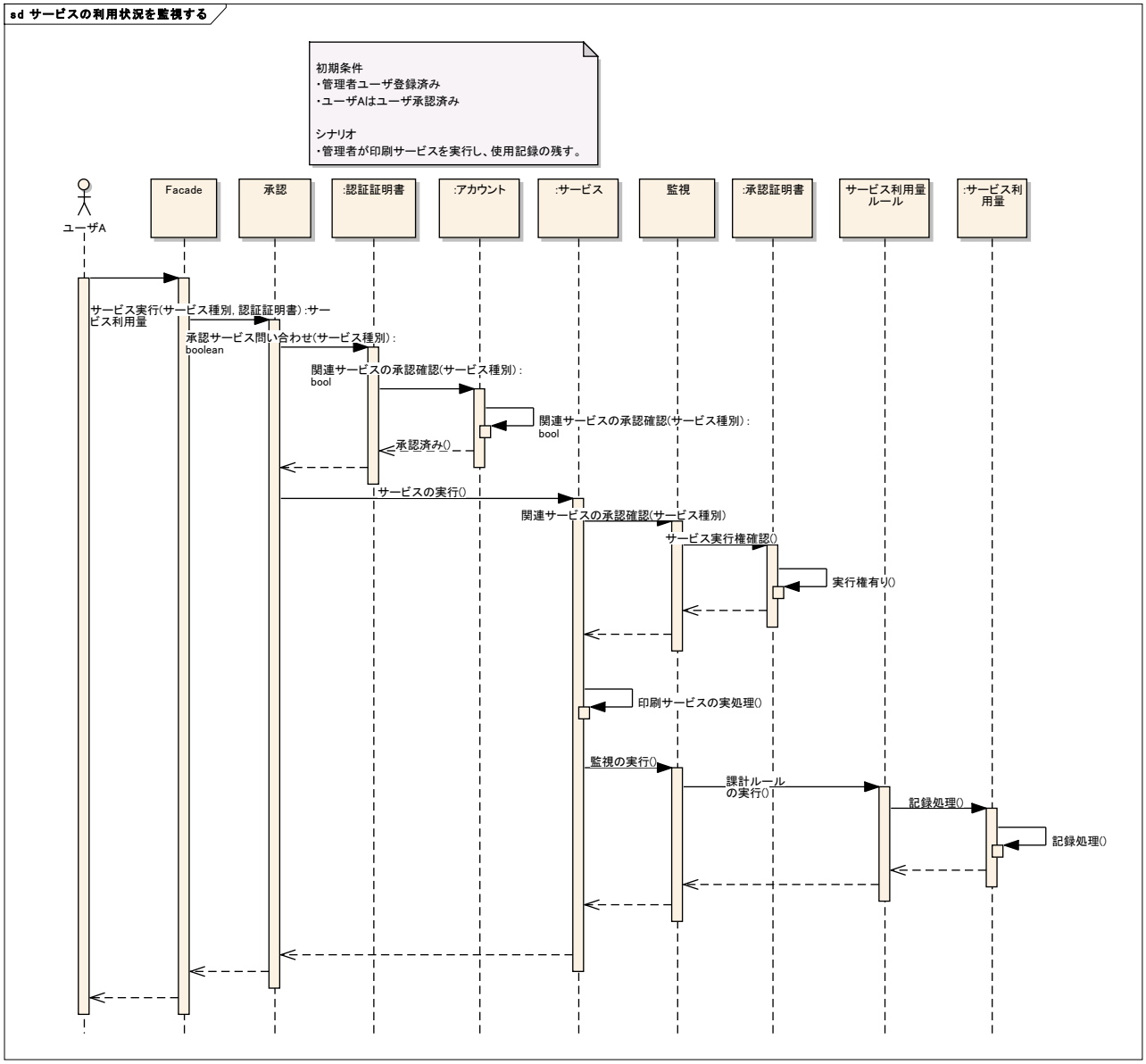
ユーザを認証する際のシーケンス



サービスの実行を承認する際のシーケンス

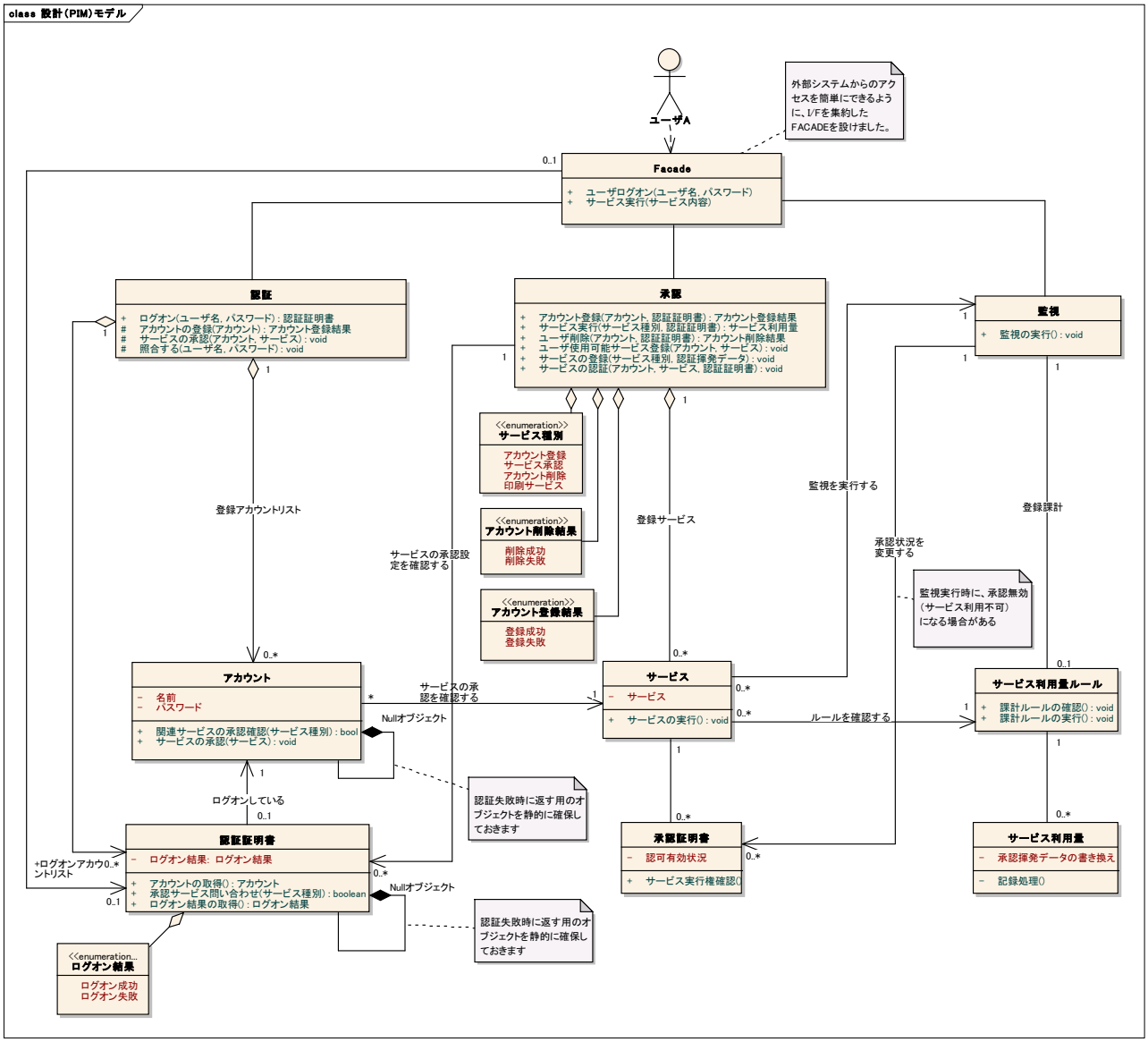


サービスの利用状況を監視するシーケンス



PIM 設計モデル

クラス図



エンティティに着目して分析したモデル

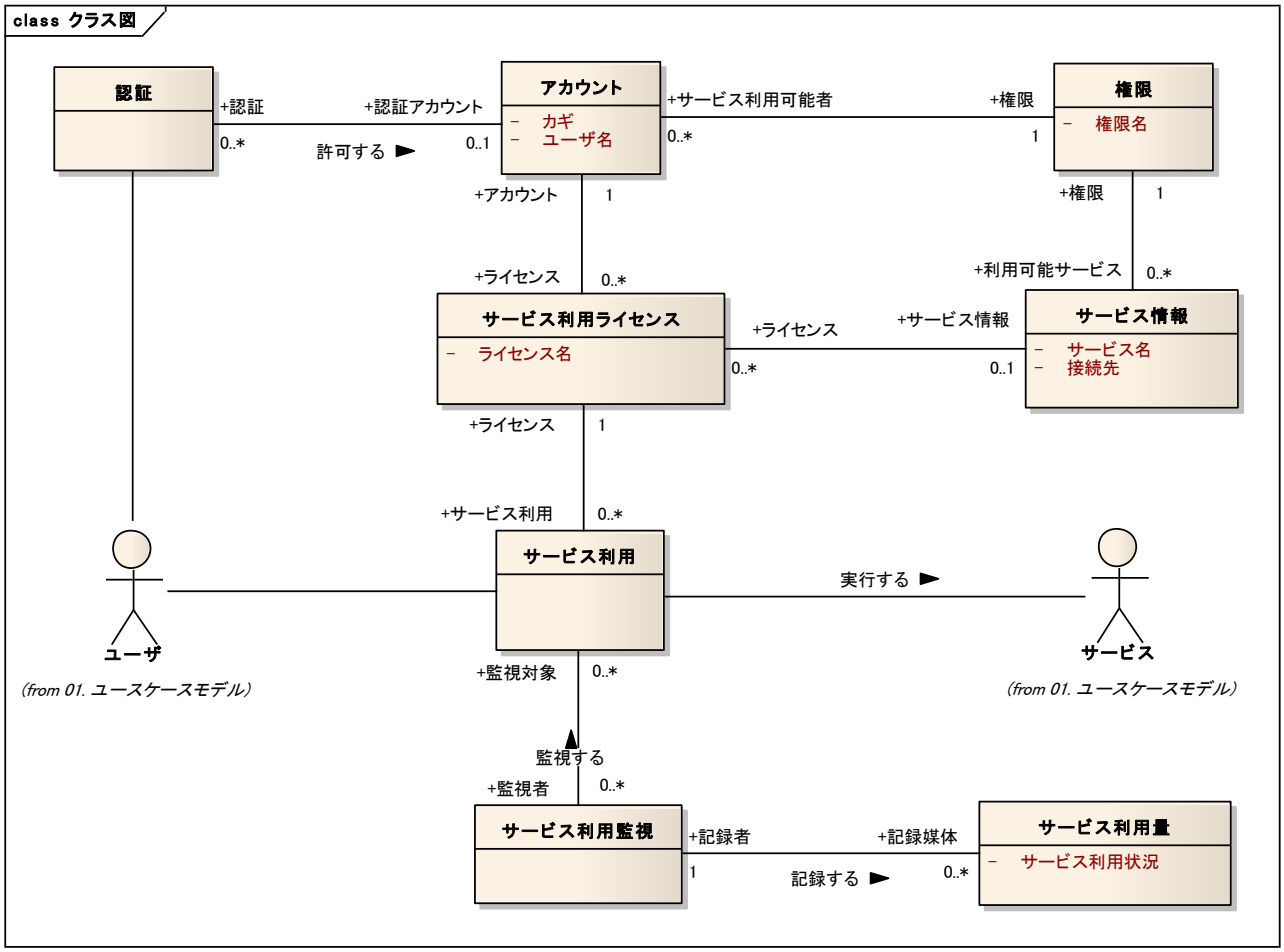
分析モデル

モデリングのコンセプト

認証機能は、ユーザがサービスを使用したいという要求に対して、正当なユーザか？正当なサービスか？正しく使用されているか？といった観点からサービス使用に対して制限を掛けるという機能になります。この事からユーザがサービスを利用するまでに、ユーザにサービス利用の制限をかけるエンティティが幾つか出現すると考えられます。

これらのエンティティを明確化することをモデリングの出発点として分析モデルを作成しました。

クラス図



このクラス図において、「認証」「サービス利用」「サービス利用監視」がそれぞれ認証、承認、監視の役割に当たります。これら以外の情報がサービス利用に制限をかけるためのエンティティとなります。制限をかけるためにどのような情報が必要かという点を検討するために、サービスの利用時に制限するために行う確認事項を洗い出してみます。

・正当なユーザか？

・正当なサービスか？

・正当な組み合わせか？

・正当に使用されているか？（監視記録）

このうち、正当な組み合わせに関しては、サービスの実行時刻や事項場所など様々な条件が考えられますが、モデルを簡素化するために、一旦、アカウントとサービスの組み合わせが適切か？というシンプルな確認とします。

上述した確認事項から残りのエンティティを見た場合、

・「アカウント」は、サービス利用者が妥当かどうかを判断するための情報

・「サービス情報」は、正当なサービスを実行するための情報

・「権限」は、どのユーザがどのサービスを実行できるかを示した情報

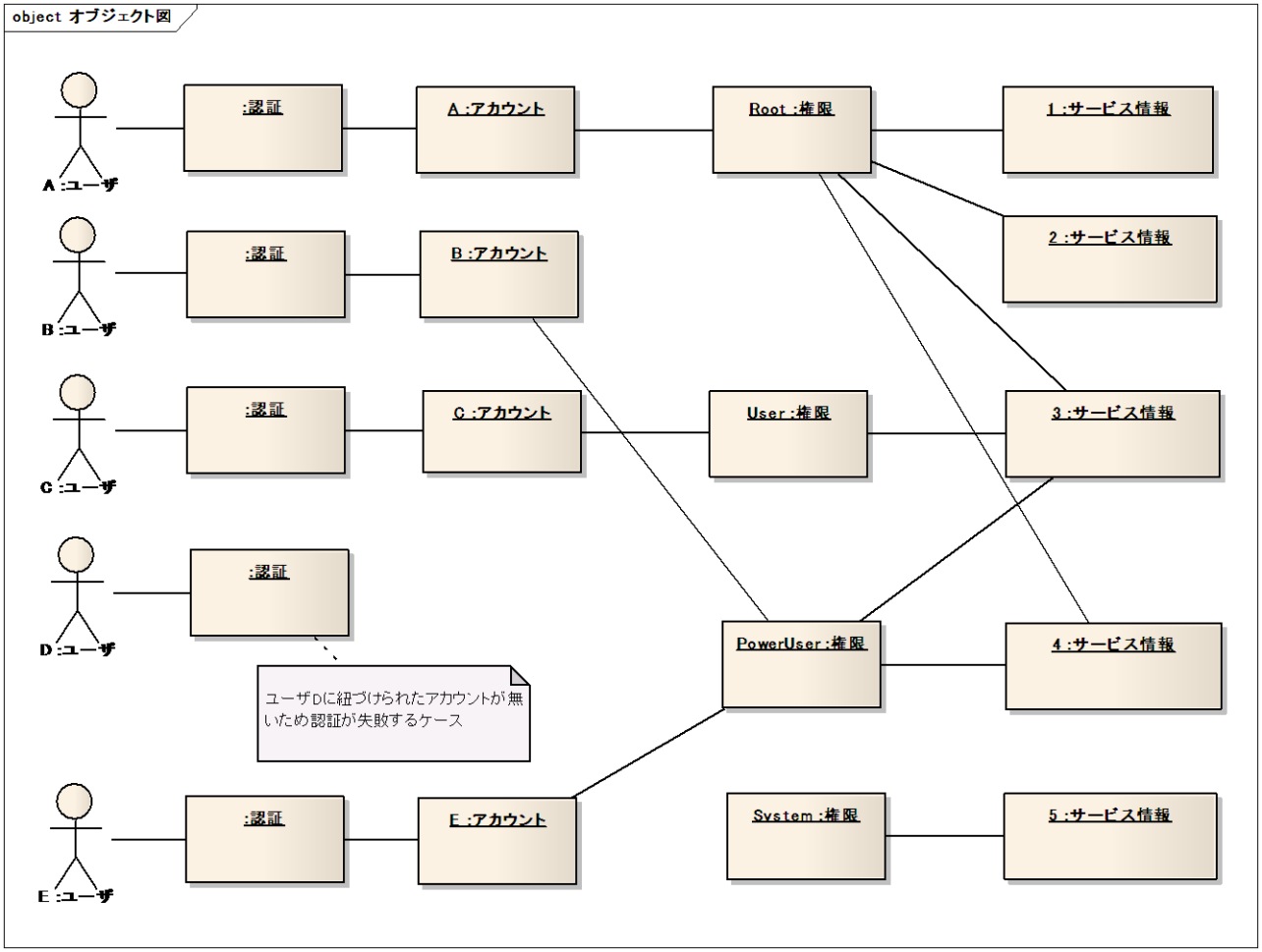
・「課計」は、サービスの利用状況の記録

という位置づけになります。これで先ほどの確認に伴うエンティティは網羅できました。

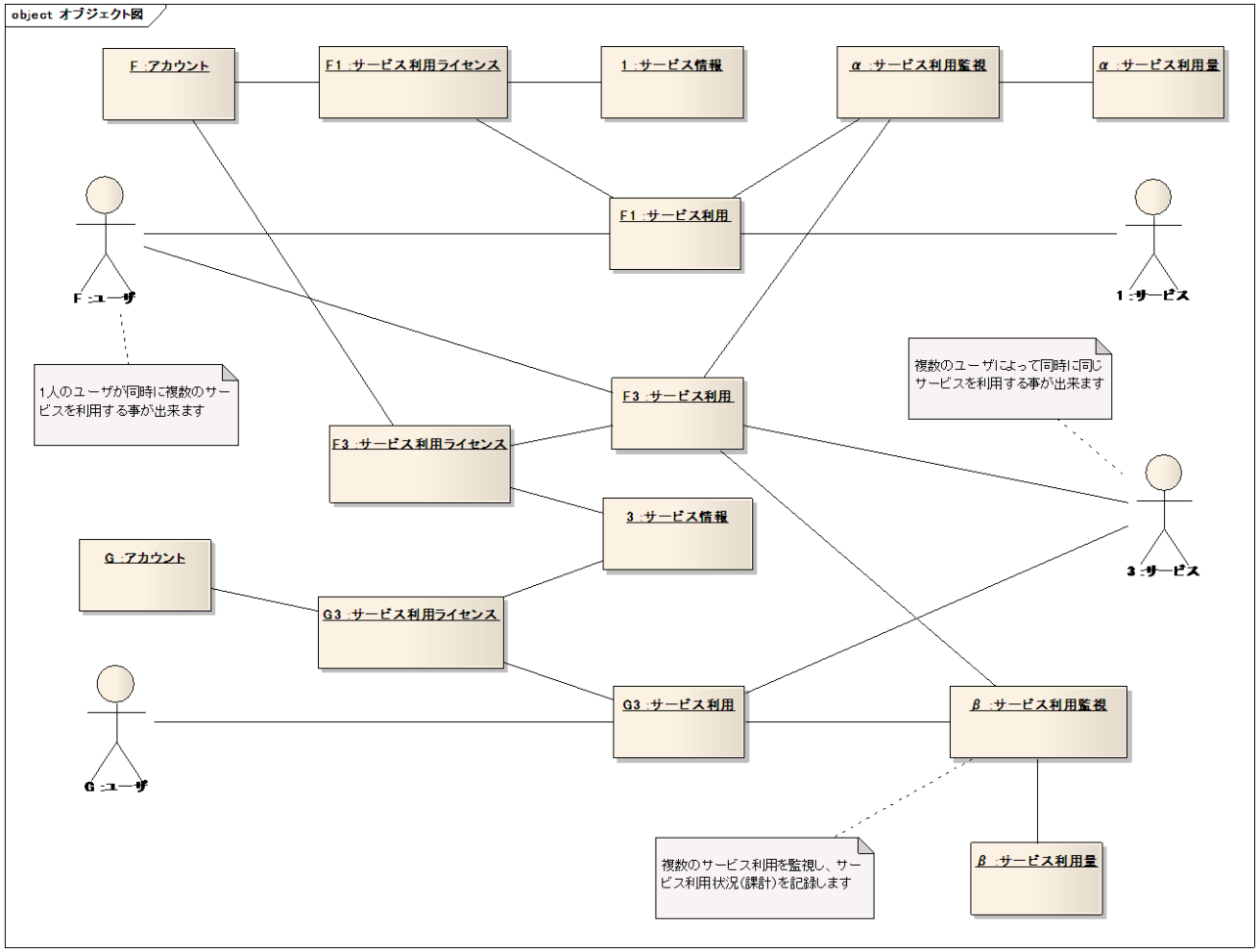
クラス図の中で残された一つの「サービス利用ライセンス」は、ユーザが特定のサービスの実行を許可された事を示す情報として作成しました。

オブジェクト図

サービス利用前のオブジェクト構成



サービス利用中のオブジェクト構成



最初のオブジェクト図が、ユーザから利用可能なサービスを特定するまでのオブジェクト構成を表したもので、二つ目のオブジェクト図が、利用可能なサービスが特定された後、サービスを利用して終了するまでのオブジェクト構成を表したものとなります。

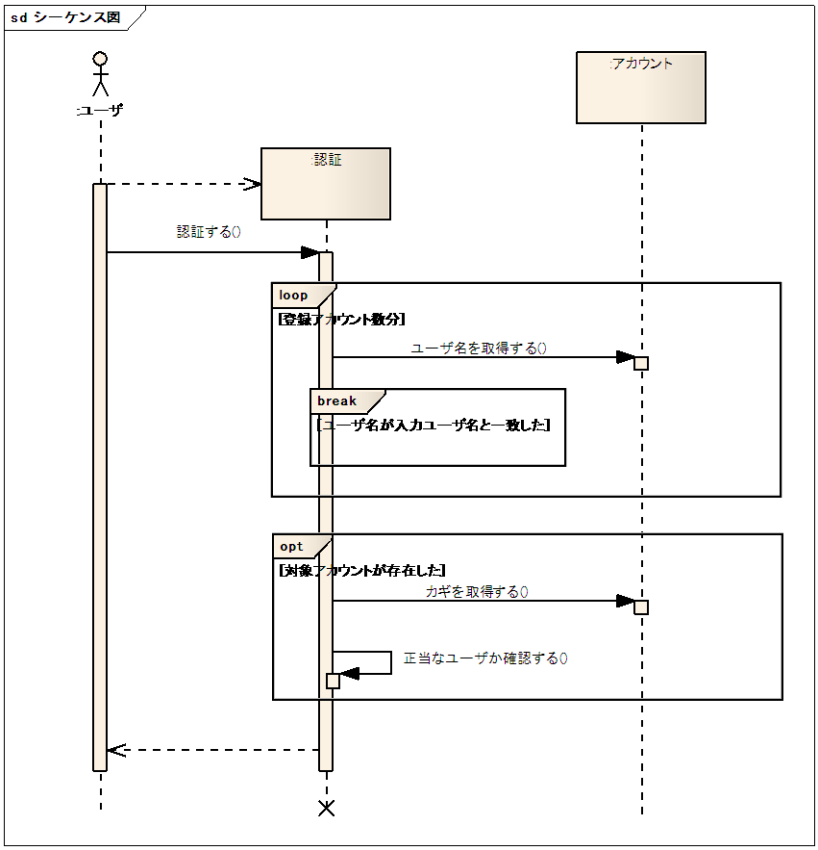
「アカウント」「権限」「サービス情報」という関連は、認証が始まる前から既に存在しています。「認証」や「サービス利用」は、これらのエンティティや関連から、認証、承認といった制限を実現します。

「サービス利用ライセンス」と「アカウント」「サービス情報」間の関連は、承認が出来た後に作られる関連となります。また、「サービス利用監視」「サービス利用」間は、「サービス利用監視」の責務に応じて、様々な「サービス利用」の監視を行う事ができるような関連になっています。

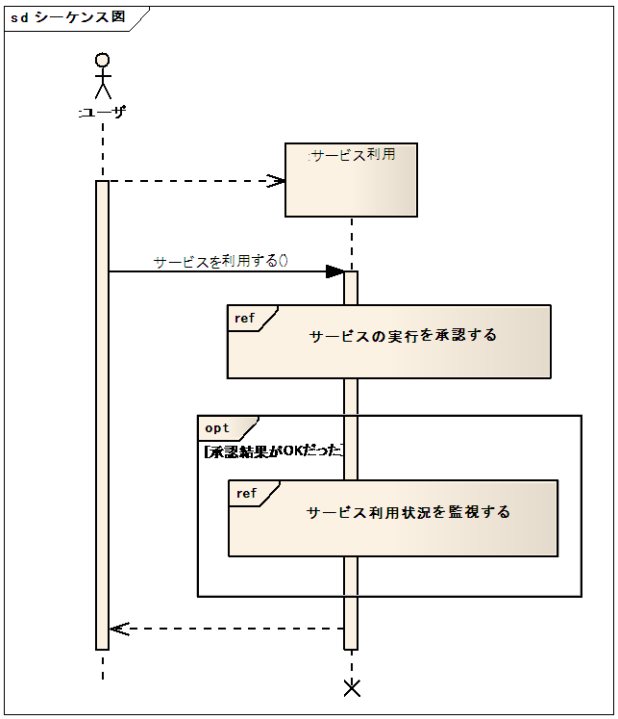
シーケンス図

ユースケースシナリオに基づいて作成したシーケンスは次の 4 つとなります。認証、承認、監視の 3 つの機能が、前述したエンティティで実現できていることが分かります。

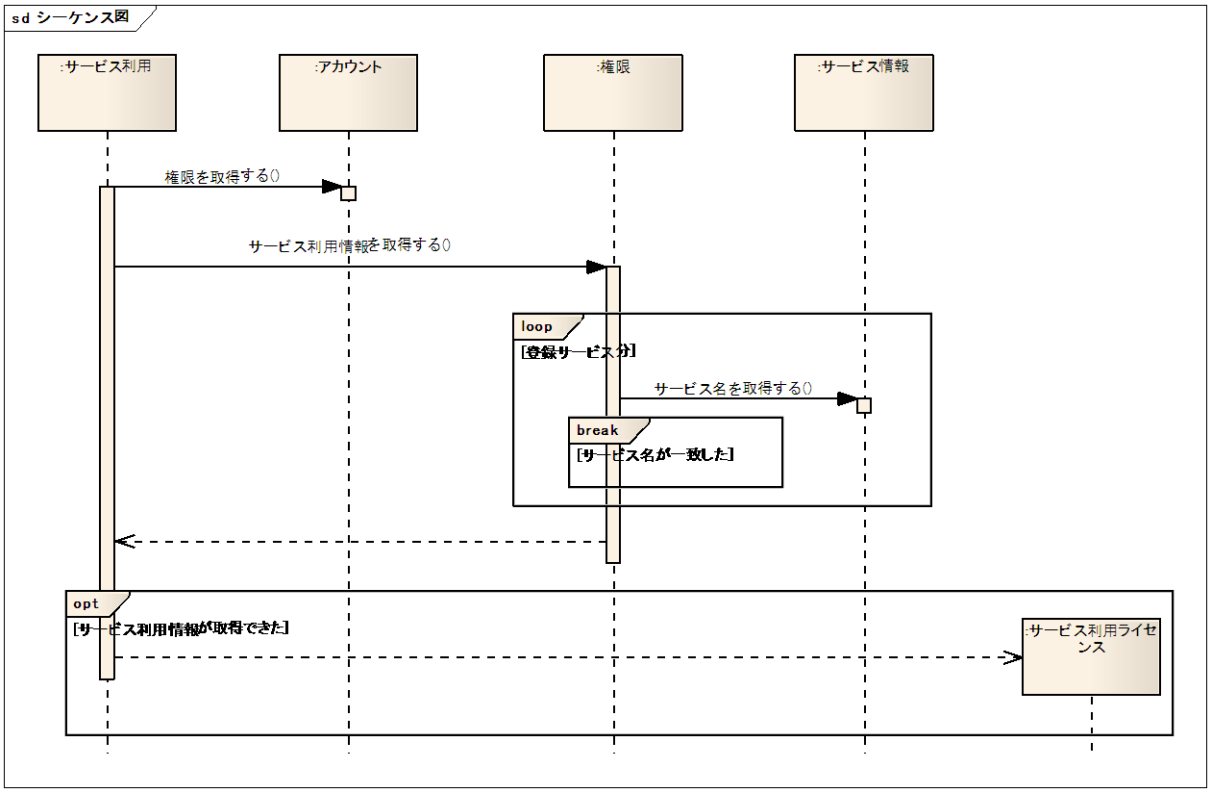
ユーザを認証する際のシーケンス



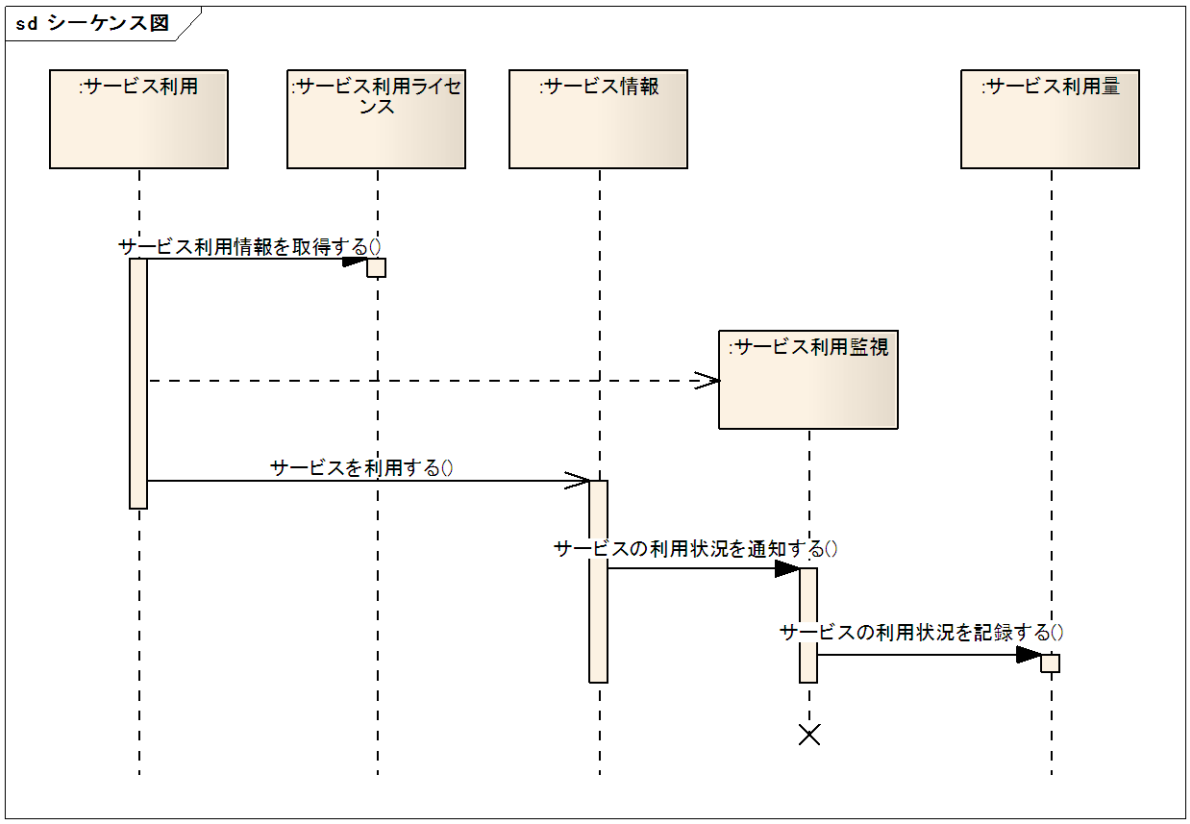
サービスを利用する際のシーケンス



サービスの実行を承認する際のシーケンス



サービス利用状況を監視する際のシーケンス



PIM 設計モデル

モデリングのコンセプト

分析モデルでは、ユーザがサービスを利用するまでに出現するエンティティを明確化することをモデリングの出発点としました。PIM 設計モデルでは、分析モデルで明確になったクラスに対して、責務をインタフェース（Boundary）、機能（Control）、データ（Entity）という 3 つの視点で分割することをモデリングの出発点とします。

このような構造にすることにより、例えば、認証時に PC 画面からのパスワード認証であっても、指紋認証などのように機器デバイスからの認証であってもユーザインタフェース部分を変更するだけで機能部分に手を加えることなく対応する事ができます。

クラス図

分析モデルで出たクラスを、インタフェース・機能・データに当てはめると次のようになります。

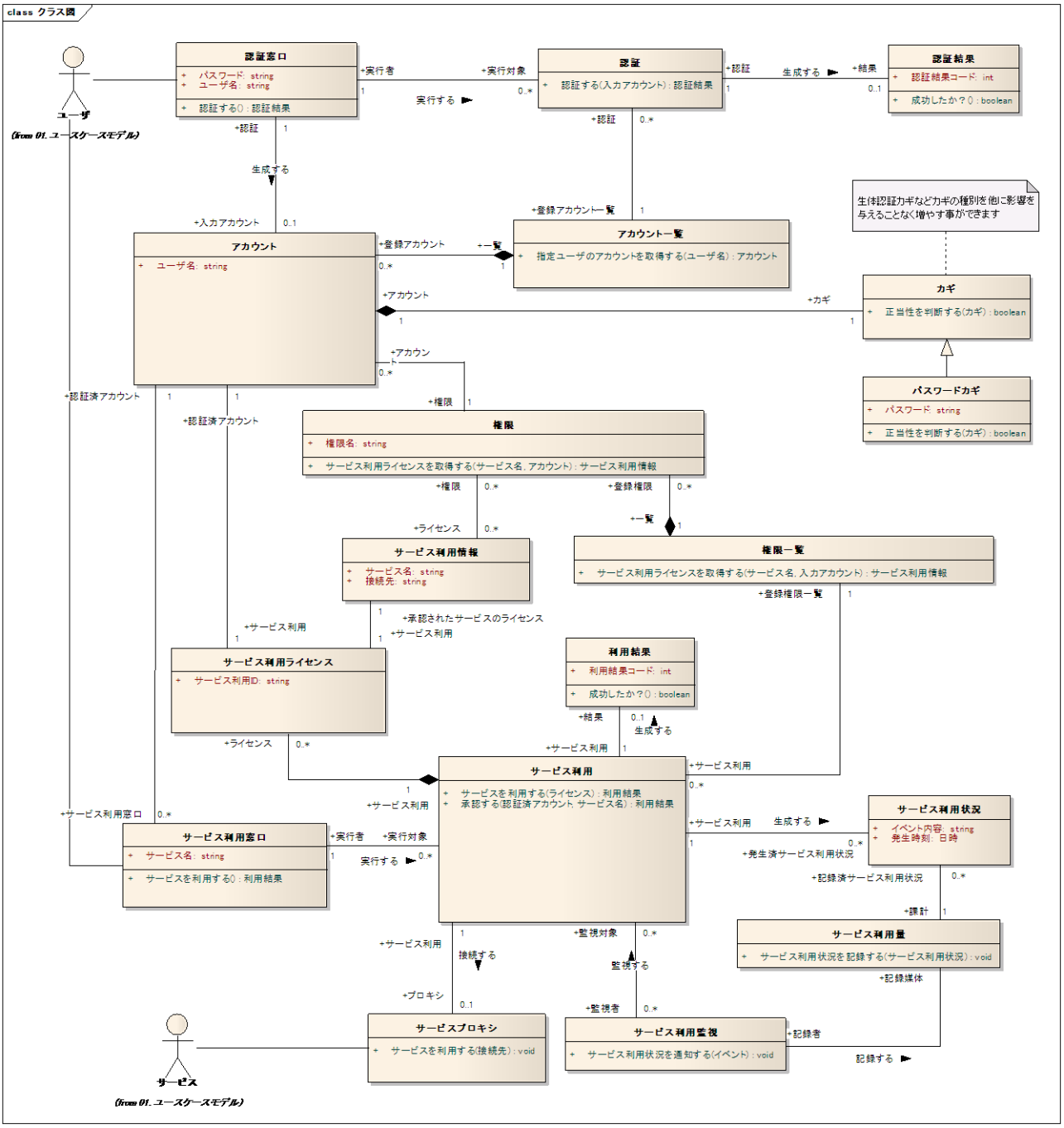
- 「認証」「サービス利用」は、　インタフェース＋機能
- 「サービス利用監視」は、機能
- 「アカウント」「サービス情報」「権限」「課計」「サービス利用ライセンス」は、データ

このため、「認証」と「サービス利用」は、それぞれ次のように責務を分離しました。

- 「認証窓口」「サービス利用窓口」「サービスプロキシ」は、インタフェース
- 「認証」「サービス利用」は、機能

また、「アカウント」「サービス情報」「権限」の静的なデータは、どこか格納しておくものが必要になります。そのため、それぞれ一覧という名前をつけたクラスを導入しました。

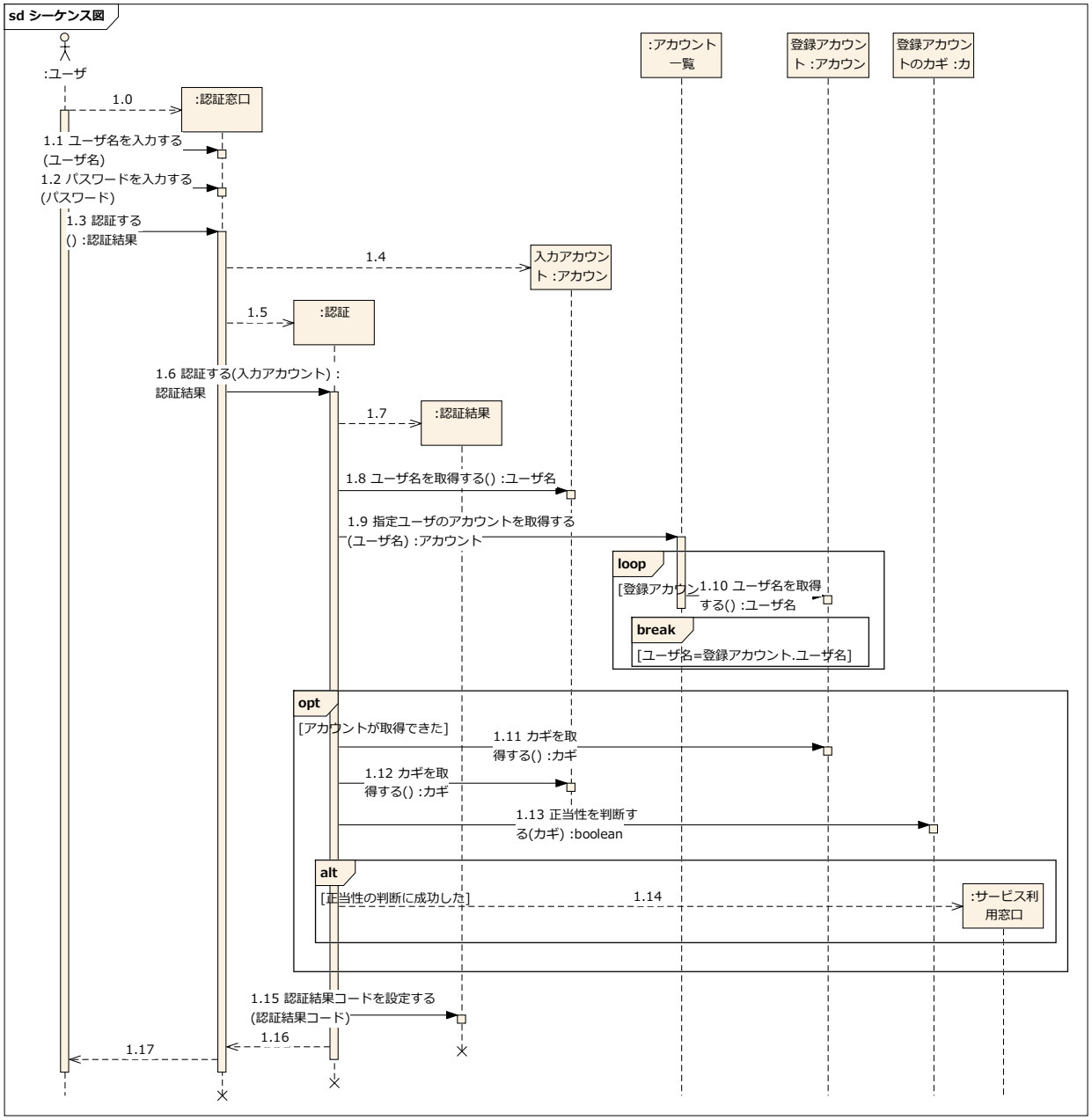
さらに機能の結果保持用のクラスを導入して、操作を追加すると次のようなクラス図となります。


class クラス図
ユーザ
(from 01. ユースケースモデル)
認証窓口
+ パスワード: string
+ ユーザ名: string
+ 認証する() : 認証結果
+実行者
+実行対象
実行する
認証
+ 認証する(入力アカウント) : 認証結果
+認証
生成する
+結果
認証結果
+ 認証結果コード: int
+ 成功したか？() : boolean
+認証
生成する
+入力アカウント
アカウント
+ ユーザ名: string
+登録アカウント
+一覧
アカウント一覧
+ 指定ユーザのアカウントを取得する(ユーザ名) : アカウント
+登録アカウント一覧
生体認証カギなど力ギの種別を他に影響を与えることなく増やす事ができます
カギ
+ 正当性を判断する(カギ) : boolean
パスワードカギ
+ パスワード: string
+ 正当性を判断する(カギ) : boolean
+アカウント
+カギ
+アカウン
+認証済アカウント
+認証済アカウント
+権限
権限
+ 権限名: string
+ サービス利用ライセンスを取得する(サービス名, アカウント) : サービス利用情報
+権限
+登録権限
+一覧
+ライセンス
サービス利用情報
+ サービス名: string
+ 接続先: string
権限一覧
+ サービス利用ライセンスを取得する(サービス名, 入力アカウント) : サービス利用情報
+登録権限一覧
+承認されたサービスのライセンス
+サービス利用
+サービス利用
サービス利用ライセンス
+ サービス利用ID: string
利用結果
+ 利用結果コード: int
+ 成功したか？() : boolean
+結果
生成する
+ライセンス
+サービス利用
+サービス利用
サービス利用
+ サービスを利用する(ライセンス) : 利用結果
+ 承認する(認証済アカウント, サービス名) : 利用結果
+サービス利用
+サービス利用窓口
サービス利用窓口
+ サービス名: string
+ サービスを利用する() : 利用結果
+実行者
+実行対象
実行する
+サービス利用
生成する
サービス利用状況
+ イベント内容: string
+ 発生時刻: 日時
+発生済サービス利用状況
+記録済サービス利用状況
+課計
サービス利用量
+ サービス利用状況を記録する(サービス利用状況) : void
+記録媒体
+サービス利用
接続する
+監視対象
監視する
+プロキシ
+監視者
サービスプロキシ
+ サービスを利用する(接続先) : void
サービス利用監視
+ サービス利用状況を通知する(イベント) : void
+記録者
記録する
サービス
(from 01. ユースケースモデル)

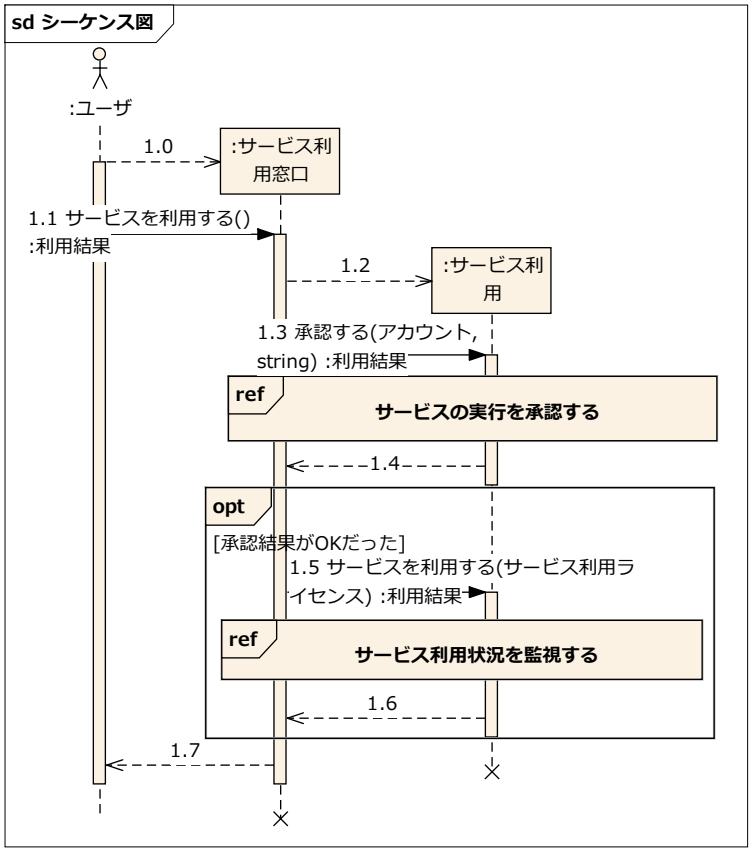
シーケンス図

分析モデルをベースに PIM 設計モデルのクラス構造に基づいて作成したシーケンス図が次の４つの図になります。アクターがユーザインタフェースの責務を持つクラスとしかアクセスしていません。

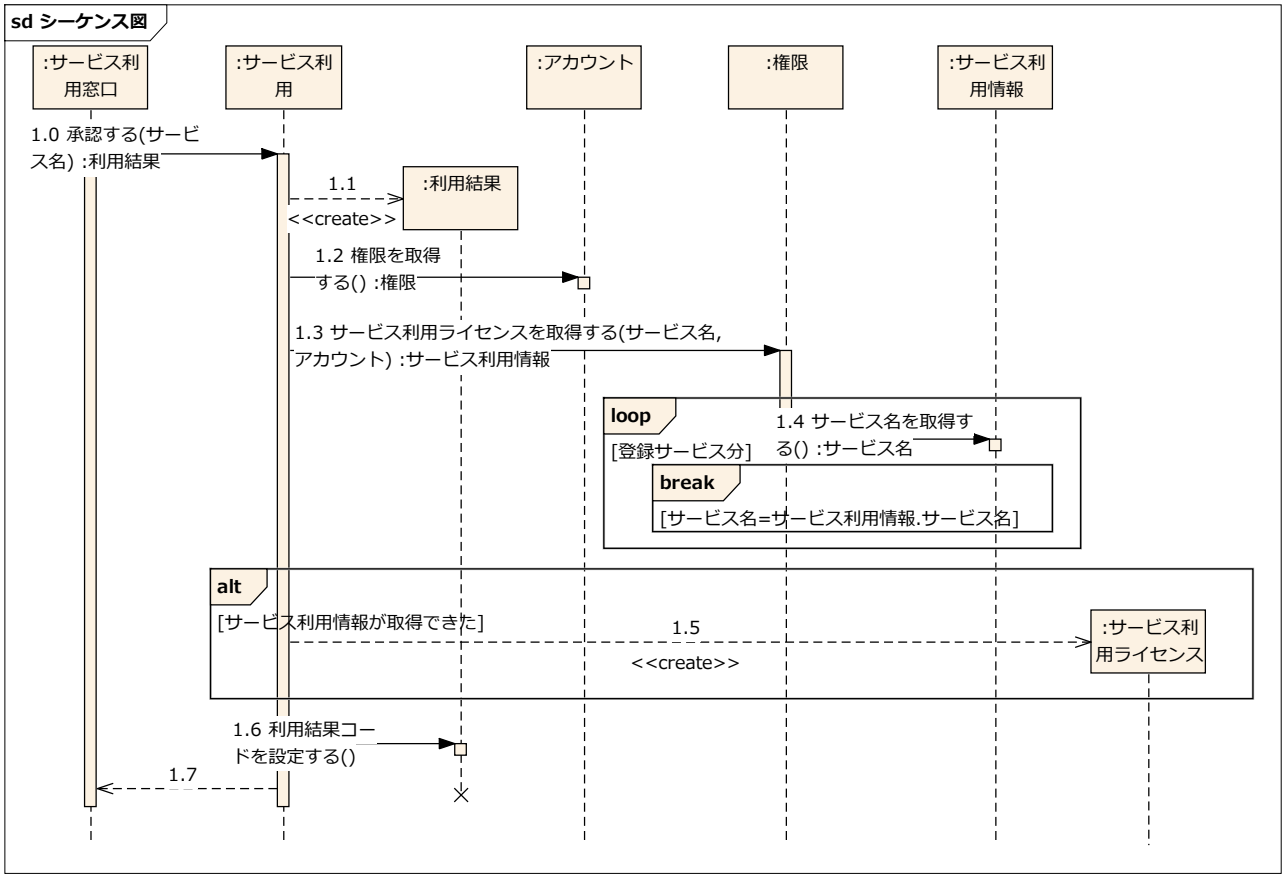
ユーザを認証する際のシーケンス



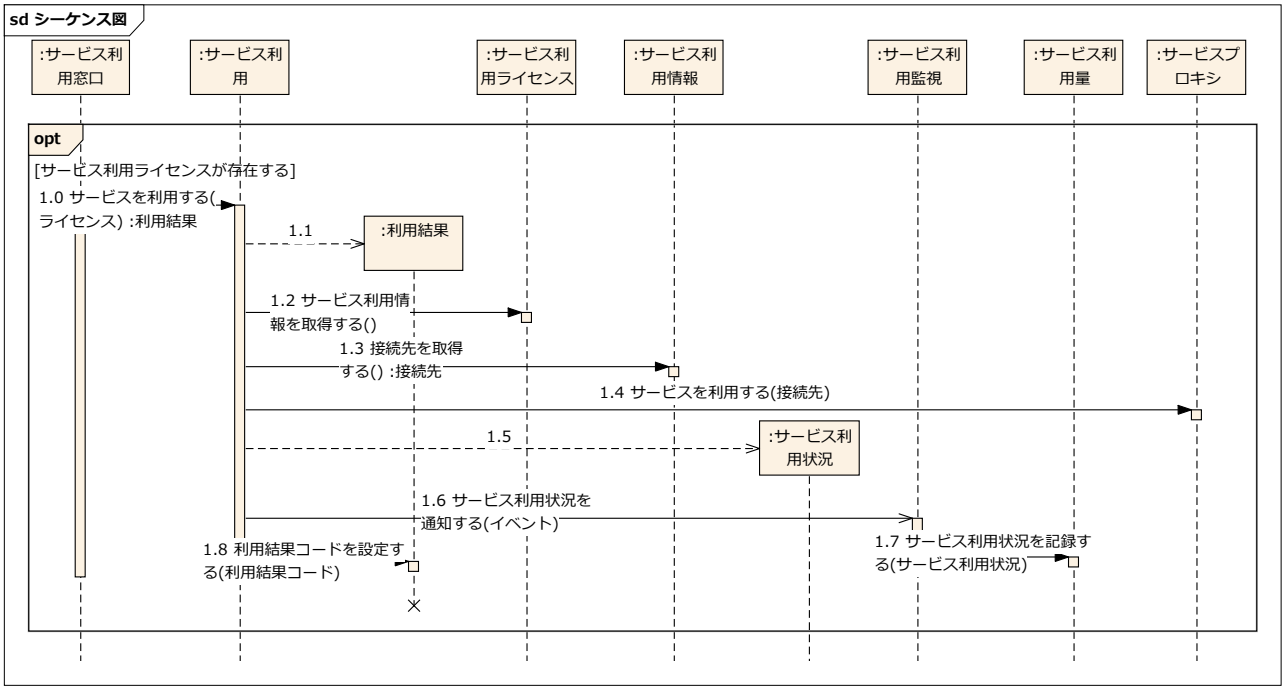
サービスを利用する際のシーケンス



サービスの実行を承認する際のシーケンス



サービス利用状況を監視する際のシーケンス



状態に着目して分析したモデル

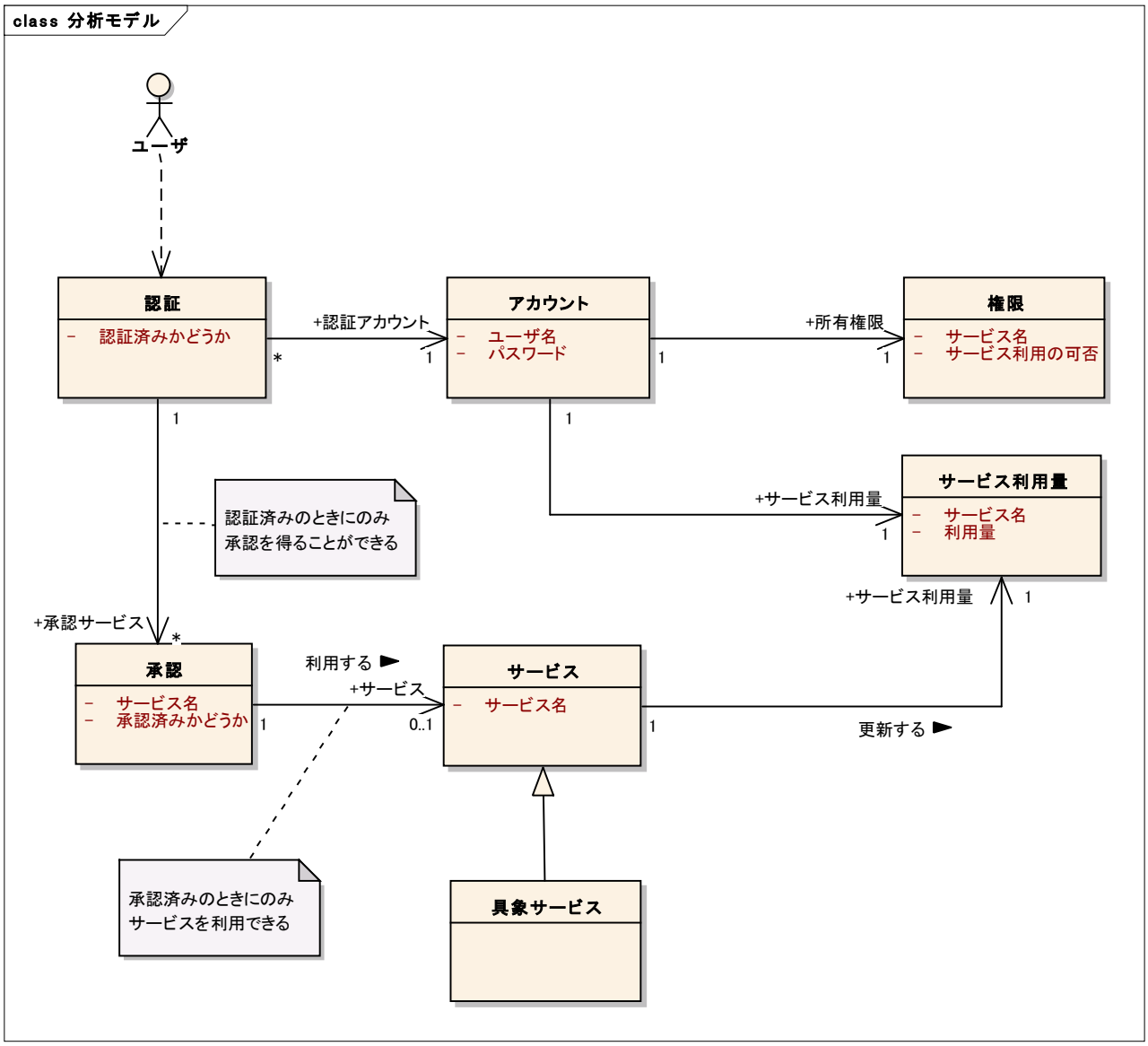
分析モデル

モデリングのコンセプト

ユーザが認証された・認証されていない、といった状態に着目したモデルです。

認証モデルは、ユーザが認証されているか・認証されていないか、によってユーザからの処理依頼に対する振舞いを適切に切替えることが重要です。振舞いの切替えに、ヌケ・モレが無いことを、コーディング時ではなくて、構造設計レベルで抑えようというのがこのモデルのコンセプトです。

クラス図



ユーザとの認証に絡む振舞いを担うクラスとして「認証」を設けます。

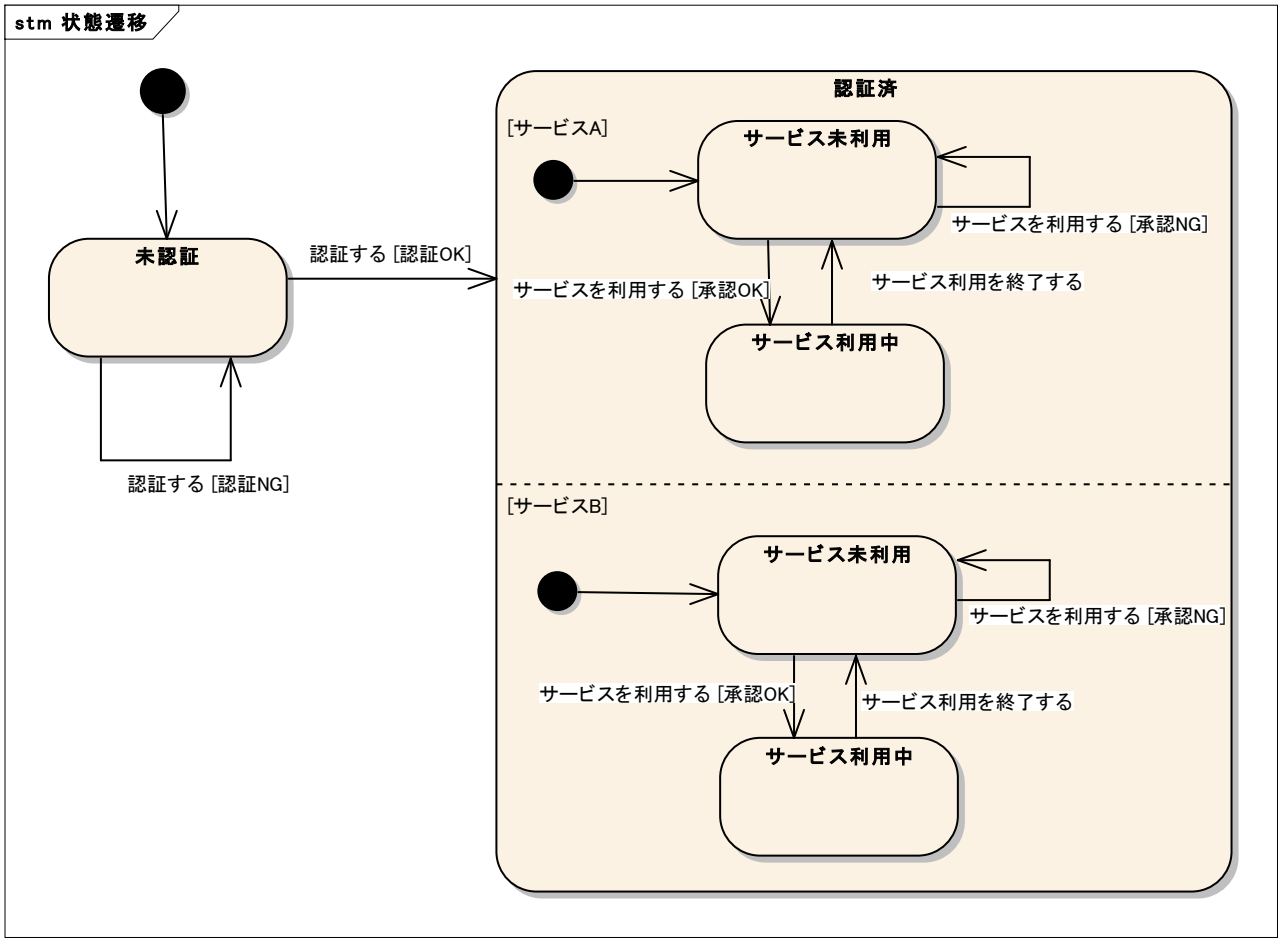
ユーザ毎の情報を保持するクラスとして「アカウント」を設けます。「アカウント」には、認証に必要な情報「権限」と、そのアカウントが過去にサービスを利用した記録である「サービス利用量」を関連付けます。

ユーザが認証されたに利用出来るサービスは複数あり、サービス毎に承認が必要と考え、サービス毎の承認ずみかどうかを表すために「承認」を設けます。

ユーザのサービスを利用によって使用量を監視・記録するために「サービス」を設けます。各々のサービス固有処理は、「サービス」から派生した「具象サービス」が担うものとします。

ユーザによるサービス利用を監視した結果をユーザ毎に分けて記録するために、「サービス」からも、「アカウント」に関連づけた「サービス利用量」に関連を持たせます。

ステートマシン図



認証モデル全体が、ユーザ認証、サービス利用承認によりどのように状態が変化するかを示します。

ユーザが認証されていない状態を「未認証」、ユーザが認証された状態を「認証済」と呼ぶことにます。最初の状態は「未認証」です。ユーザが認証を試みて、システムに正当なユーザであると判断されると「認証済」へ遷移します。

「認証済」状態にネストする形で、サービスを利用していない「サービス未利用」と、サービスを利用中の「サービス利用中」の 2 つの状態があります。「認証済」に遷移した直後は、「サービス未利用」です。これらの状態は、サービス毎にあります。ステートマシン図

では、サービス A とサービス B の 2 つのサービスについて、それぞれ状態を持てることを示しています。

ユーザがサービスを利用しようとすると、システムが認可するかどうか判断します。システムが承認した場合は「サービス利用中」に遷移します。システムが承認しない場合は、「サービス未利用」に留まります。

ただし、「サービス未利用」と「サービス利用中」の状態は、システムに 1 つではなく、サービス毎に状態をもつ必要があります。

以下に、各状態における振る舞いの違いを示します。振る舞いが未定義の部分は―とし、「何もしない」とします。

状態		イベント		
		ユーザを認証する	サービスを利用する	サービスの利用終了
未認証		認証できれば「認証済」へ遷移	―	―
認証済	サービス未利用	―	承認できれば、サービスを起動し、「サービス利用中」へ遷移	―
	サービス利用中	―	―	サービスを終了し、「サービス未利用」へ遷移

例えば、「ユーザを認証する」は、「未認証」のときしか動作せず、それ以外の「認証済」状態では何もしません。

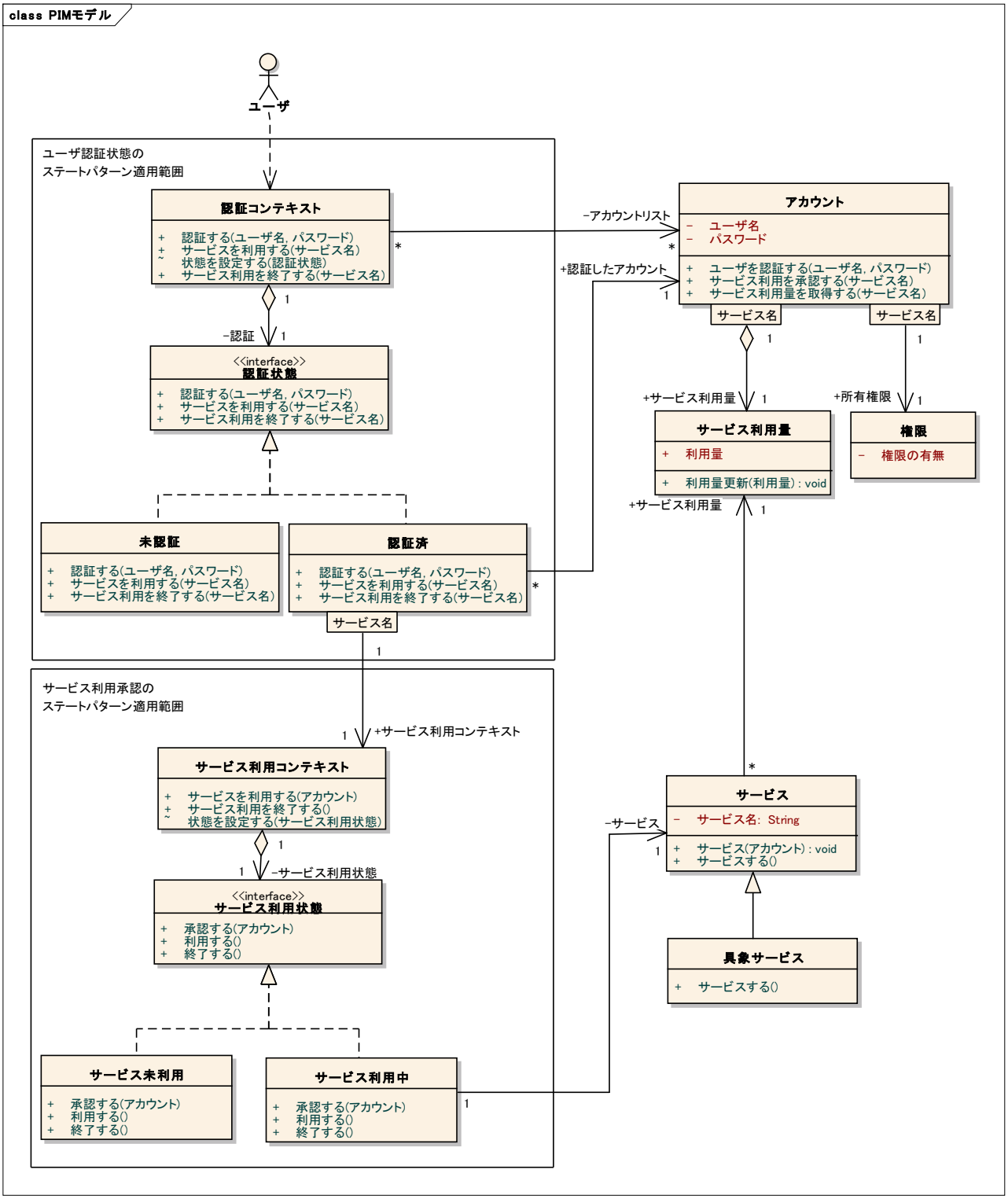
PIM 設計モデル

モデリングのコンセプト

未認証・認証済、サービル未利用・サービス利用中、それぞれの状態を表すために、デザインパターンの１つ、ステートパターンを適用します。

ステートパターンは、振舞いに関するデザインパターンの１つです。ステートパターンを用いることで、状態によって振る舞いを変えるための条件分岐が不要になります。また、状態遷移図に対して状態遷移表がそうであるように、状態毎の振舞いの違いを考えるときのモレ・ヌケを防止する効果があります。

クラス図

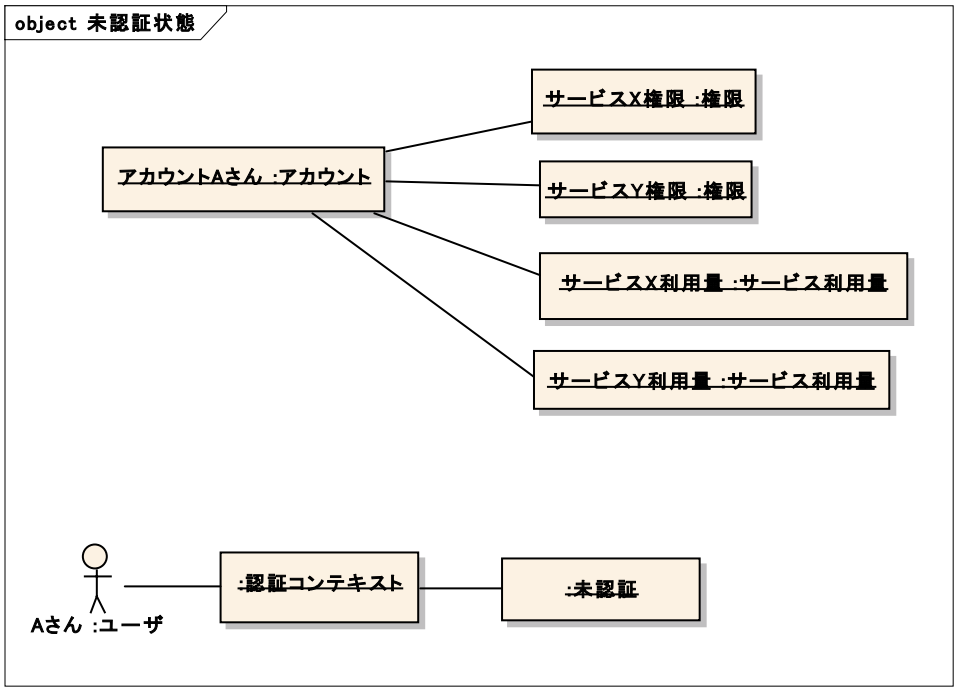


システムに登録されたユーザ情報である「アカウント」には、サービス毎にそのサービスを利用可能か示す「権限」と、サービスを利用したときに計上される「サービス利用料」を関連づけます。

実際のサービスを提供するクラスは、認証モデルから独立して拡張できるように、「具象サービス」として「サービス」クラスから派生させています。

オブジェクト図

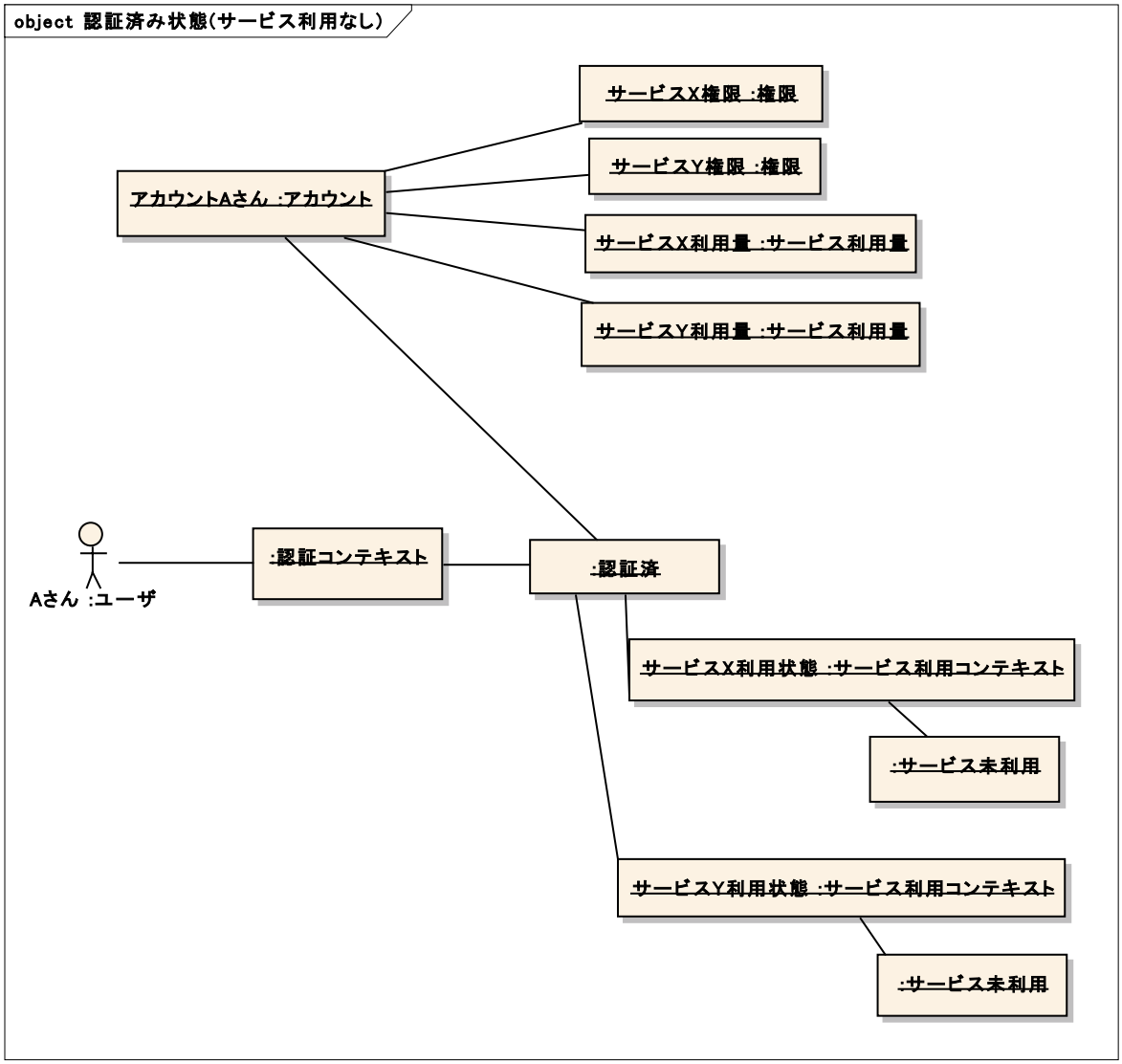
認証していない段階のオブジェクト構成



このオブジェクト図では、1 つのアカウント「アカウント A さん」があり、2 つのサービス「サービス X」と「サービス Y」があるシステムを示したものです。

未認証状態では、ユーザ A さんはアカウントの情報を辿ることも、サービスを使うこともできません。

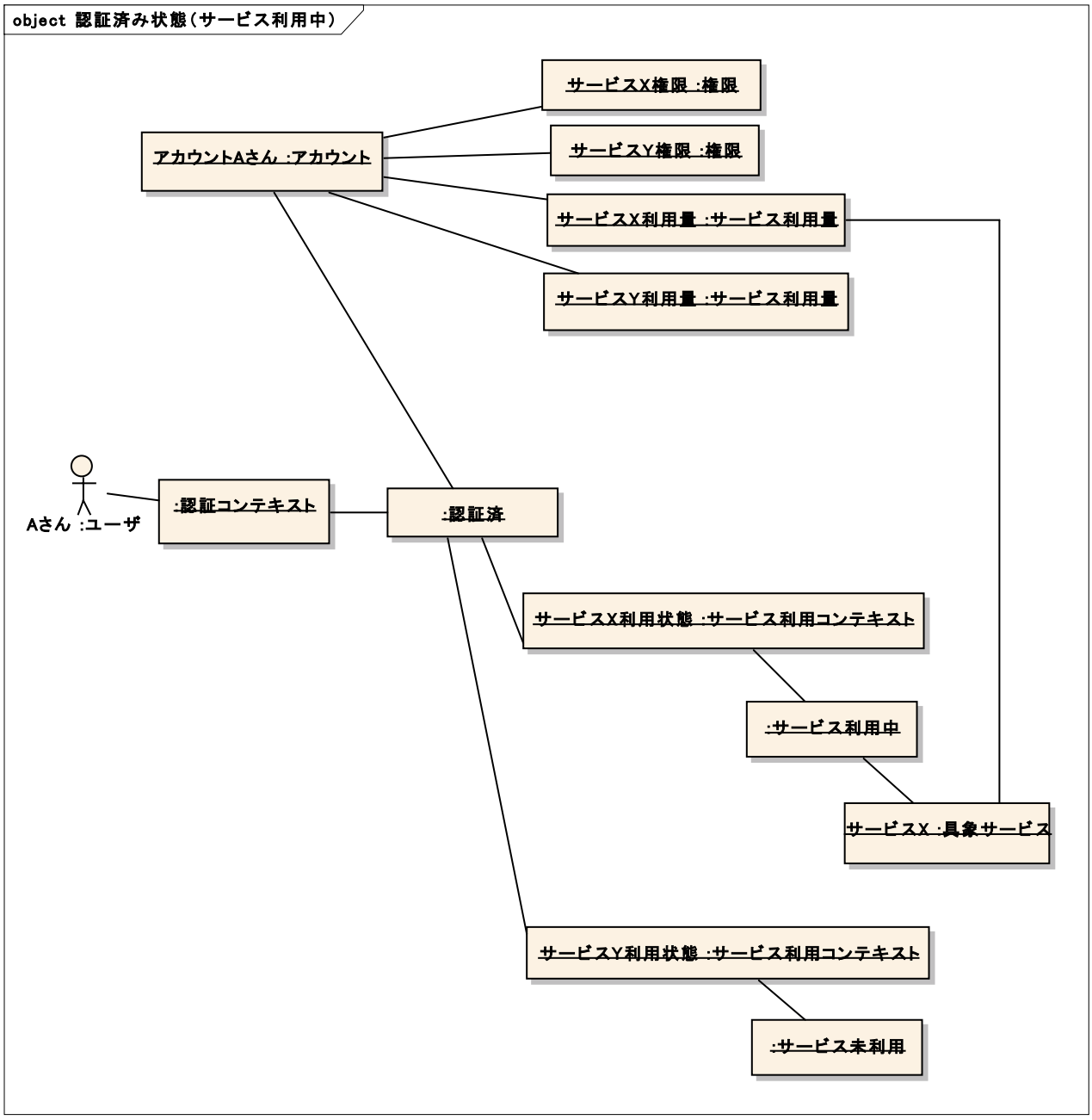
認証済みでサービスが利用されていない状態のオブジェクト構成



認証状態では、「認証済み」のオブジェクトが「アカウント A さん」との関連を持ち、「ユーザ A さん」が、アカウント情報を辿れるようになります。

また、「サービス X」と「サービス Y」の利用コンテキストが生成され、サービスの利用を試みることができます。ただし、「サービス利用コンテキスト」には「サービス未利用」状態が関連付けられており、承認処理を経由しなければサービスを利用することができなくなっています。

認証済みでサービス X を利用中のオブジェクト構成



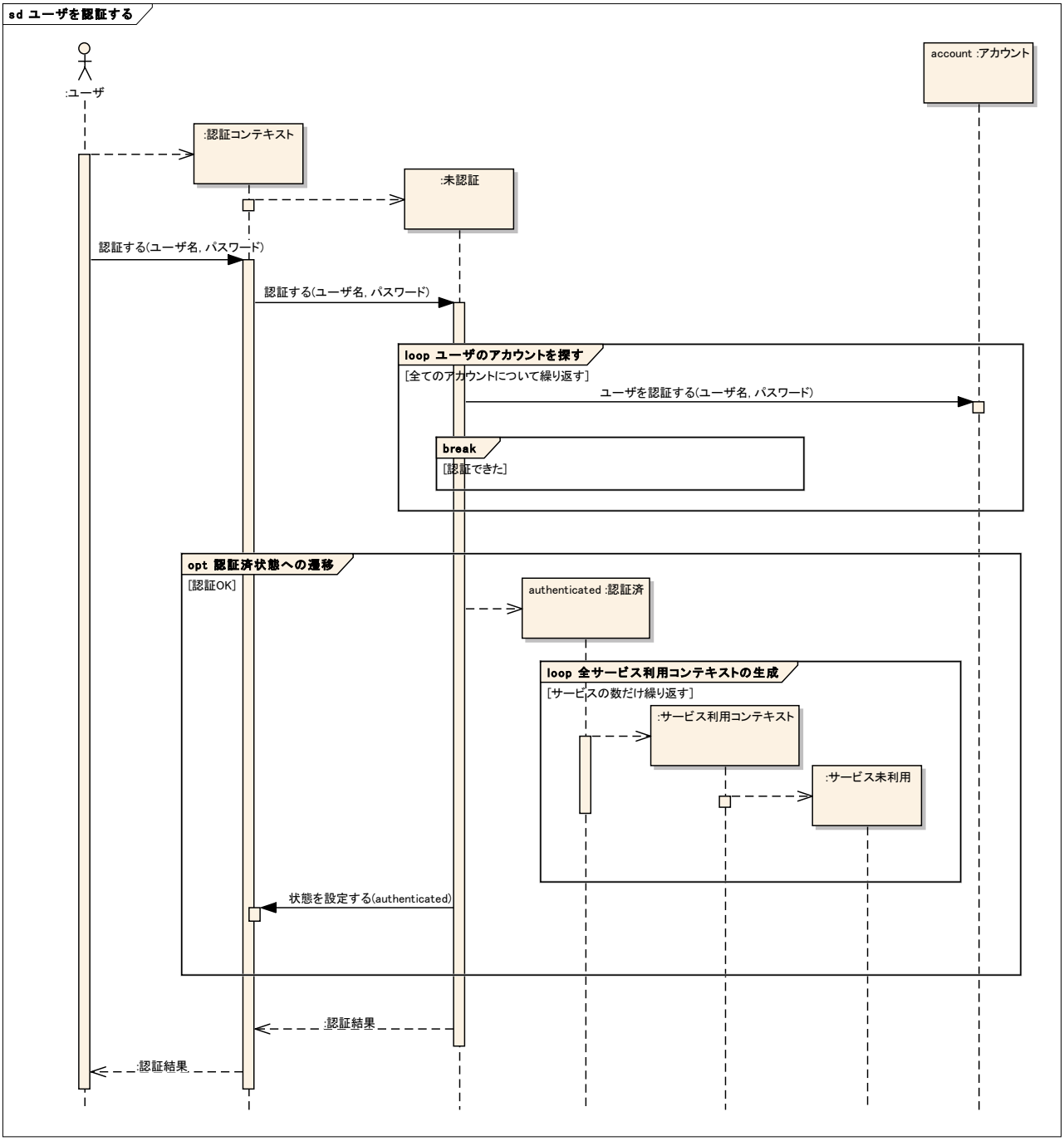
この図は、「ユーザ A さん」が、「サービス X」を利用している状態を示しています。

サービス X のためのサービス利用コンテキスト「サービス X 利用状態」には、「サービス利用中」オブジェクトが関連付けられ、その先に「サービス X」オブジェクトが関連付けられています。また、「サービス X」は、「サービス X 利用量」との関連を持ち、適宜、サービス利用量を更新します。

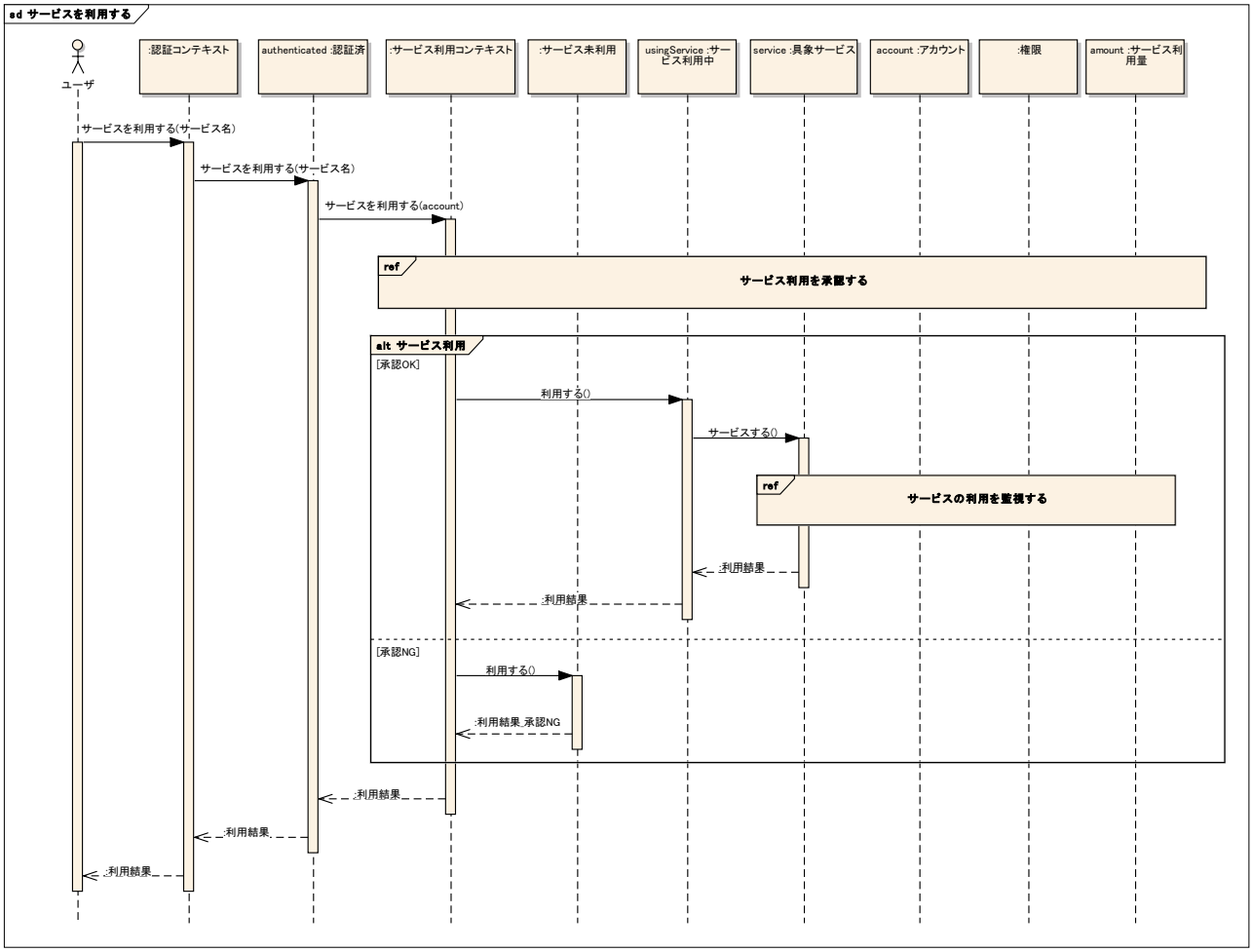
ステートパターンを適用し、状態に応じたオブジェクトを組み合わせて認証モデルを実現します。

シーケンス図

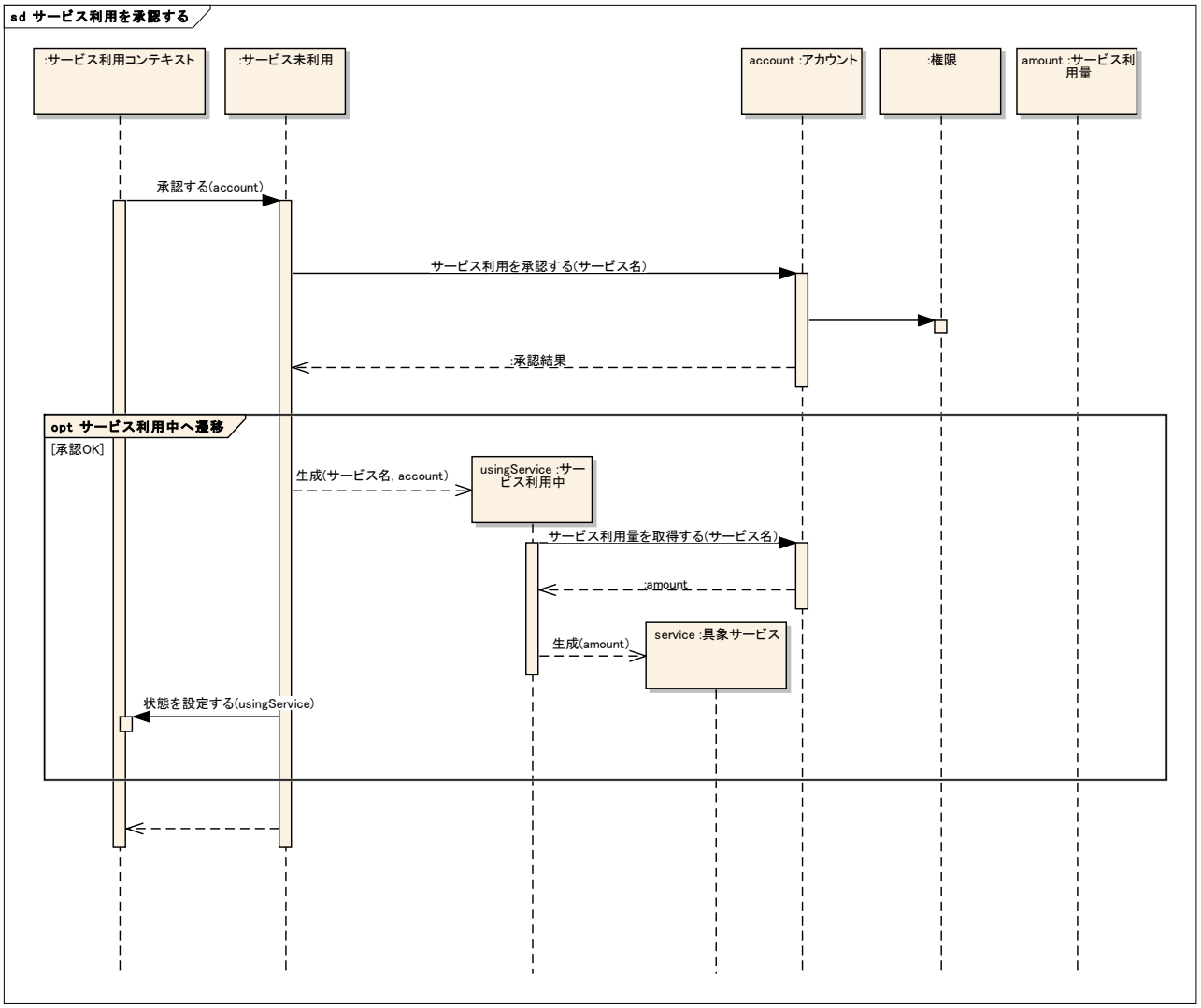
ユーザを認証する際のシーケンス



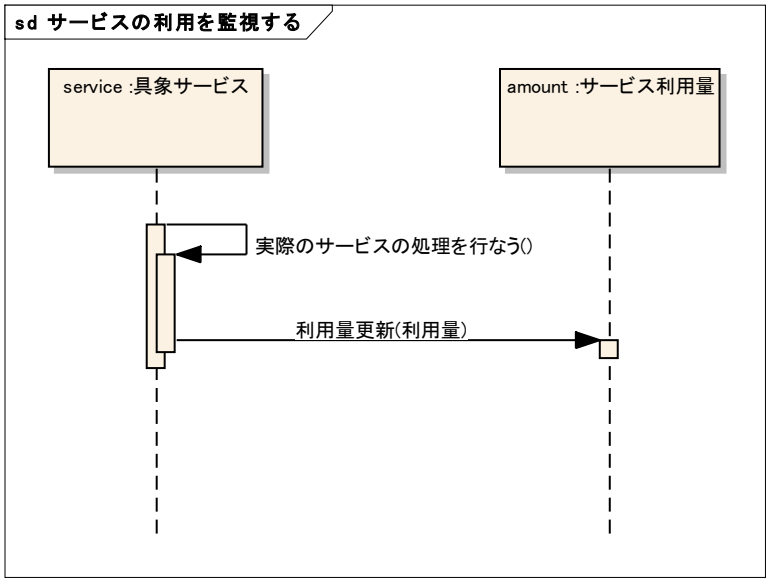
サービスを利用する際のシーケンス



サービスの実行を承認する際のシーケンス

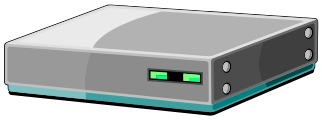


サービス利用状況を監視する際のシーケンス



自己診断

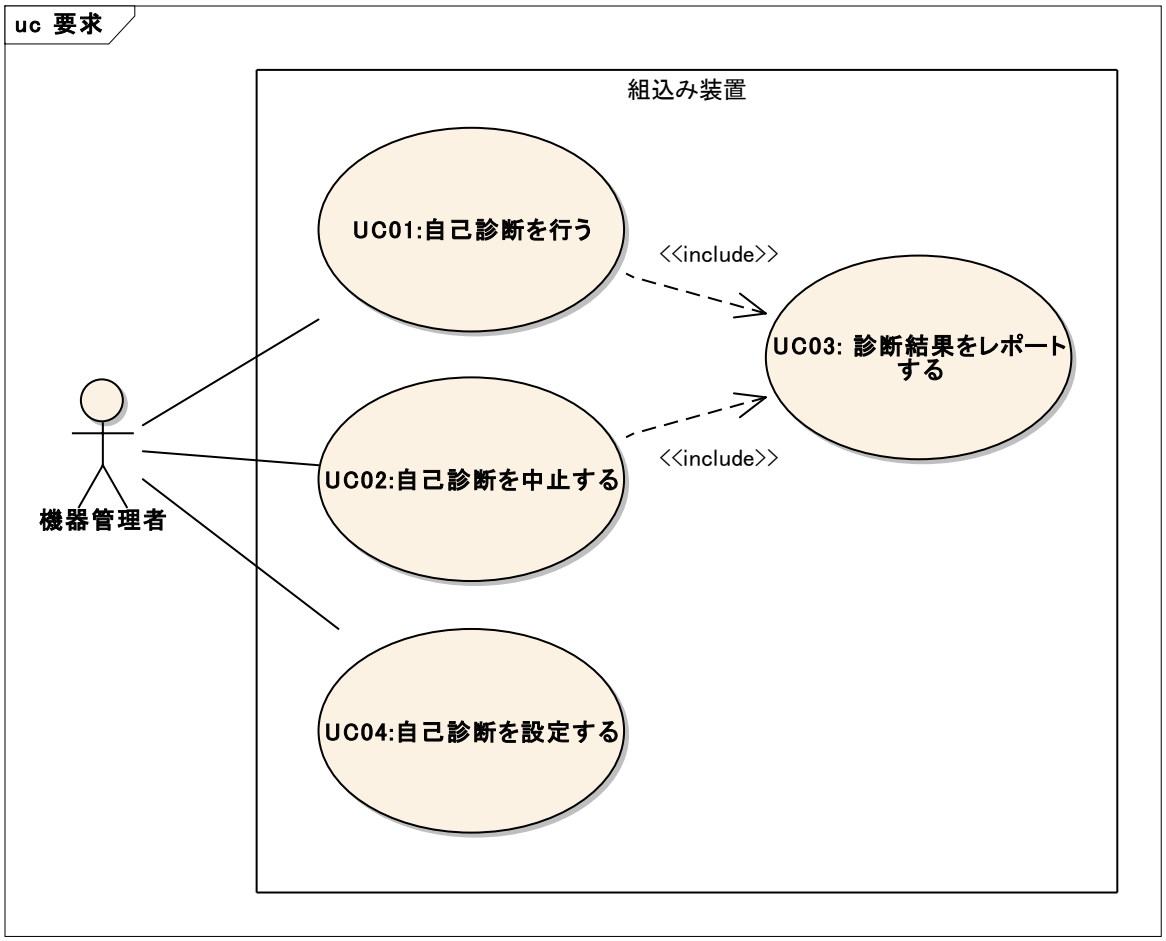
自動車や通信機器などには警告灯を備えるものや故障を示す表示ができるものがあります。これらには、装置が正しく動いているかどうかを調べる自己診断機能があり、異常を検知すると、(対処や)表示を行います。また、トラブル発生時に原因究明や対応を行うため、手動で自己診断ができる機器もあります。



要求仕様

1. 組込み装置には、装置が正しく動作しているかどうかを調べる自己診断機能があります。
2. 自己診断は、機器管理者の要求により起動される場合と、予め設定された条件に合わせて自動的に起動される場合があります。いずれも、機器管理者の指示により中止することが可能です。
3. 診断の対象には、デバイス単体や、複数デバイスから構成される機能ユニット、そして装置上で動作するアプリケーションが含まれます。
   機能ユニットは階層構造を持つことが可能で、上位の機能ユニットは複数の下位機能ユニットを含むことができます。
4. 一つの装置内では複数の自己診断を実施することが可能です。各自己診断は、1 つの診断対象に対する診断でも、それらを任意に組み合わせたものでも構いません。
   （例：起動時診断や月次診断では全機能ユニット、週次診断では重要機能ユニット、日時診断では最重要ユニットなど。）
   また、自己診断に含まれる各診断は、並行・逐次、どちらの形でも実施可能です。
5. 診断対象ごとに、診断項目が決まっており、各診断ではどの診断項目を診断するかを設定することが可能です。
   （例：起動時診断や月次診断では診断項目 1～10 まで、週次診断では 1～5 と 6～10 を交互に、日時診断では 1 と 2 のみなど。）
6. 診断結果は、診断項目ごとに OK/NG の結果と、NG の場合には、エラーコードをレポートします。
7. 診断結果の履歴は保存する必要がありません。
8. 診断途中で中止されたときには、それまでの診断内容をレポートして終了します。

ユースケース図


uc 要求
組込み装置
UC01:自己診断を行う
UC02:自己診断を中止する
UC03: 診断結果をレポートする
UC04:自己診断を設定する
<<include>>
<<include>>
機器管理者

ユースケース記述

<UC01:自己診断を行う>

■概要

機器管理者が組込み装置に診断開始を指示すると、組込み装置は診断を行い、診断結果をレポートする。

■アクター

機器管理者

■事前条件

・組込み装置に実行する自己診断が設定されている

■事後条件

・自己診断が終了し、診断結果がレポートされている

■メインフロー

1. アクターは、実行する自己診断を指定し、システムに診断開始を指示する
2. システムは、診断を行う
3. システムは、「診断結果をレポートする」ユースケースを実行する
4. UC を終了する

■課題や T．B．D 項目

なし

■備考

・複数の自己診断を並行して行うことが可能。この際、組込み装置内で必要な排他制御を行う。

<UC02:自己診断を中止する>

■概要

組込み装置が自己診断実行中に、機器管理者が自己診断中止を指示すると、組込み装置は自己診断を中止し、それまでの診断結果をレポートする。

■アクター

機器管理者

■事前条件

・組込み装置は自己診断を実行中である

■事後条件

・自己診断が中止された

・診断結果がレポートされた

■メインフロー

1. アクターは、システムに、自己診断の中止を指示する
2. システムは、自己診断を中止する
3. システムは、「診断結果をレポートする」ユースケースを実行する
4. UC を終了する

■課題や T．B．D 項目

・複数の自己診断を並行して実行中の場合、中止する診断の指定方法(またはすべてを中止)

■備考

・自己診断を実行していない場合に、本ユースケースを実行しても、何も行われない(エラーにならない)。自己診断中止の指示中に自己診断を終了した場合も同様。

<UC03: 診断結果をレポートする>

■概要

組込み装置が診断結果をレポートする。

■アクター

なし

■事前条件

なし

■事後条件

診断結果がレポートされている

■メインフロー

1. システムは、診断結果をレポートする
2. UC を終了する

■課題や T．B．D 項目

・レポートの出力手段
・レポートのフォーマット(形式)

■備考

・本ユースケースは他のユースケースから起動される

<UC04:自己診断を設定する>

■概要

機器管理者が組込み装置に自己診断を設定する。

■アクター

機器管理者

■事前条件

なし

■事後条件

組込み装置に自己診断が設定されている

■メインフロー

1. アクターは、システムに、自己診断を設定することを指示する
2. アクターは、システムに、各種設定内容を指示する設定する
3. システムは、指示された内容の自己診断を保存する
4. UC を終了する

■代替フロー

なし

■課題や T．B．D 項目

・どの自己診断を並行・逐次で実行するかの指定方法
・レポートの出力手段とレポートのフォーマット(形式) の指定方法

■備考

設定内容は次の通り。

・自己診断の名前

・自己診断の起動方法

・診断の種類（1 つの診断対象に対する診断 or 既存の自己診断を組み合わせた診断）

1 つの診断対象に対する診断を設定した場合は、1 つの診断対象を指定する

既存の自己診断を組み合わせた診断の場合は、複数の自己診断を指定する

シナリオ

<SC0101:自己診断を行う>

1. 山田太郎さんがルーターにイーサネット診断の開始を指示する。
2. ルーターは診断を行う。
3. ルーターは診断結果をレポートする。

<SC0201: 診断を中止する>

1. 山田太郎さんがルーターにイーサネット診断の開始を指示する。
2. ルーターは診断を行う。
3. 山田太郎さんがルーターに診断の中止を指示する。
4. ルーターは自己診断を中止する。
5. ルーターは診断結果をレポートする。

<SC0401: 自己診断を設定する(1 つの診断対象に対する診断)>

1. 山田太郎さんがルーターに自己診断を設定することを指示する。
2. 山田太郎さんはルーターに設定する自己診断の名前を『イーサネットメモリチェック』と指示する。
3. 山田太郎さんはルーターに自己診断の起動方法を毎日午前 1 時と指示する。
4. 山田太郎さんはルーターに「1 つの診断対象に対する診断」を設定することを指示する。
5. 山田太郎さんはルーターに自己診断の診断対象をイーサネットと指示する。
6. 山田太郎さんはルーターに自己診断の診断内容をメモリチェックと指示する。
7. ルーターは毎日午前 1 時、イーサネットを対象に、メモリチェックの自己診断を行うように設定する。

<SC0402: 自己診断を設定する(1 つの診断対象に対する診断その 2)>

1. 山田太郎さんがルーターに自己診断を設定することを指示する。
2. 山田太郎さんはルーターに設定する自己診断の名前を『イーサネットレジスタチェック』と指示する。
3. 山田太郎さんはルーターに自己診断の起動方法を毎日午前 2 時と指示する。
4. 山田太郎さんはルーターに「1 つの診断対象に対する診断」を設定することを指示する。
5. 山田太郎さんはルーターに自己診断の診断対象をイーサネットと指示する。
6. 山田太郎さんはルーターに自己診断の診断内容をレジスタチェックと指示する。
7. ルーターは毎日午前 2 時、イーサネットを対象に、レジスタチェックの自己診断を行うように設定する。

<SC0403: 自己診断を設定する(既存の自己診断を組み合わせた診断)>

1. 山田太郎さんがルーターに自己診断を設定することを指示する。
2. 山田太郎さんはルーターに設定する自己診断の名前を『イーサネット』と指示する。

3. 山田太郎さんはルーターに自己診断の起動方法を毎週日曜日午前 3 時と指示する。
4. 山田太郎さんはルーターに「既存の自己診断を組み合わせた診断」を設定することを指示する。
5. 山田太郎さんはルーターに『イーサネットメモリチェック』自己診断を組み合わせるように指示する。
6. 山田太郎さんはルーターに『イーサネットレジスタチェック』自己診断を組み合わせるように指示する。
7. ルーターは毎週日曜日午前 3 時、『イーサネットメモリチェック』自己診断と『イーサネットレジスタチェック』自己診断を組み合わせた自己診断を行うように設定する。

モデル一覧

モデル名	概要	ポイント
エンティティに着目して分析したモデル	要求仕様の内容をエンティティを中心に分析し、診断と診断項目の静的構造に着目したモデルです。	要求仕様の内容を忠実に分析しています。
メタファを使って分析したモデル	装置の自己診断機能を、日常の診断(健康診断など)にたとえてモデリングしています。	身近な話題から入るので、モデリングの導入編としてお勧めです。

エンティティに着目した分析したモデル

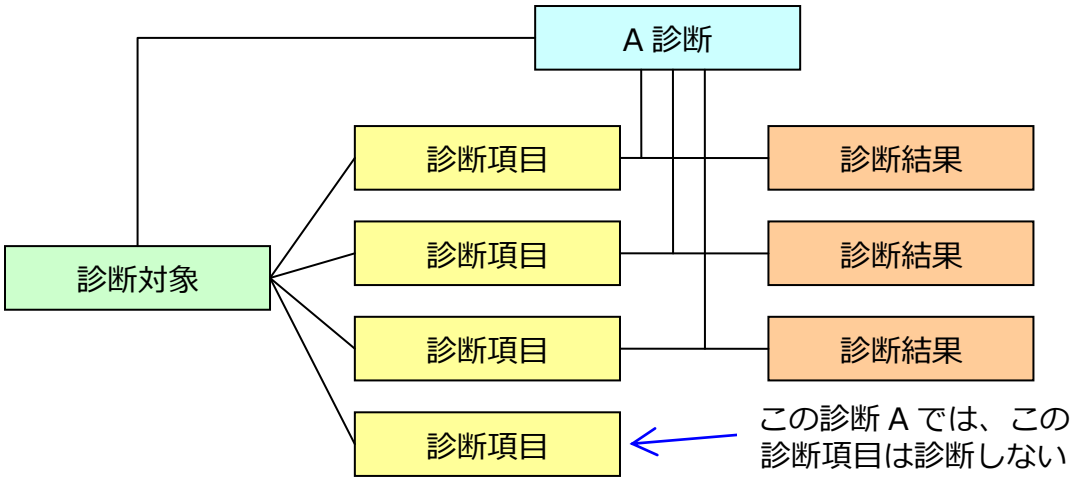
要求仕様をエンティティに着目し、診断を診断対象の診断項目と対応する結果と捉え、このような診断を組み合わせることでさまざまな診断をモデリングしています。

モデリングのコンセプト

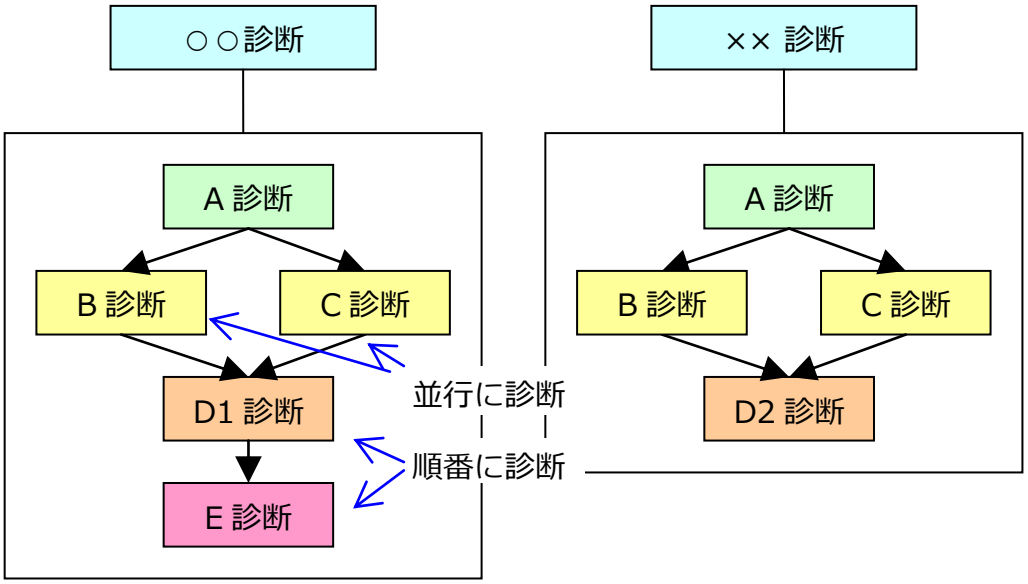
要求仕様の内容を忠実に分析し、モデリングします。

要求仕様から以下の特徴が読み取れました。

(i) (後述の組み合わせた診断に対して) 単一の対象に対する診断とは 1 個の診断対象に対して定められた診断項目を診断し、診断結果を得ることです。診断対象に対して診断項目は決まっていますが、診断毎にどの診断項目を診断するかを設定することが可能です。(UML 図ではなく)簡単に図示すると以下のようになります。



(ii)診断は複数の診断から組み合わせることもでき、以下のように 並行/順番に実行されます(組み合わせた診断)。(UML 図ではなく)簡単に図示すると以下のようになります。



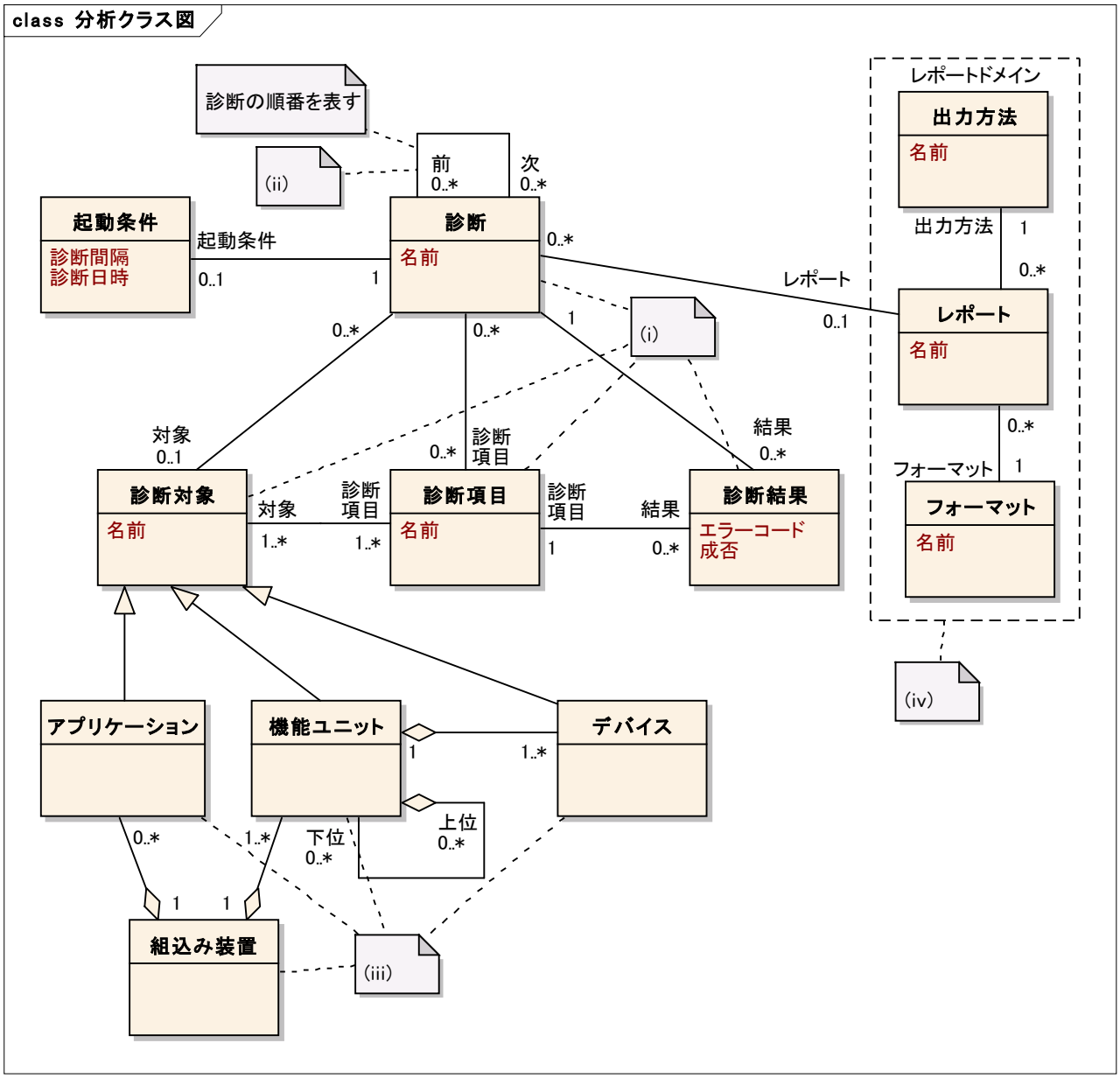
分析モデル

前述の(i)より、診断、診断対象、診断項目、診断結果、の各クラスとそれらの関係、(ii)より診断クラスの自己関連が導き出されます。さらに以下を加え、クラス図を作成しました。

(ⅲ)要求仕様 3 より診断対象の構造(組込み装置、機能ユニット、デバイス、アプリケーション、の各クラス)

(ⅳ)レポートを表すレポートドメイン(レポート、フォーマット、出力方法、の各クラス)

クラス図



診断には名前があり、個々の診断が区別されます。

診断は診断条件、診断対象、診断項目、レポートに関連があり、診断が何時(診断条件)、何を(診断対象)、どの診断項目で、どのようにレポートするかを表します。

診断条件、診断対象、診断項目、レポートは関連がない場合もあります。組み合わせた診断の場合、個々の診断には必要ないので、関連がない場合もあります。また、機器管理者の要求により起動される場合には、診断条件がない場合もあります。

診断結果は診断と診断項目に関連があります。どの診断のどの診断項目の診断結果なのかを表します。診断を行っていない場合、診断結果はありません。

組み合わせた診断は診断の自己関連で表されます。実施する診断を順番に繋ぎます。この際に 1 個の診断から複数の診断へ分岐すると、それらの複数の診断が並列に実施されることを表します。また、複数の診断から 1 個の診断に合流すると、並列の実施から順次の実施となります。

組込み装置は機能ユニットから構成され、機能ユニットはデバイスから構成されます。機能ユニットは構造をとることも可能で、ある機能ユニットは機能ユニットから構成されることもあります。また、アプリケーションも組込み装置の構成要素です。

デバイス、機能ユニット、アプリケーションは診断対象です。すなわち、これらを汎化したものが診断対象です。

レポートは、出力方法とフォーマット(形式)があります。

出力方法の例としてはコンソール出力、(ログ)ファイル、電子メール、Web ページ、LED などがあります。

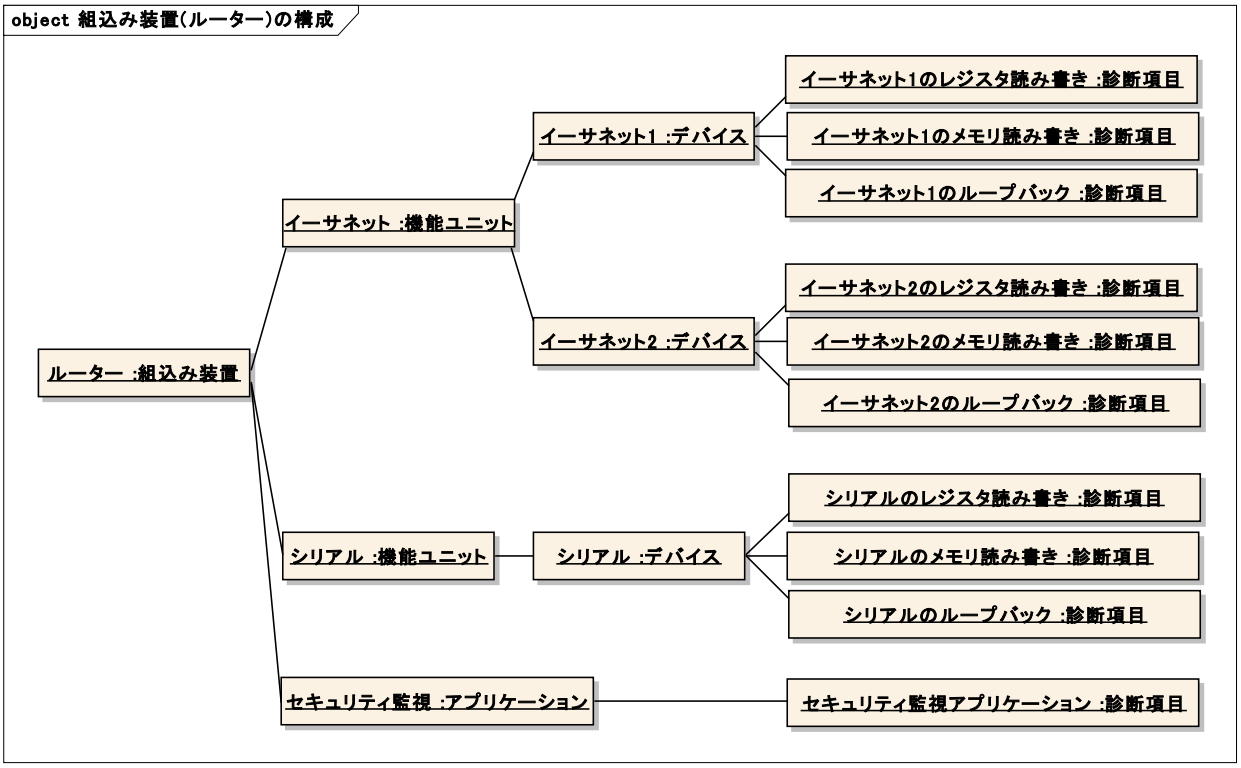
フォーマットの例としてはプレーンテキスト、MIME、HTML などがあります。

レポート、出力方法、フォーマットは意味的にまとまっているので、レポートドメインとします。

オブジェクト図

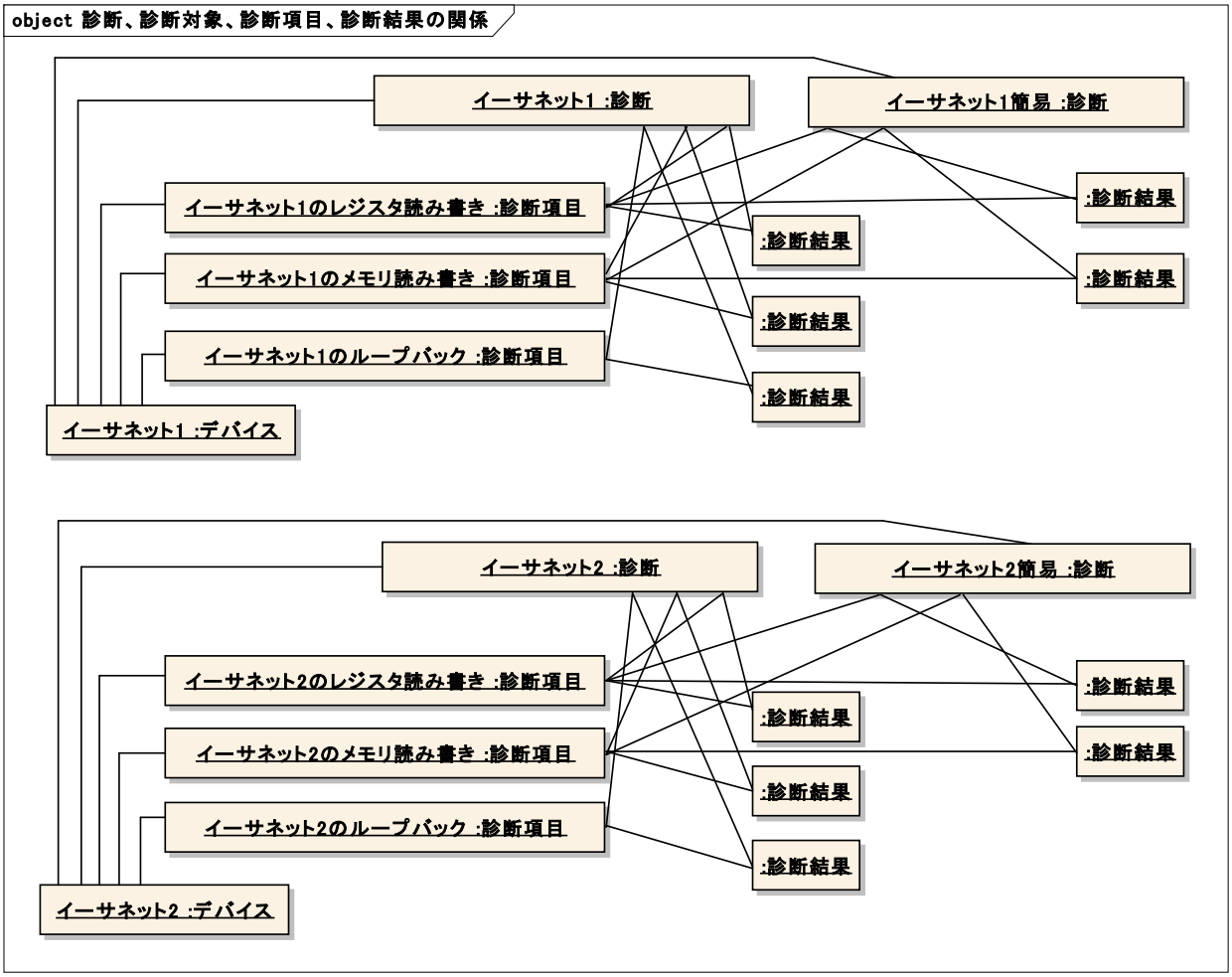
組込み装置(ルーター)のオブジェクト図

イーサネット×2、シリアル×1 のルーターのオブジェクト図を示します。



「診断、診断対象、診断項目、診断結果の関係」オブジェクト図

先のルーターで診断と診断に関連するモノを示したオブジェクト図です。

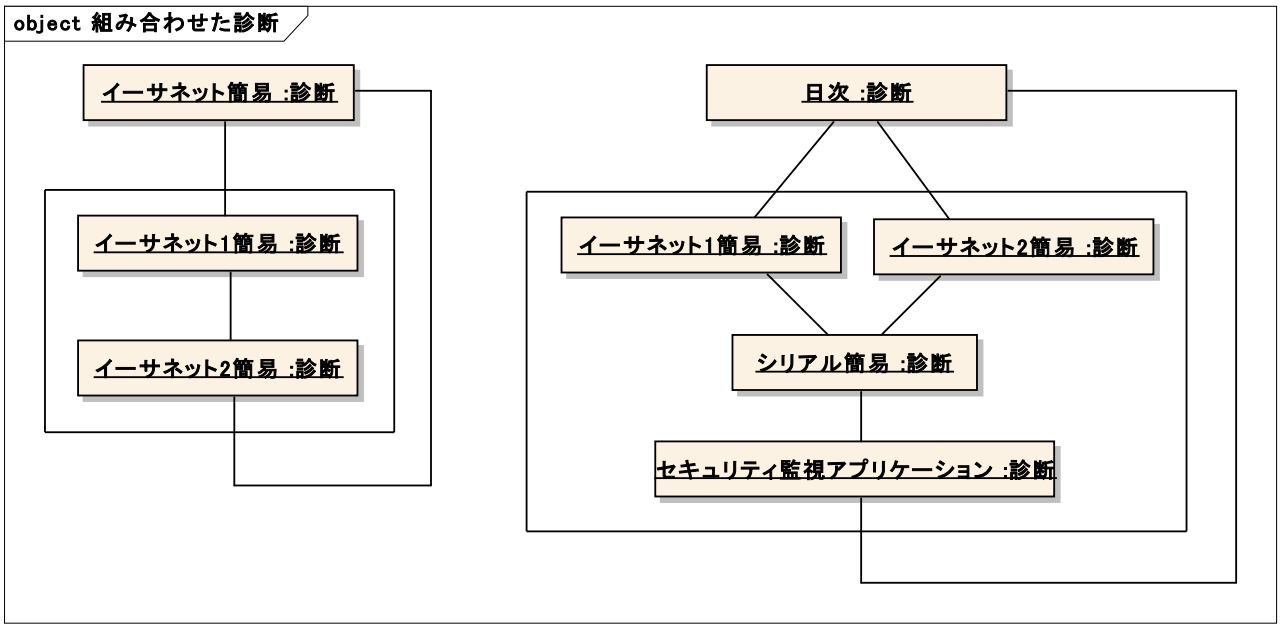


「イーサネット:デバイス」は「レジスタ読み書き:診断項目」、「メモリ読み書き:診断項目」、「ループバック:診断項目」と関連があり、デバイスのインスタンスが異なると、同じ診断項目でも別の診断項目のインスタンスとなります。

ある診断対象(「イーサネット:デバイス」)に対する診断は、診断対象のすべての診断項目を行ってもよいですし(「イーサネット 1: 診断」や「イーサネット 2: 診断」)、一部の診断項目を行ってもよいです(「イーサネット 1 簡易: 診断」や「イーサネット 2 簡易: 診断」)。

「組み合わせた診断」オブジェクト図

同じく先のルーターで組み合わせた診断のオブジェクト図です。



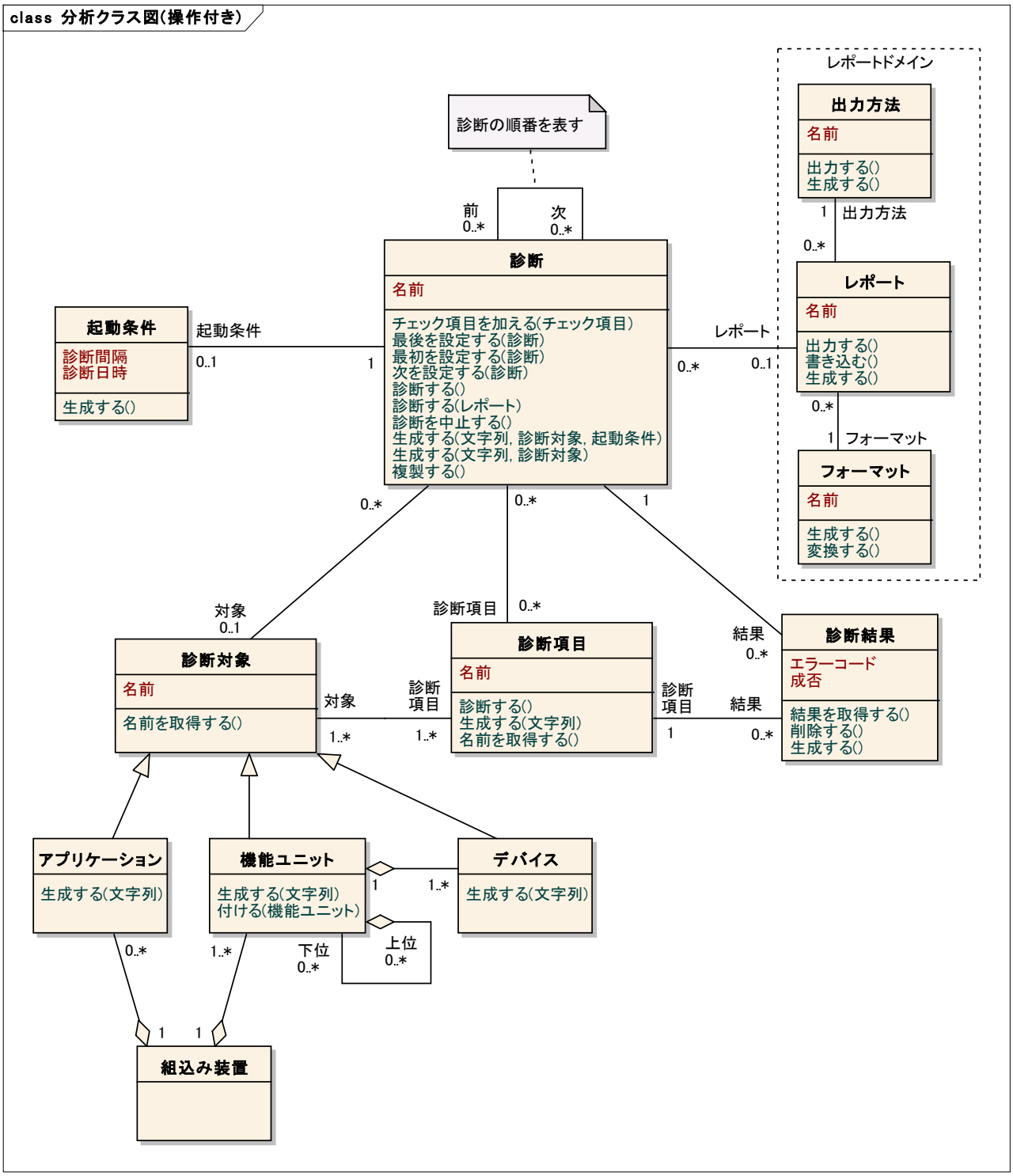
「イーサネット簡易:診断」は「イーサネット 1 簡易:診断」を実施し、次に「イーサネット 2 簡易:診断」を実施します。

「日次:診断」は「イーサネット 1 簡易:診断」と「イーサネット 2 簡易:診断」を並列に実施し、次に「シリアル簡易:診断」を実施し、次に「セキュリティ監視アプリケーション:診断」を実施します。

「イーサネット 2 簡易:診断」から「イーサネット簡易:診断」へのリンク、「セキュリティ監視アプリケーション:診断」から「日次:診断」へのリンクは診断の終了を通知するためにあります。

クラス図(操作付き)

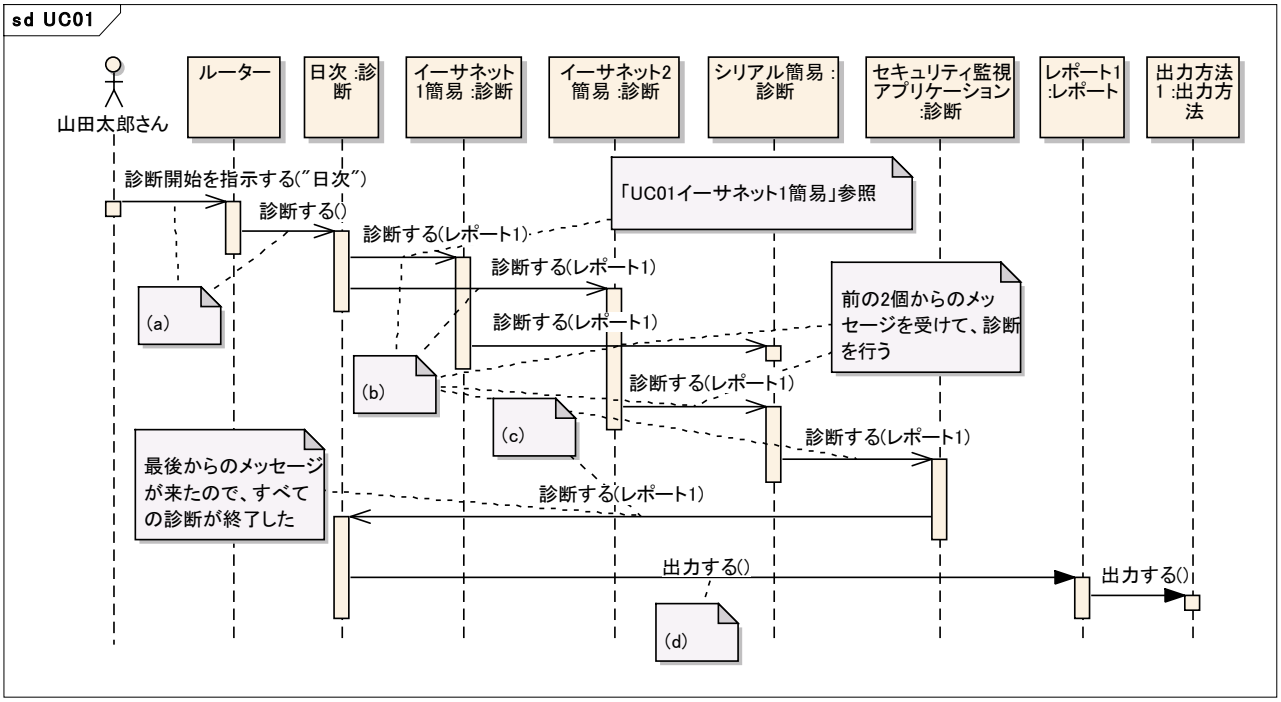
後述のシーケンス図を作成し、操作の検討を行ったクラス図です。



シーケンス図

操作の検討と実現性の確認のため、ユースケースのメインフローまたはシナリオからシーケンス図を作成しました。オブジェクト図と同様にルーターを題材としています。

ユースケース「UC01:自己診断を行う」のシーケンス図

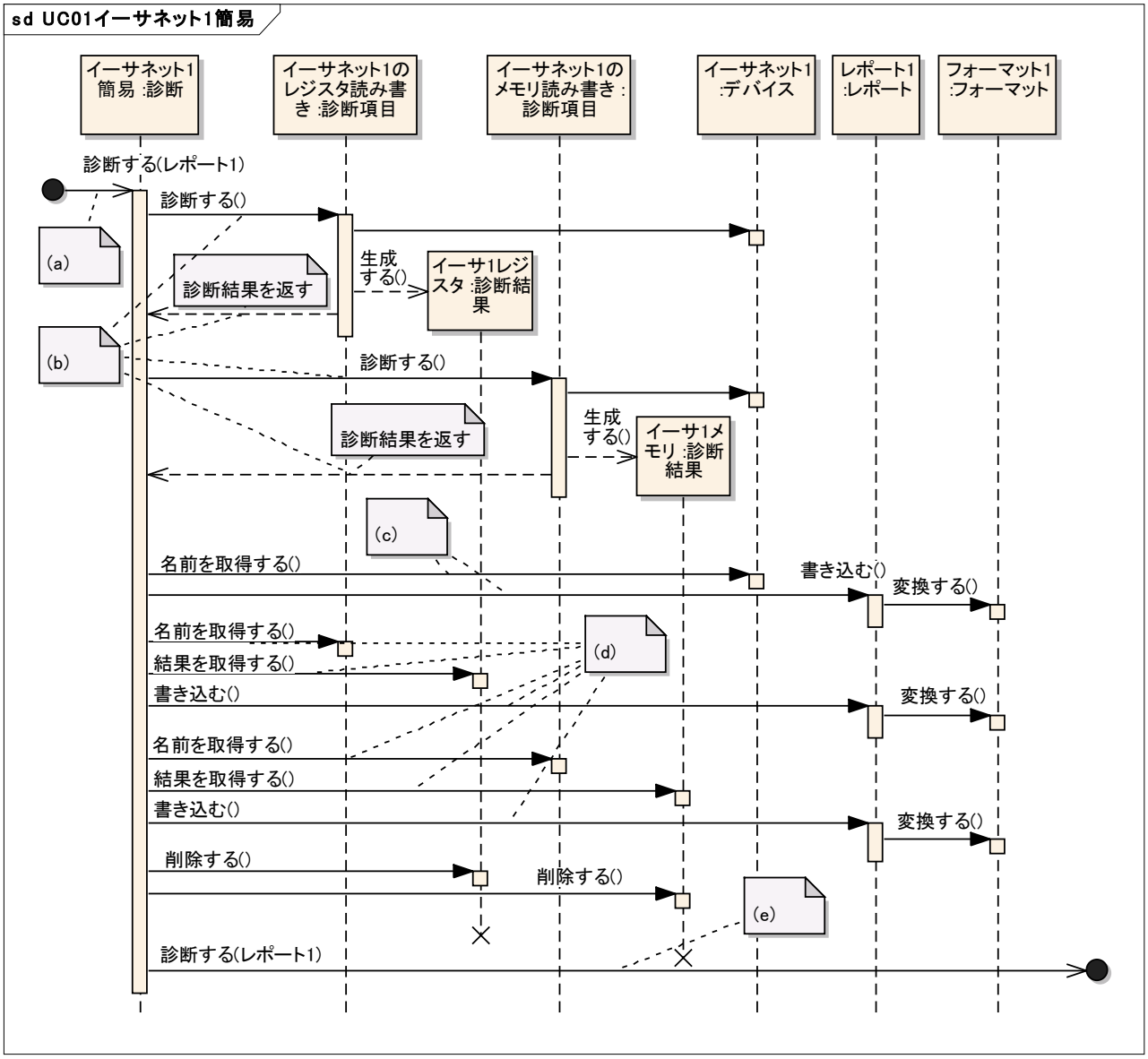


(a)アクターから診断開始の 指示を受けると、ルーターは該当する診断インスタンスに「診断する」メッセージを送信します。

(b)「日次:診断」から診断の構造に従って、個々の診断へ「診断する」メッセージを送信します(「日次:診断」から「イーサネット 1 簡易:診断」への「診断する」メッセージなど)。

(c)診断の構造の最後にある「セキュリティ監視アプリケーション:診断」が完了を示す「診断する」メッセージを「日次:診断」へ送信します。

(d)「日次:診断」がレポートに出力します。



(a)「イーサネット 1 簡易:診断」は「診断する」メッセージを受信します。

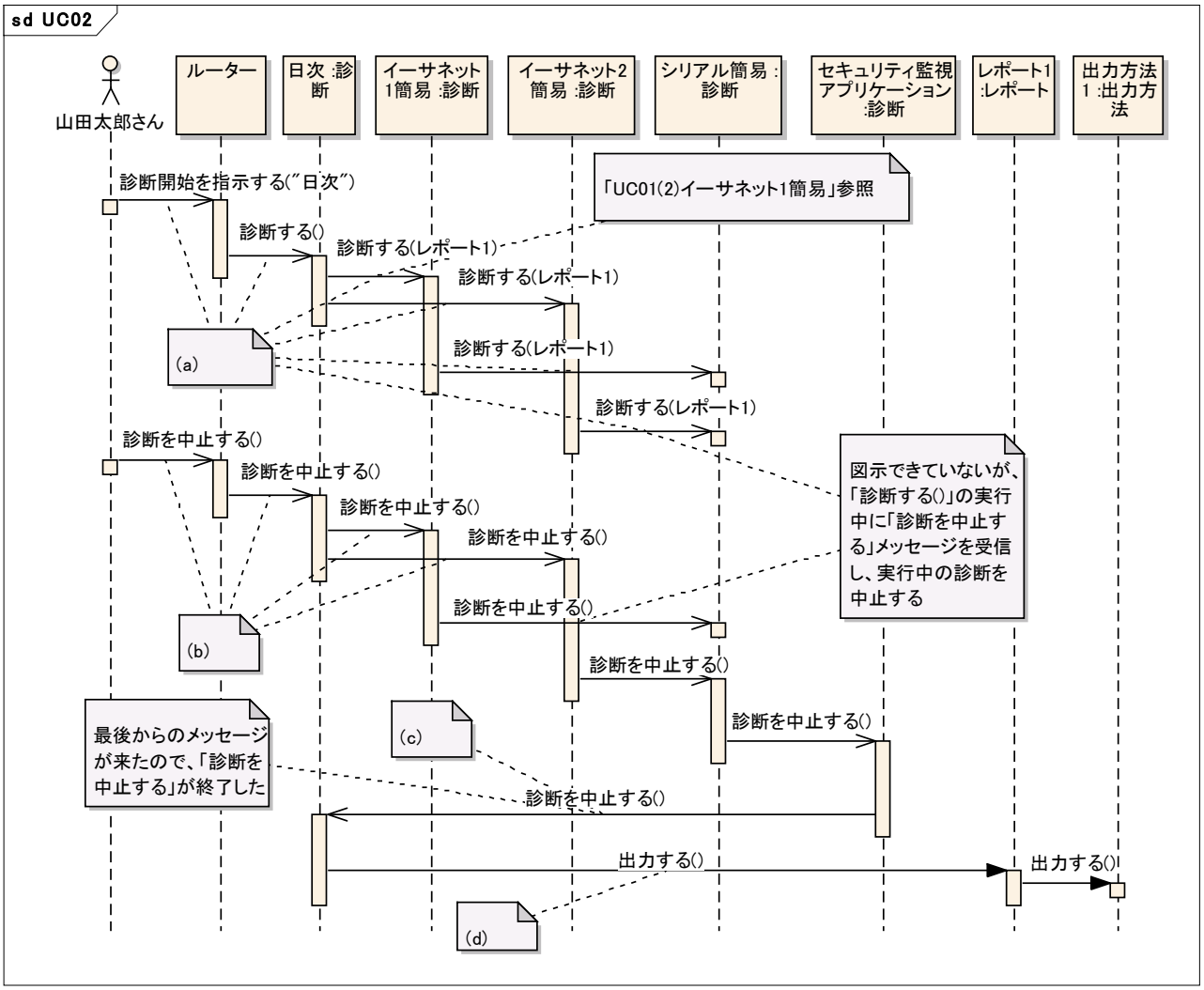
(b)「イーサネット 1 簡易:診断」は各診断項目へ「診断する」メッセージを送信し、診断結果を受け取ります。

(c)「イーサネット 1 簡易:診断」はデバイスから名前を取得し、レポートに書き込みます。

(d)「イーサネット 1 簡易:診断」は各診断結果から結果を取得し、レポートに書き込みます。

(e)(すべて終了したので) 「イーサネット 1 簡易:診断」は「次」の(関連がある)「診断」へ「診断する」メッセージを送信します。

ユースケース「UC02:自己診断を中止する」のシーケンス図



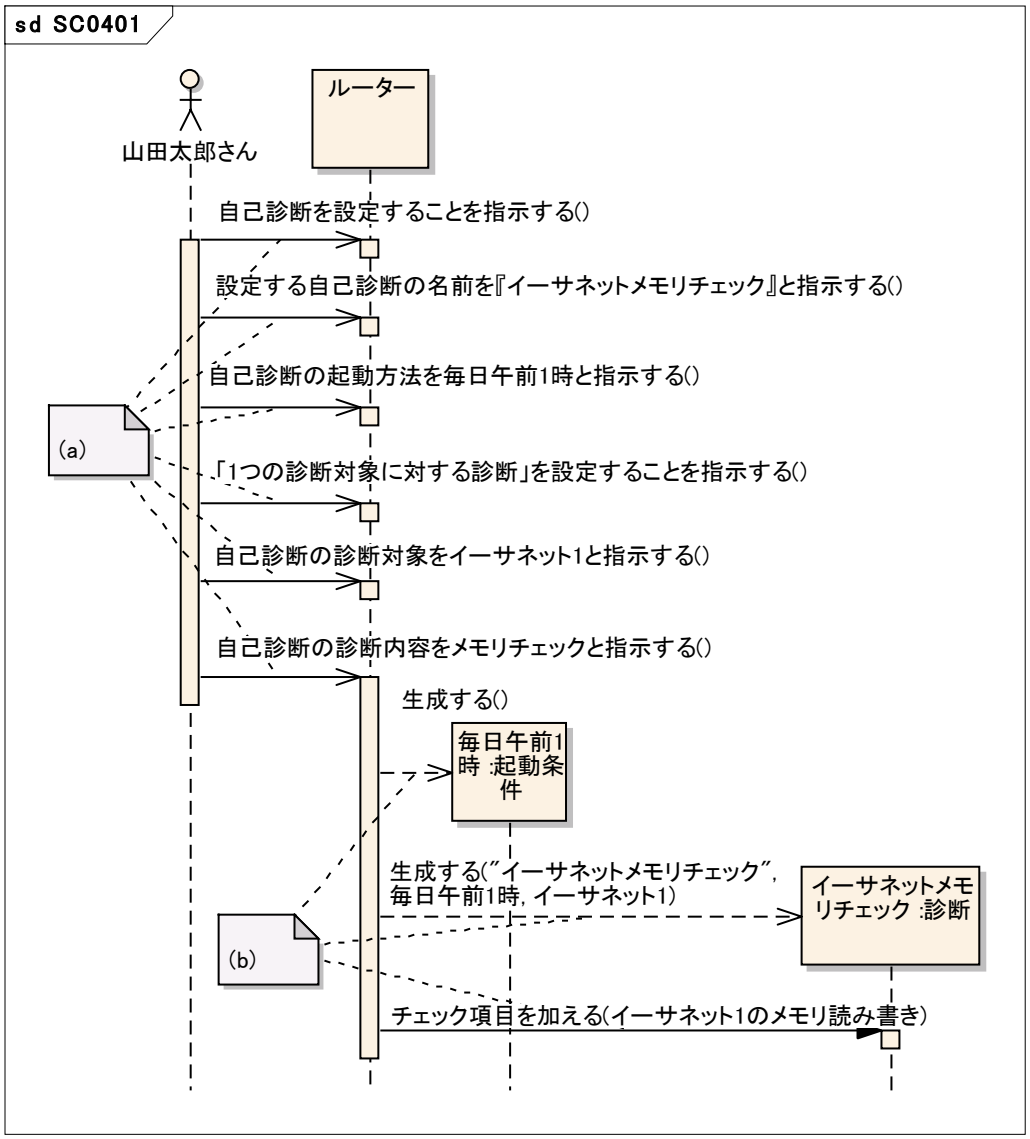
(a)アクターから診断開始の 指示を受けると、ルーターは「ユースケース「UC01:自己診断を行う」のシーケンス図」と同様に動きます。

(b)アクターから診断の中止の指示を受けると、「日次:診断」から診断の構造に従って、個々の診断へ「診断を中止する」メッセージを送信します(「日次:診断」から「イーサネット 1 簡易:診断」への「診断を中止する」メッセージなど)。

(c)診断の構造の最後にある「セキュリティ監視アプリケーション:診断」が完了を示す「診断を中止する」メッセージを「日次:診断」へ送信します。

(d)「日次:診断」がレポートに出力します。

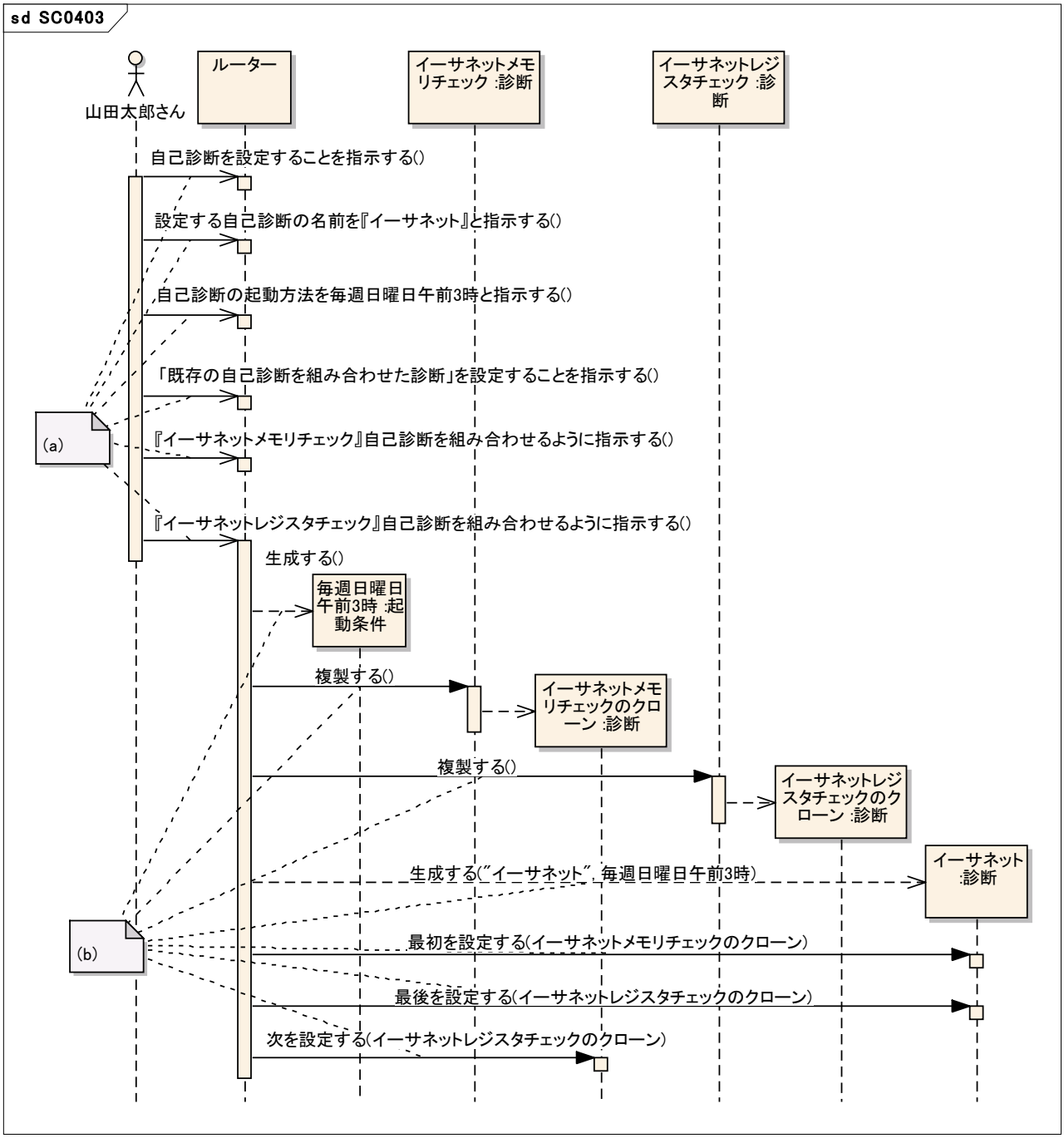
シナリオ「SC0401: 自己診断を設定する(1 つの診断対象に対する診断)」のシーケンス図



(a)アクターはルーターへ自己診断の名前、起動方法、診断対象、診断内容を指示します。

(b)ルーターは起動条件と診断を生成し、関連付けます。

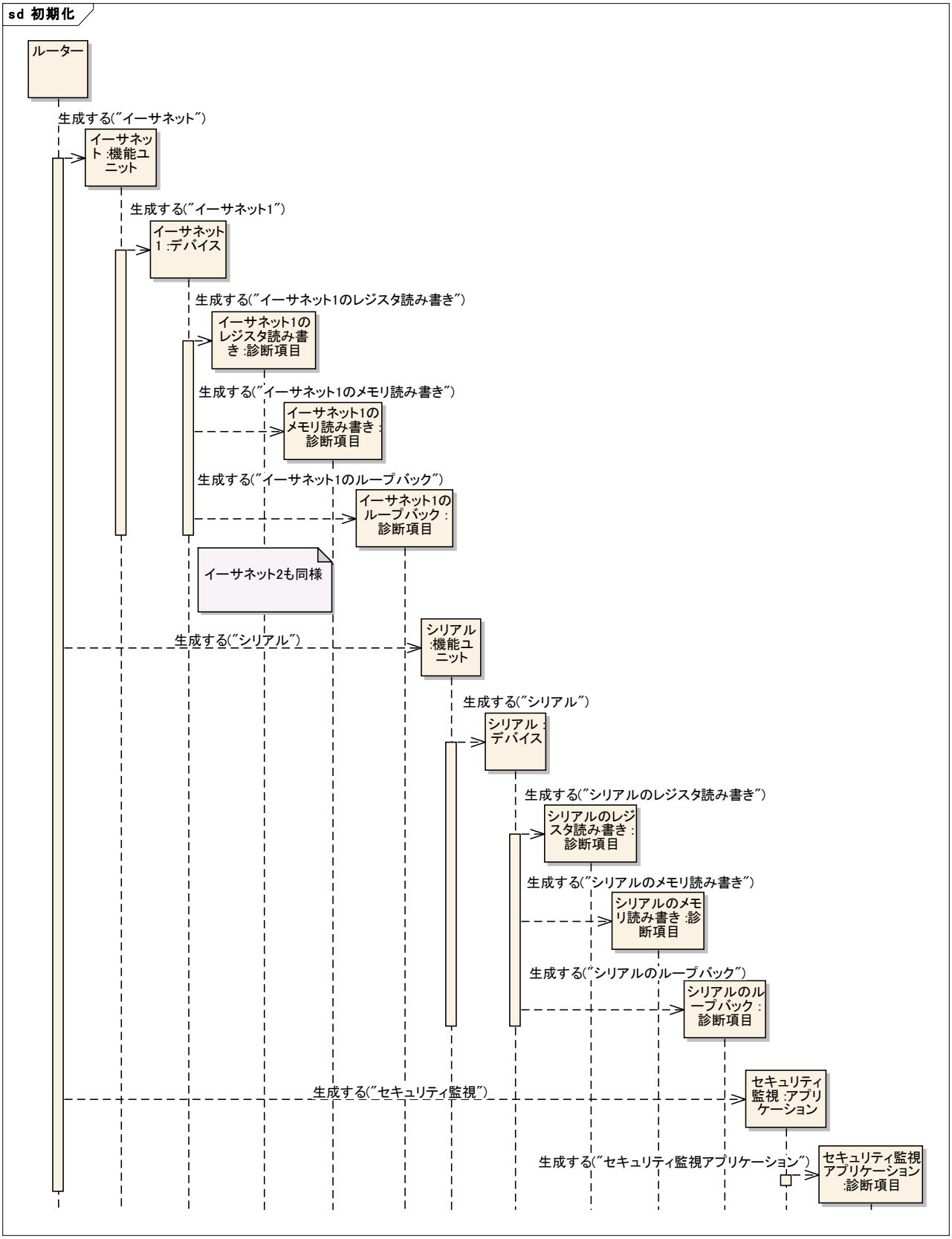
シナリオ「SC0403: 自己診断を設定する(既存の自己診断を組み合わせた診断)」のシーケンス図



(a)アクターはルーターへ自己診断の名前、起動方法、組み合わせる診断を指示します。

(b)ルーターは起動条件と診断を生成し、診断を複製し、関連付けます。

初期化のシーケンス図



ユースケースにはありませんが、初期化の際のシーケンス図です。

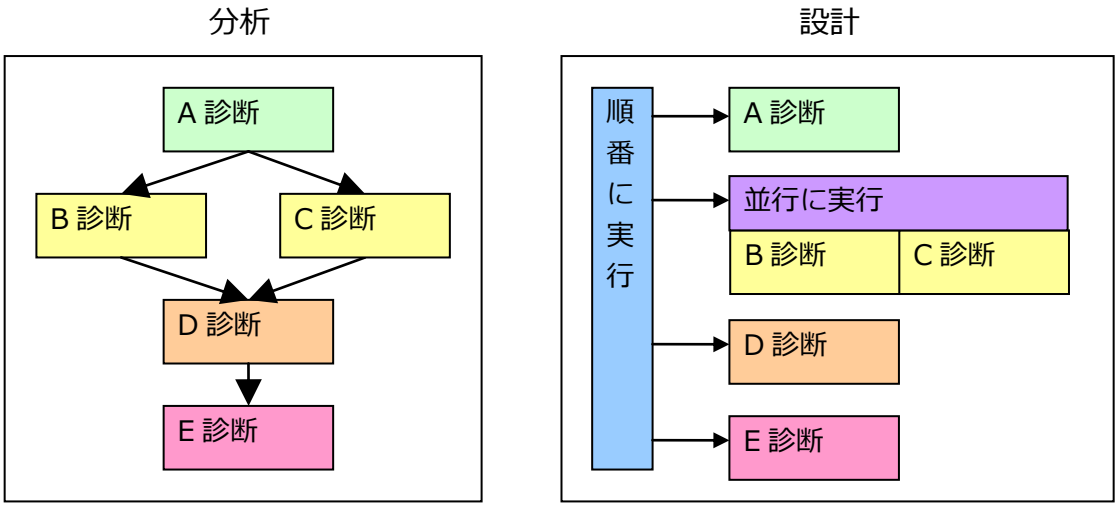
ルーター(組込み装置)は機能ユニットとアプリケーションを生成し、機能ユニットはデバイスを生成し、デバイスとアプリケーションは診断項目を生成します。

PIM 設計モデル

モデリングのコンセプト

実装できるように、かつプラットフォームに依存しないように、分析クラスを詳細化します。具体的には以下の通りです。

(ⅰ)分析では診断の順番は診断インスタンスのネットワークで表されていました。このように実装すると、流れが分岐し、再度合流する処理や、終了を通知する処理の実現が煩雑になりそうです。このため、設計では、診断を順番に行うものと並行して行うものを導入します。



(ⅱ)並列に実行するために、スレッドを導入します。ただし、プラットフォームに依存しないように、汎用的なライブラリとして導入し、PSM で実装することにします。

(ⅲ)スレッドを利用して、診断を実行する「診断実行」クラスを導入します。

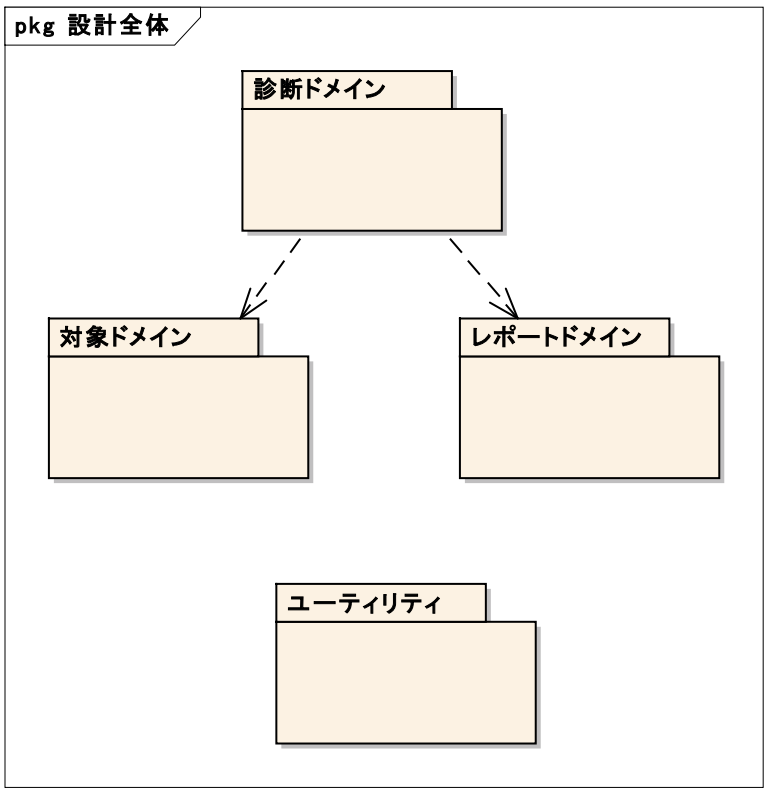
(ⅳ)自己診断を中止するため、実行する/した各診断にユニークな ID を付けます。これにより、自己診断を中止している最中に自己診断を完了してしまうなどのきわどいタイミングでも動作が不安定にならないようにします。

(ⅴ)各インスタンスを管理する「～管理」クラスを導入します。

(ⅵ)誘導可能性の明示など、設計モデルとして詳細化します。

クラス図

設計ではクラスが非常に増えたため、3 個のドメインとユーティリティに分けました。



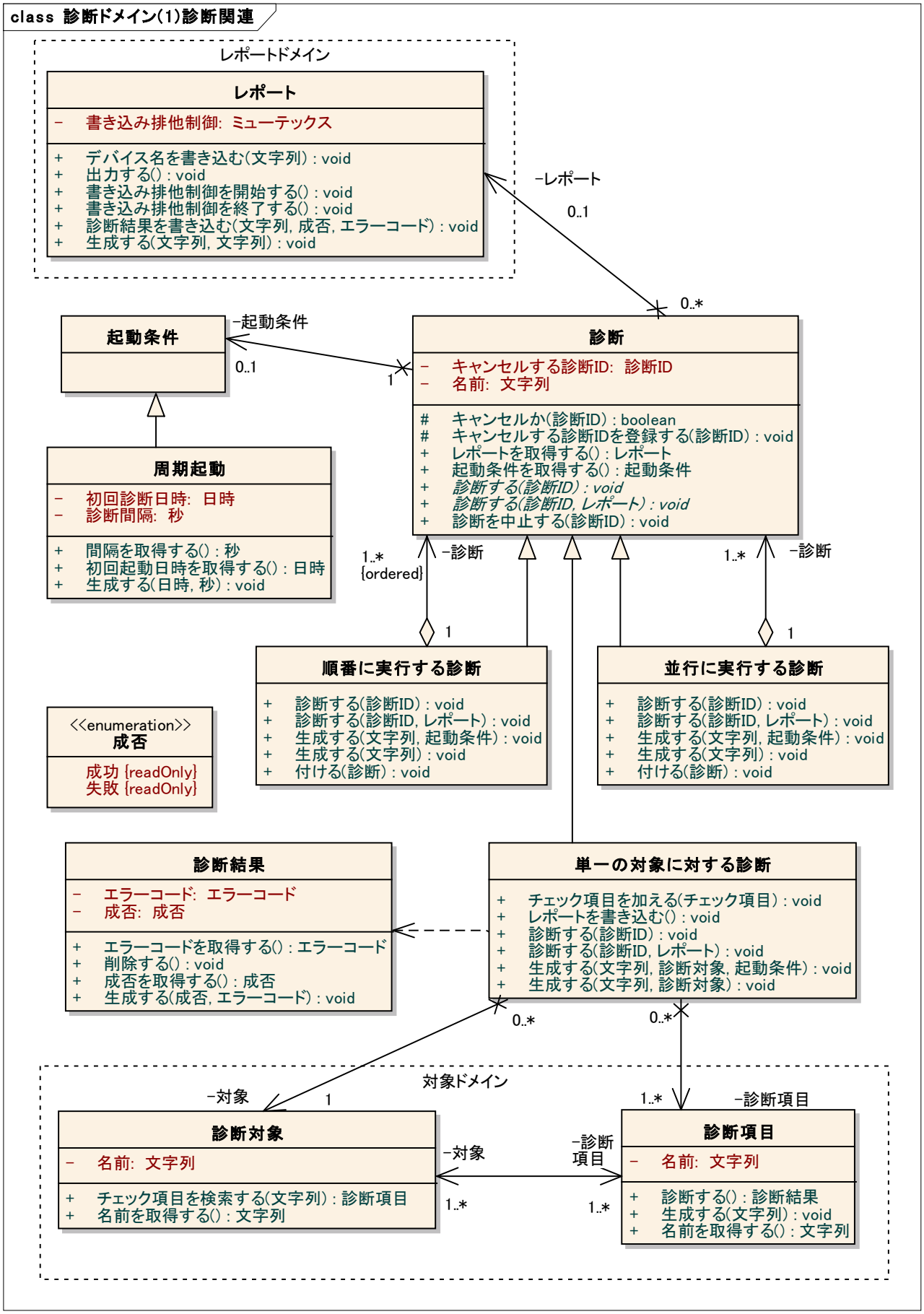
診断ドメイン：自己診断の中心となる「診断」クラスを中心とする部分

対象ドメイン：「診断対象」クラスを中心とする部分

レポートドメイン：「レポート」クラスを中心とする部分

ユーティリティ：スレッドなどを実現する汎用的なライブラリ

「診断ドメイン (1)診断関連」クラス図



スレッドを作成する/しないが異なるため、診断クラスを「順番に実行する診断」クラスと「並行に実行する診断」クラスに分けました。また、「単一の対象に対する診断」も振る舞いが異なるので、異なるクラスとしました。

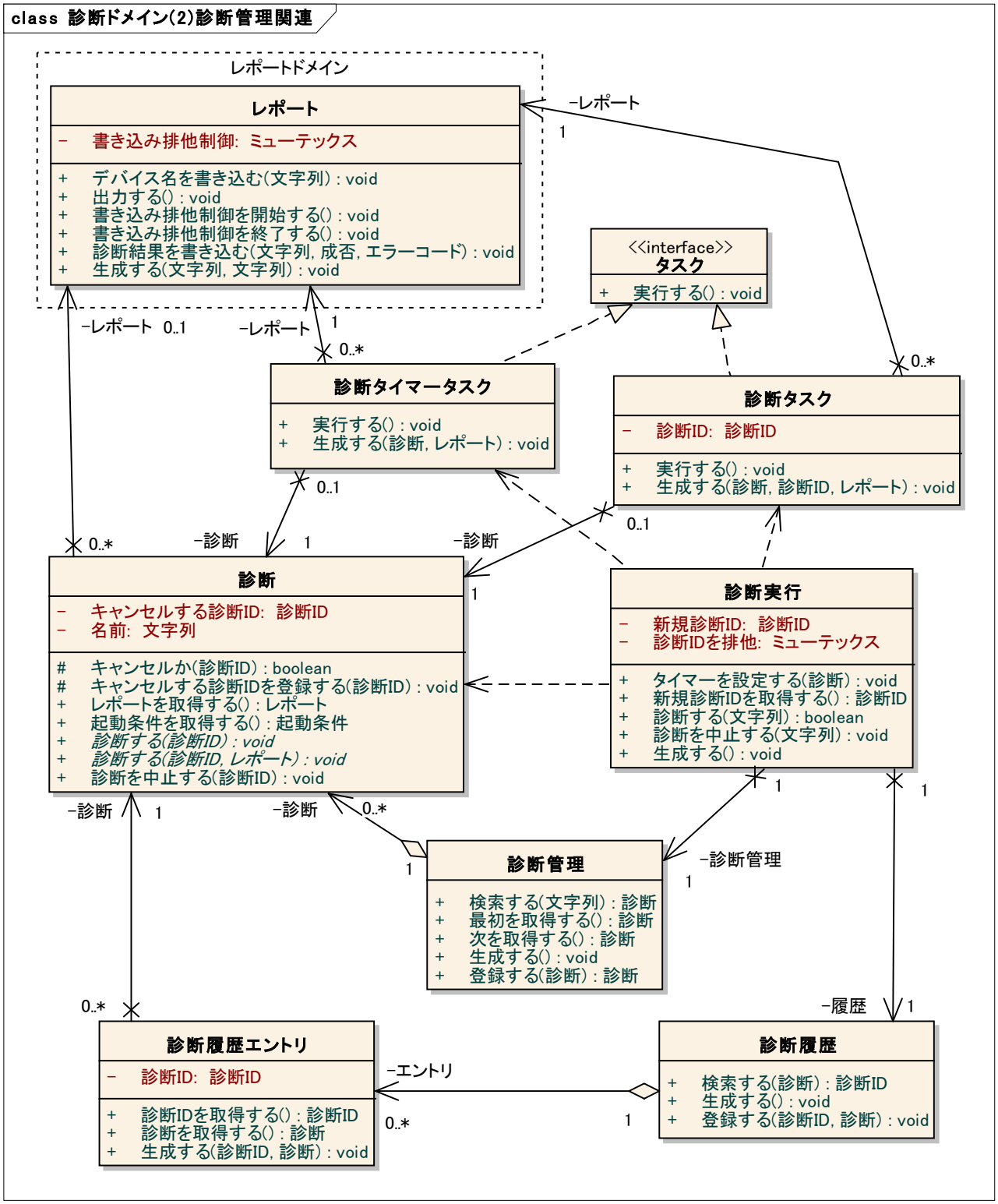
「単一の対象に対する診断」は文字通り単一の対象に対して、指定された診断項目の診断を実行します。

「並行に実行する診断」はスレッドを作成し、指定された診断を並行に実行します。

「順番に実行する診断」はスレッドを作成せず、指定された診断を順番に実行します。

「診断結果」と「診断」の関連は「診断する」操作の間だけ必要なので、依存としました。

「診断ドメイン (2)診断管理関連」クラス図

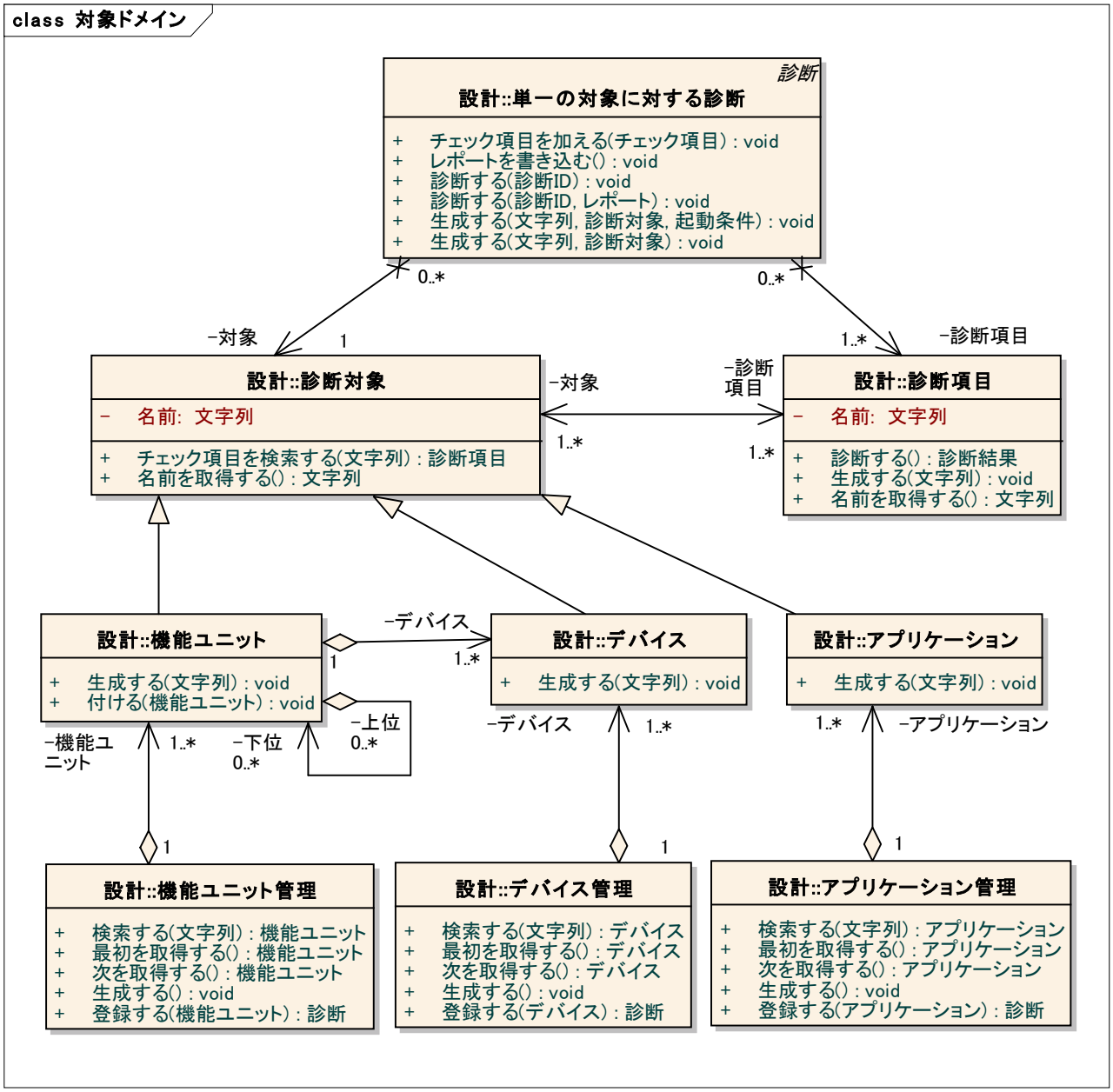


診断管理は診断を集約し、管理します。

診断実行は診断の実行を司ります。タスクインタフェースを実装した診断タスクと診断タイマータスクを利用し、診断を実行します。

診断履歴は診断を行う際に診断履歴エントリを作成し、診断の履歴を管理します。履歴は自己診断を中止する際に使用します。

「対象ドメイン」クラス図

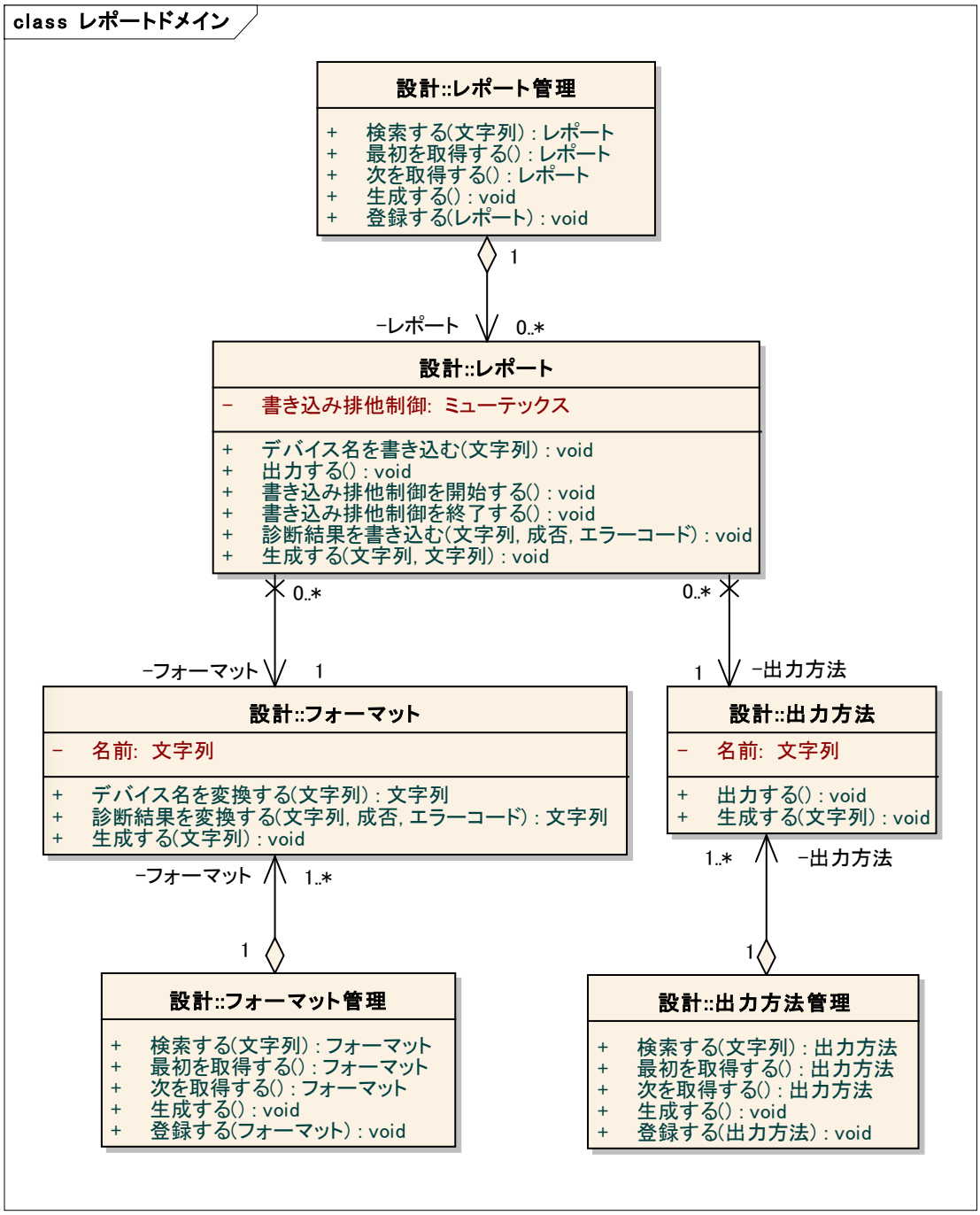


機能ユニット管理は機能ユニットを集約し、管理します。

デバイス管理はデバイスを集約し、管理します。

アプリケーション管理はアプリケーションを集約し、管理します。

「レポートドメイン」クラス図

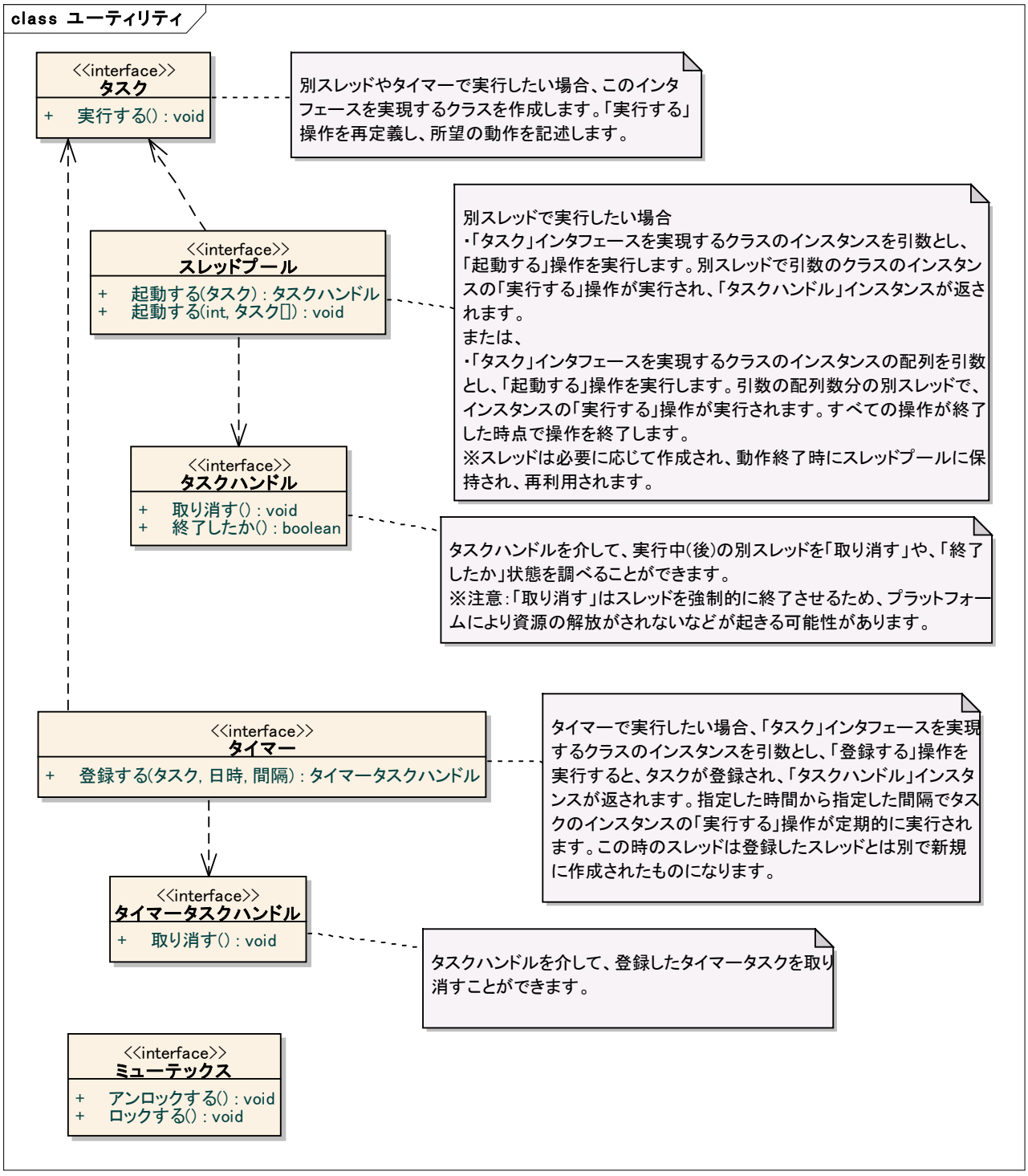


レポート管理はレポートを集約し、管理します。

フォーマット管理はフォーマットを集約し、管理します。

出力方法管理は出力方法を集約し、管理します。

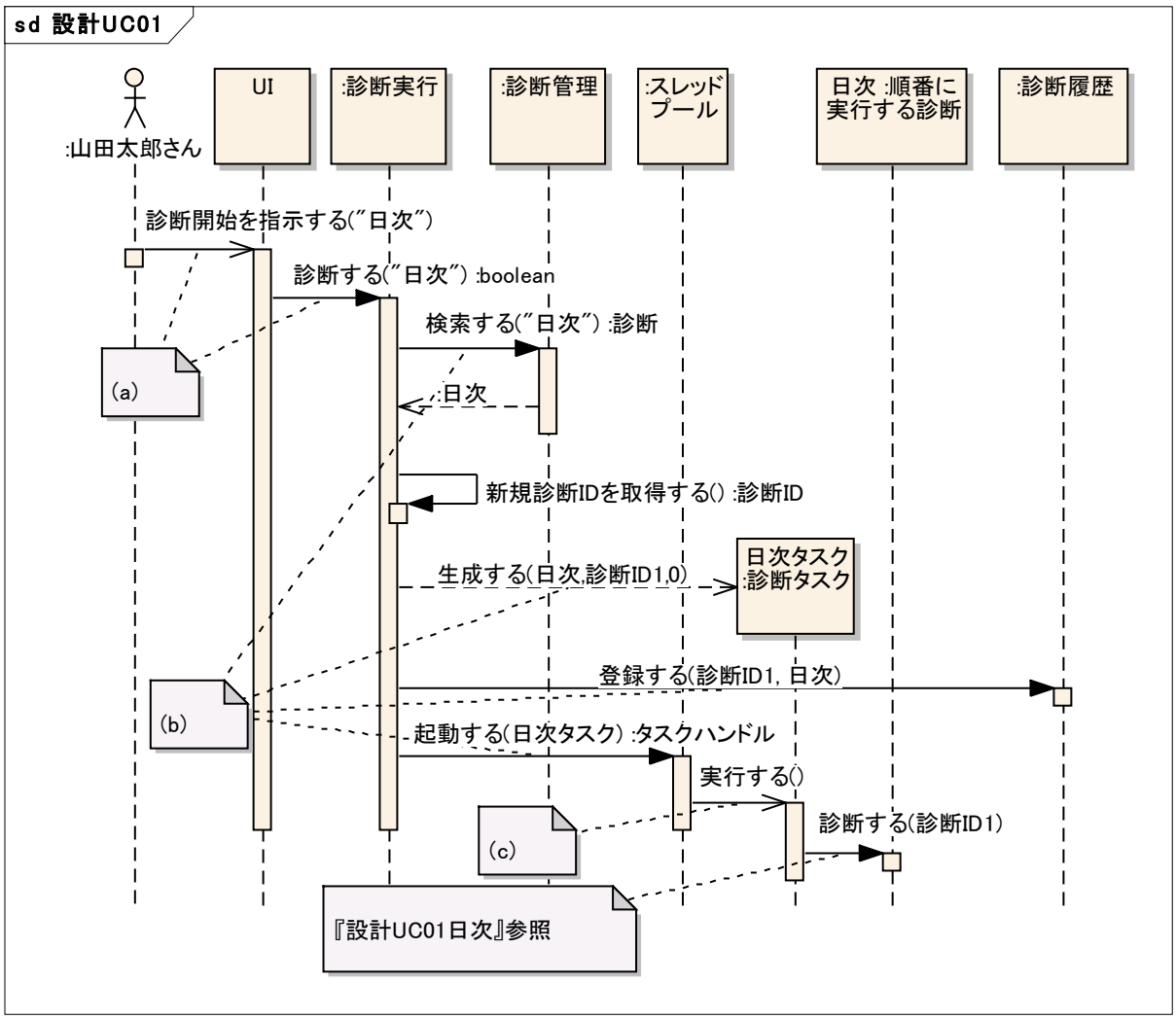
「ユーティリティ」クラス図



シーケンス図

PIM 設計モデルのクラス図を踏まえ、再度シーケンス図を作成します。

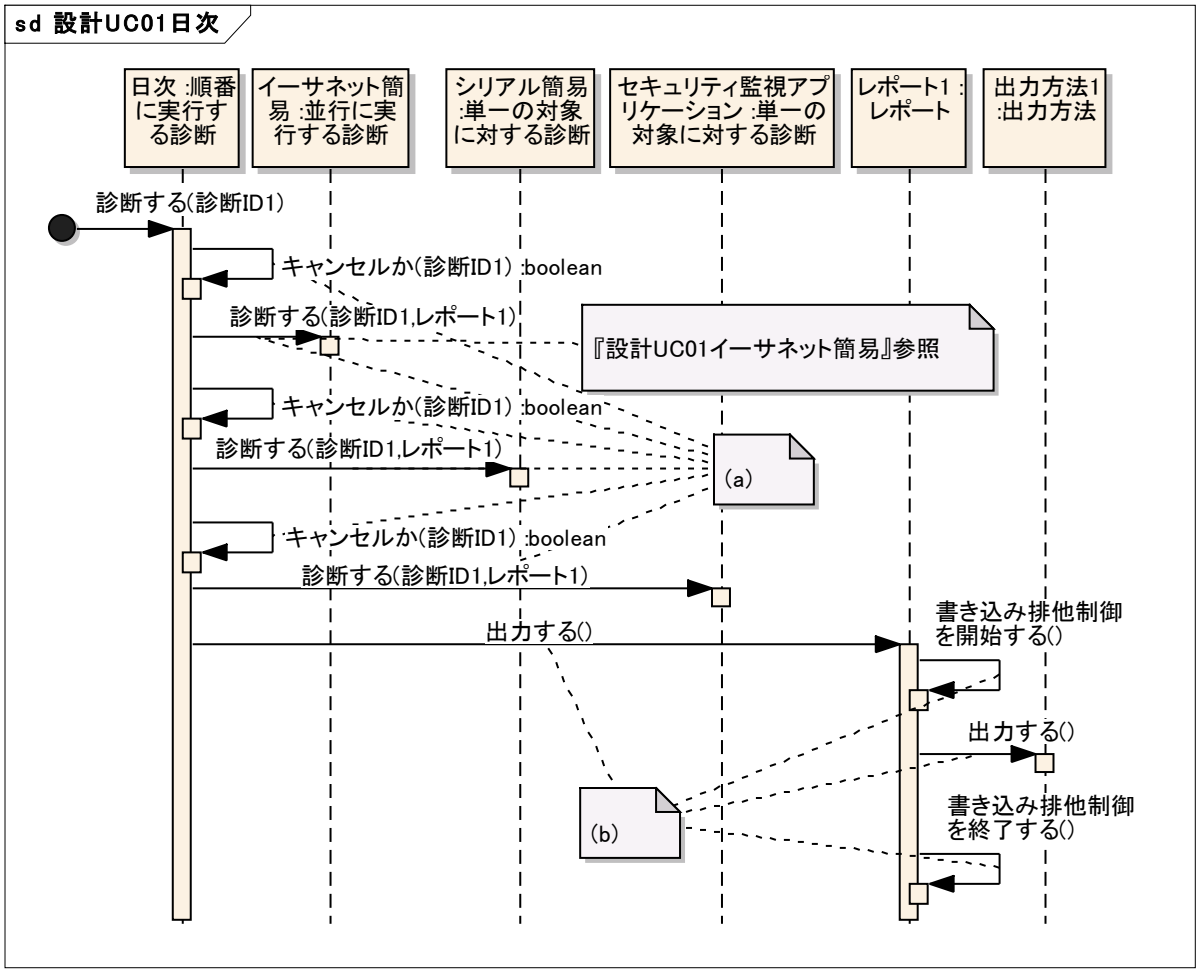
ユースケース「UC01:自己診断を行う」のシーケンス図



(a)アクターから診断開始の 指示を受けると、UI は診断実行に「診断する」メッセージを送信します。

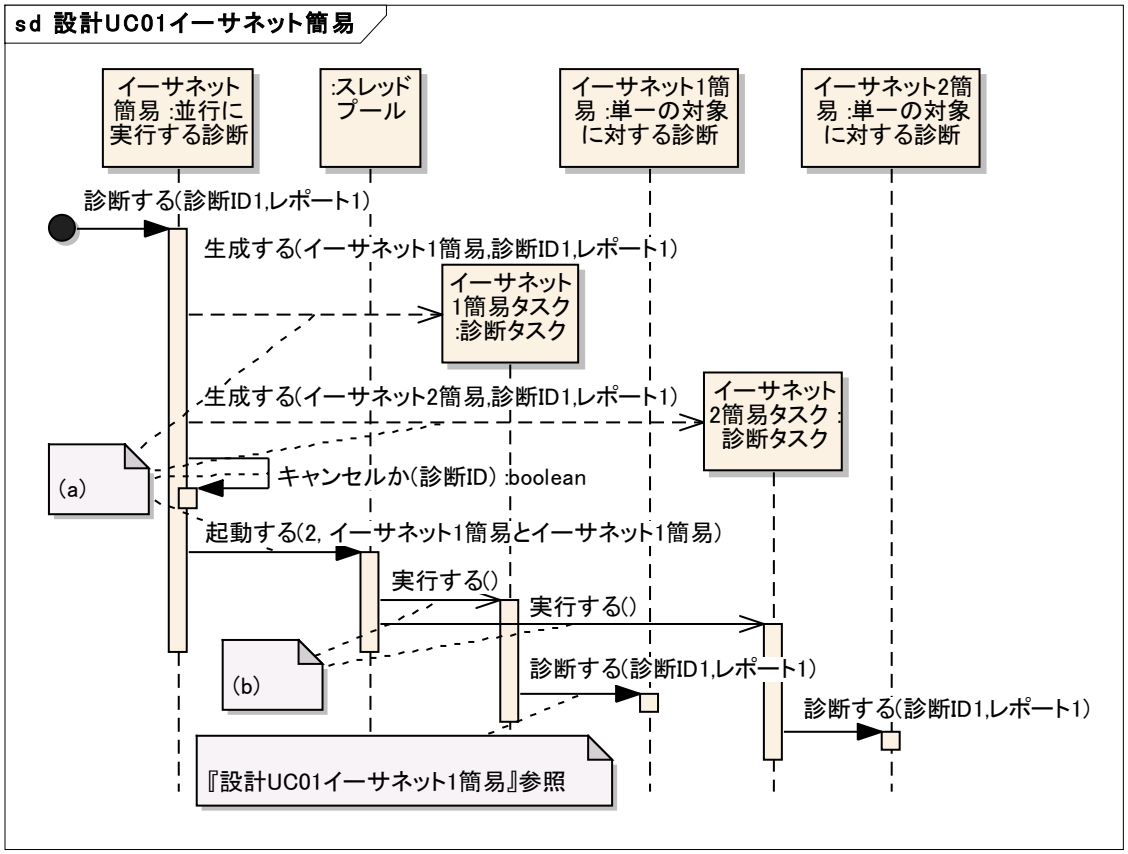
(b)診断実行は診断管理から該当する診断インスタンス(「日次: 順番に実行する診断」)を検索し、診断タスクを生成し、診断履歴に登録し、生成した診断タスクをスレッドプールへ渡して診断を起動します。

(c)スレッドプールは診断タスクへ「実行する」メッセージを送信し、「日次: 順番に実行する診断」に診断を実行させます。



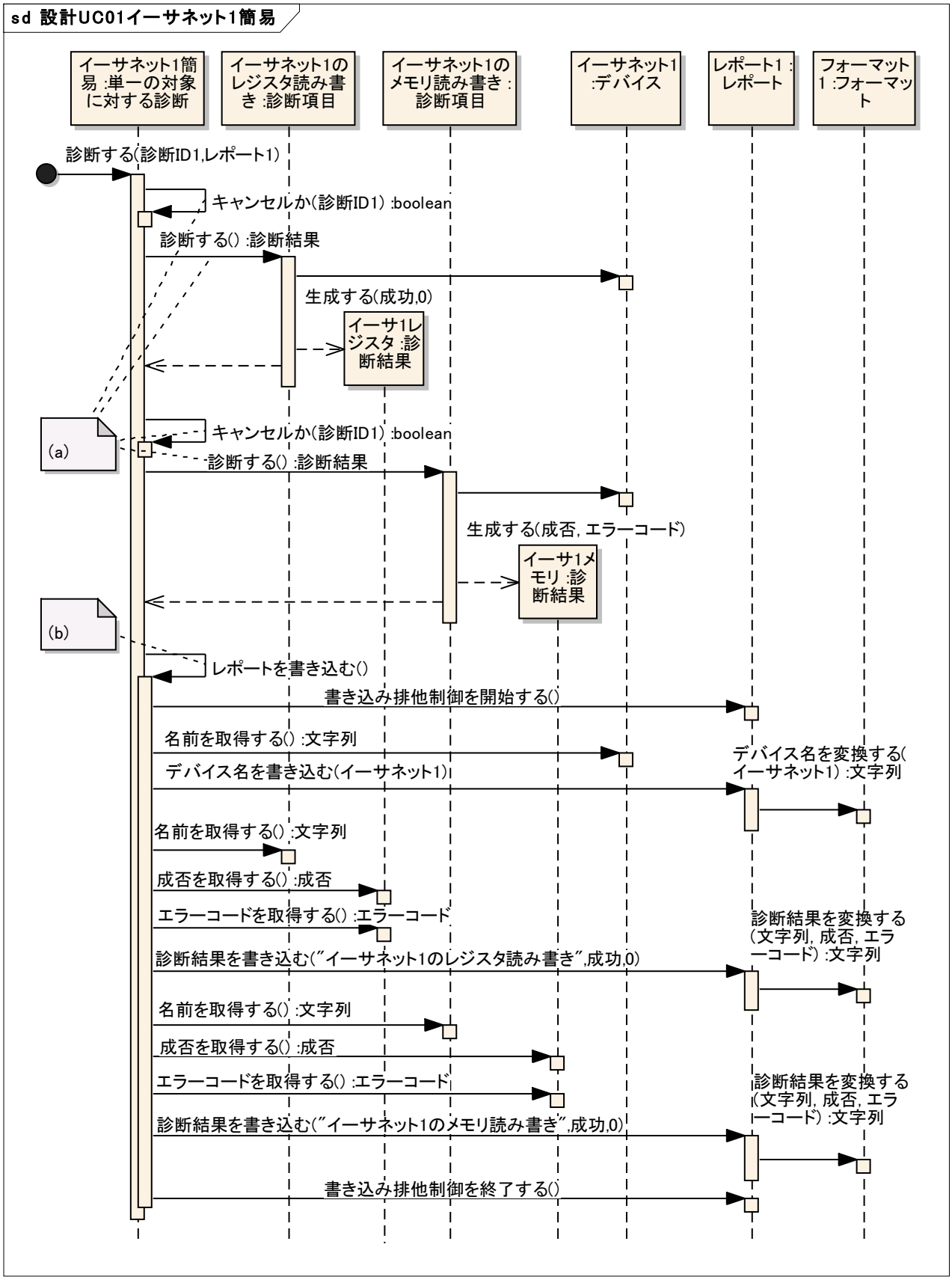
(a)「日次:順番に実行する診断」は「診断する」メッセージを受信すると、キャンセル(診断中止があるか)をチェックしながら、集約する診断インスタンスへ順次「診断する」メッセージを送信します。

(b)(すべての診断を実行し終わると)レポートに排他制御を行い、出力させます。



(a)「イーサネット簡易:並列に実行する診断」は「診断する」メッセージを受信すると、診断タスクを生成し、キャンセル(診断中止があるか)をチェックし、生成した診断タスクをスレッドプールへ渡して診断を起動します。

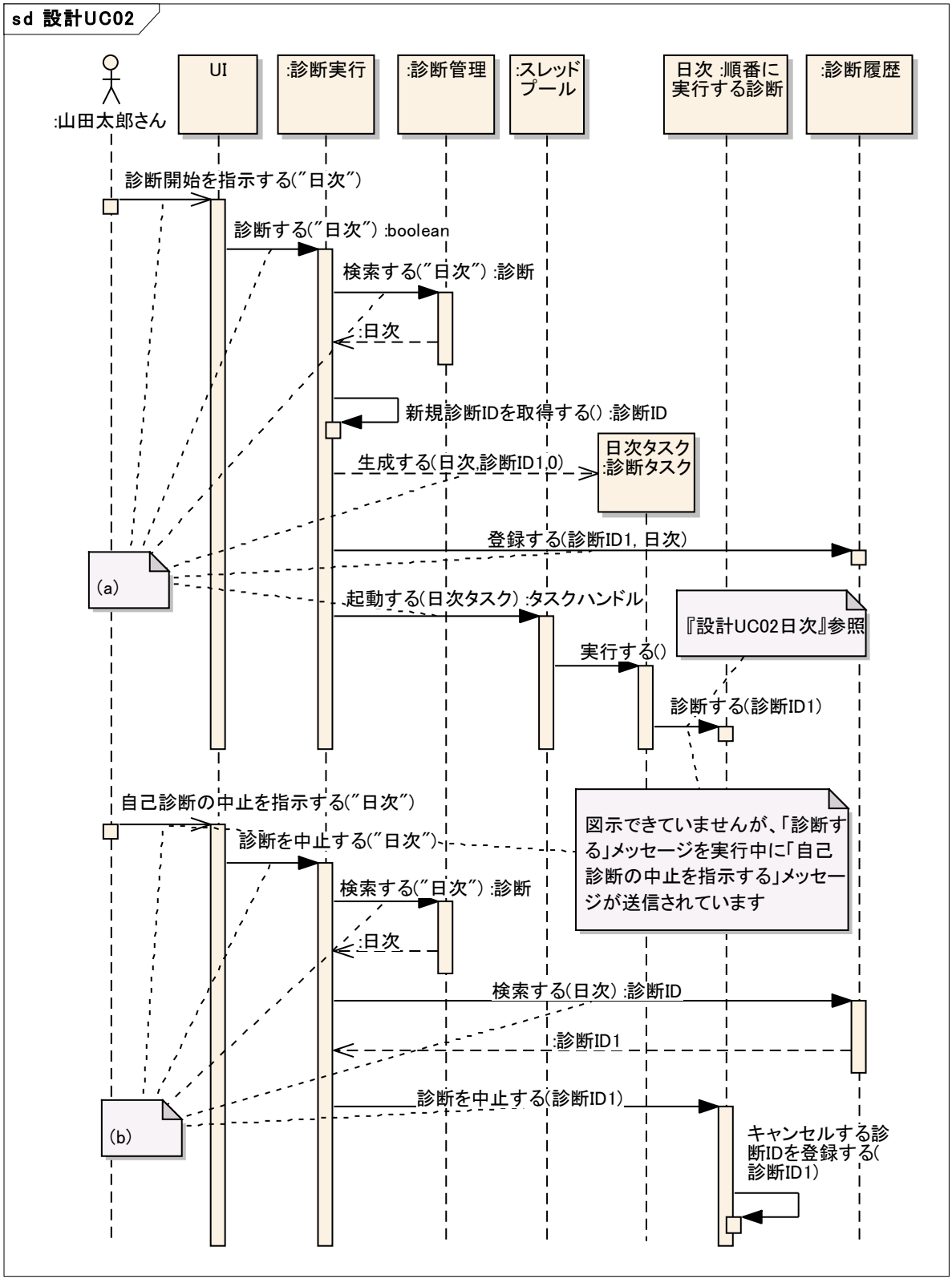
(b)スレッドプールはスレッドを使用して、各診断を並列に実行させます。



(a)「イーサネット 1 簡易:単一の対象に対する診断」は「診断する」メッセージを受信すると、キャンセル(診断中止があるか)をチェックしなら、診断項目に診断を行わせます。

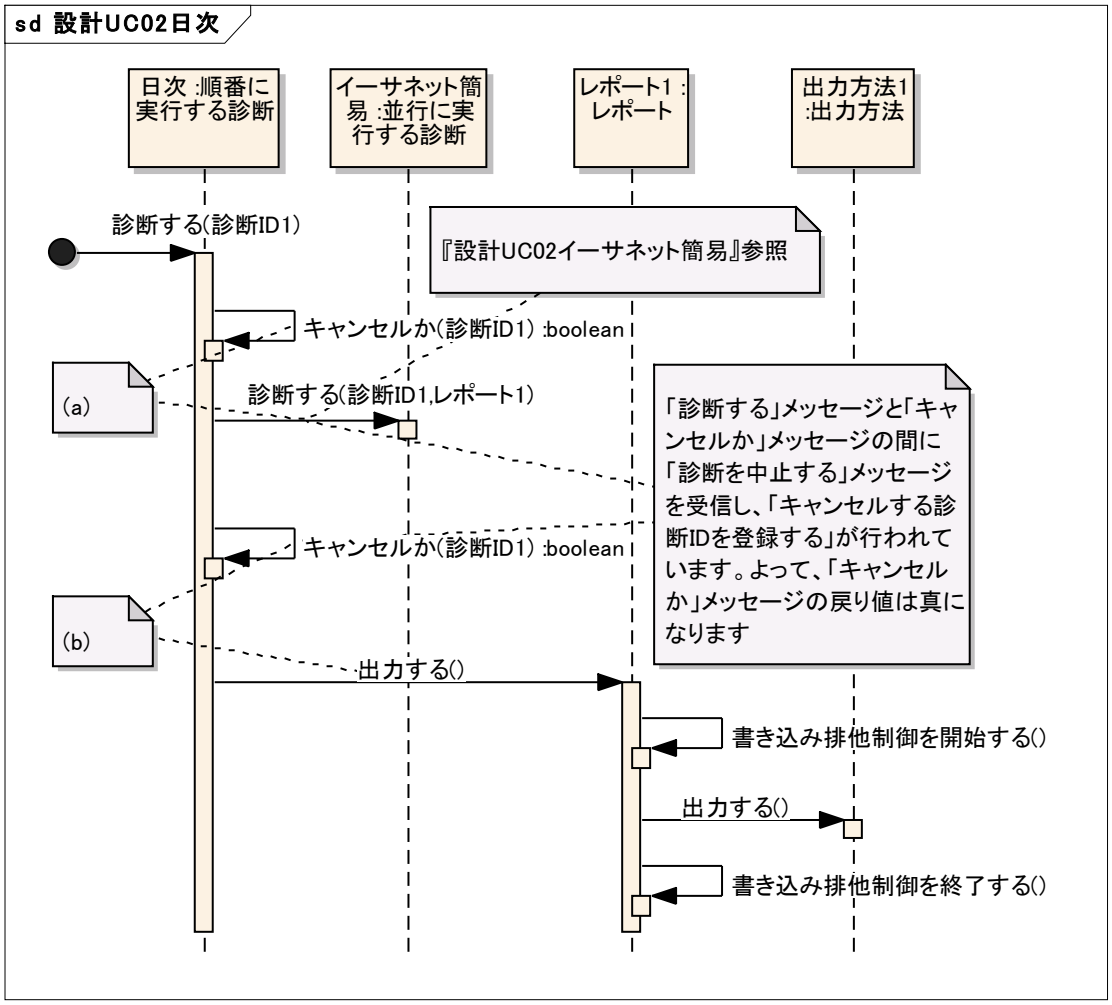
(b)(すべての診断を実行し終わると)レポートに書き込みます。

ユースケース「UC02:自己診断を中止する」のシーケンス図



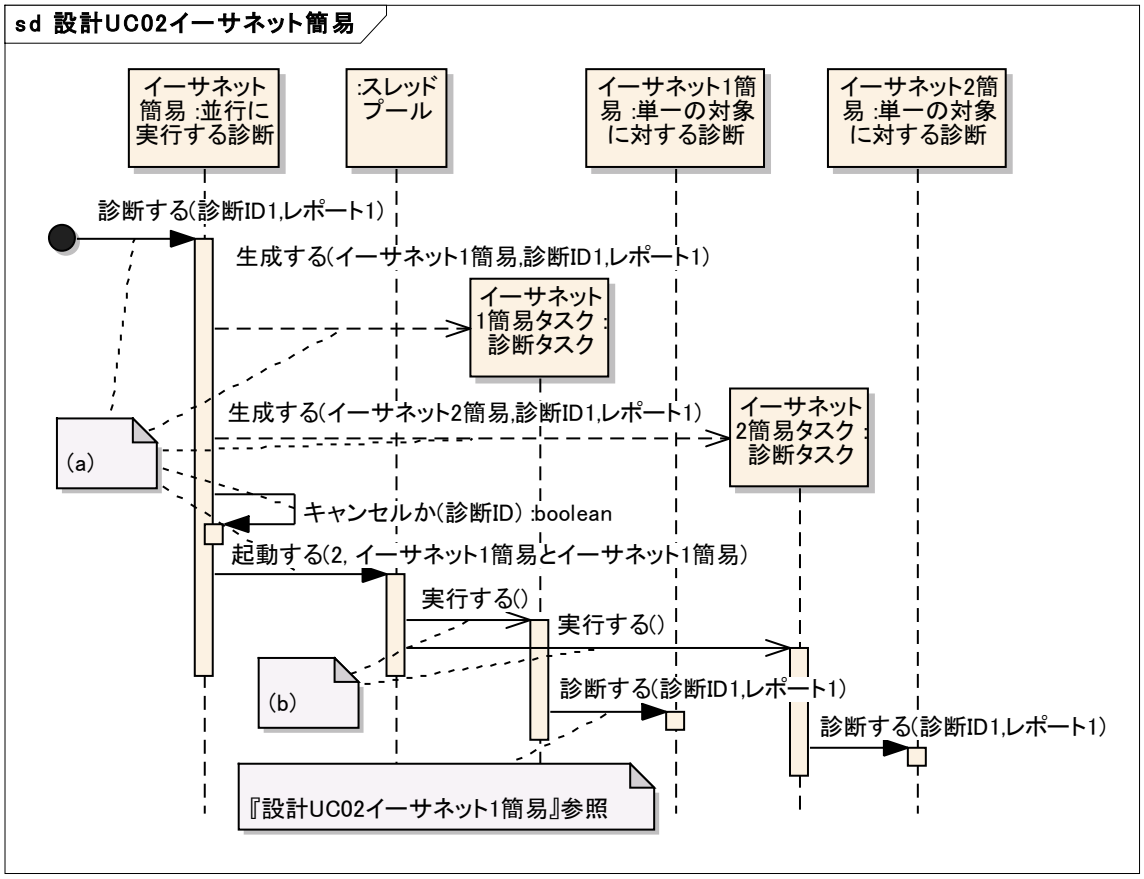
(a)アクターから診断開始の 指示を受けると、ルーターは「ユースケース「UC01:自己診断を行う」のシーケンス図」と同様に動きます。

(b)アクターから診断の中止の指示を受けると、診断実行は診断管理から該当する診断インスタンスを検索し、「診断を中止する」メッセージを送信します。



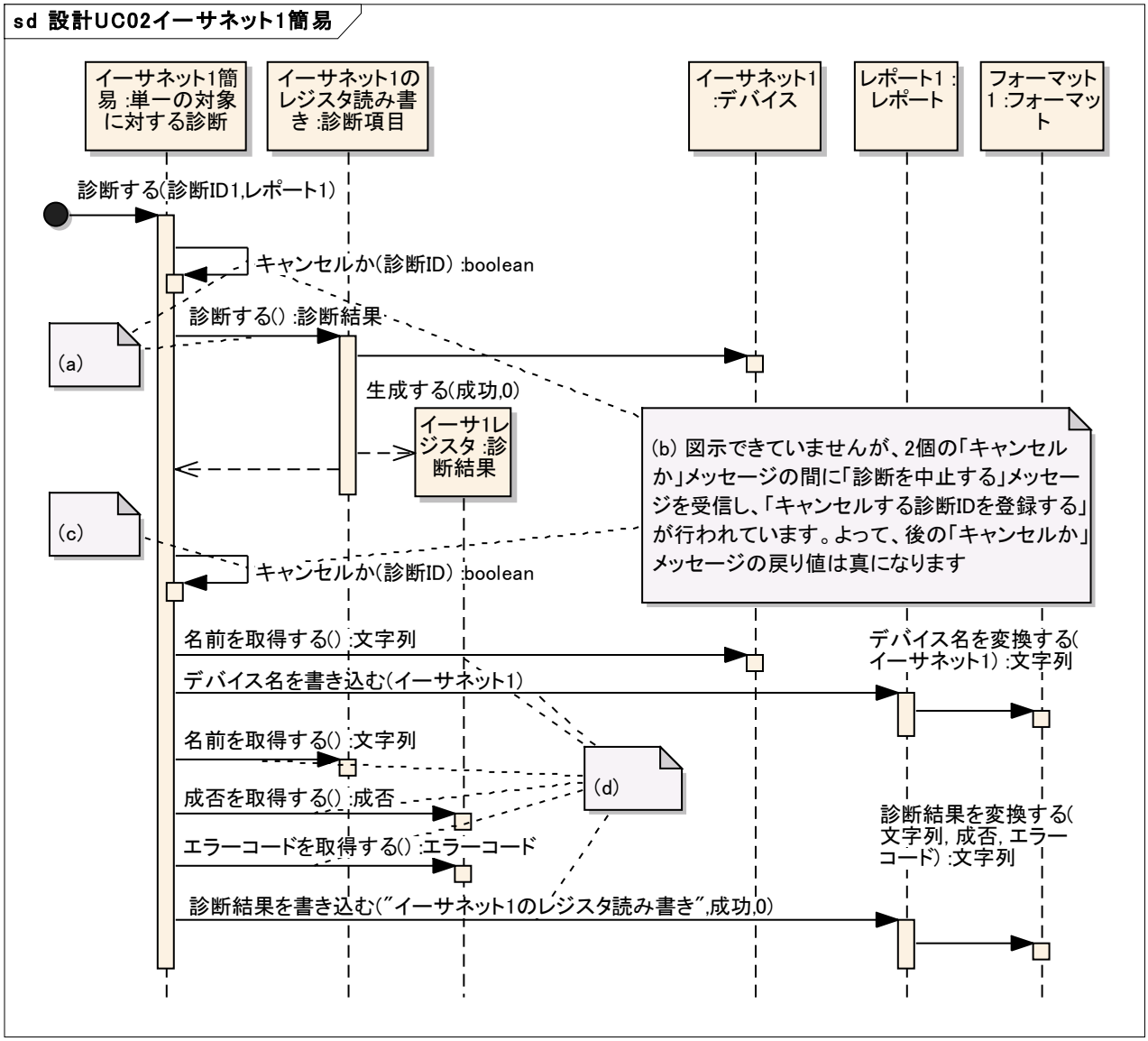
(a)「日次:順番に実行する診断」は「診断する」メッセージを受信すると、キャンセル(診断中止があるか)をチェックしながら、集約する診断インスタンスへ順次「診断する」メッセージを送信します。

(b)キャンセルがあると、「診断する」メッセージの送信を中止します。レポートに排他制御を行い、出力させます。



(a)「イーサネット簡易:並列に実行する診断」は「診断する」メッセージを受信すると、診断タスクを生成し、キャンセル(診断中止があるか)をチェックし、生成した診断タスクをスレッドプールへ渡して診断を起動します。

(b)スレッドプールはスレッドを使用して、各診断を並列に実行させます。



(a)「イーサネット 1 簡易:単一の対象に対する診断」は「診断する」メッセージを受信すると、キャンセル(診断中止があるか)をチェックしなら、診断項目に診断を行わせます。

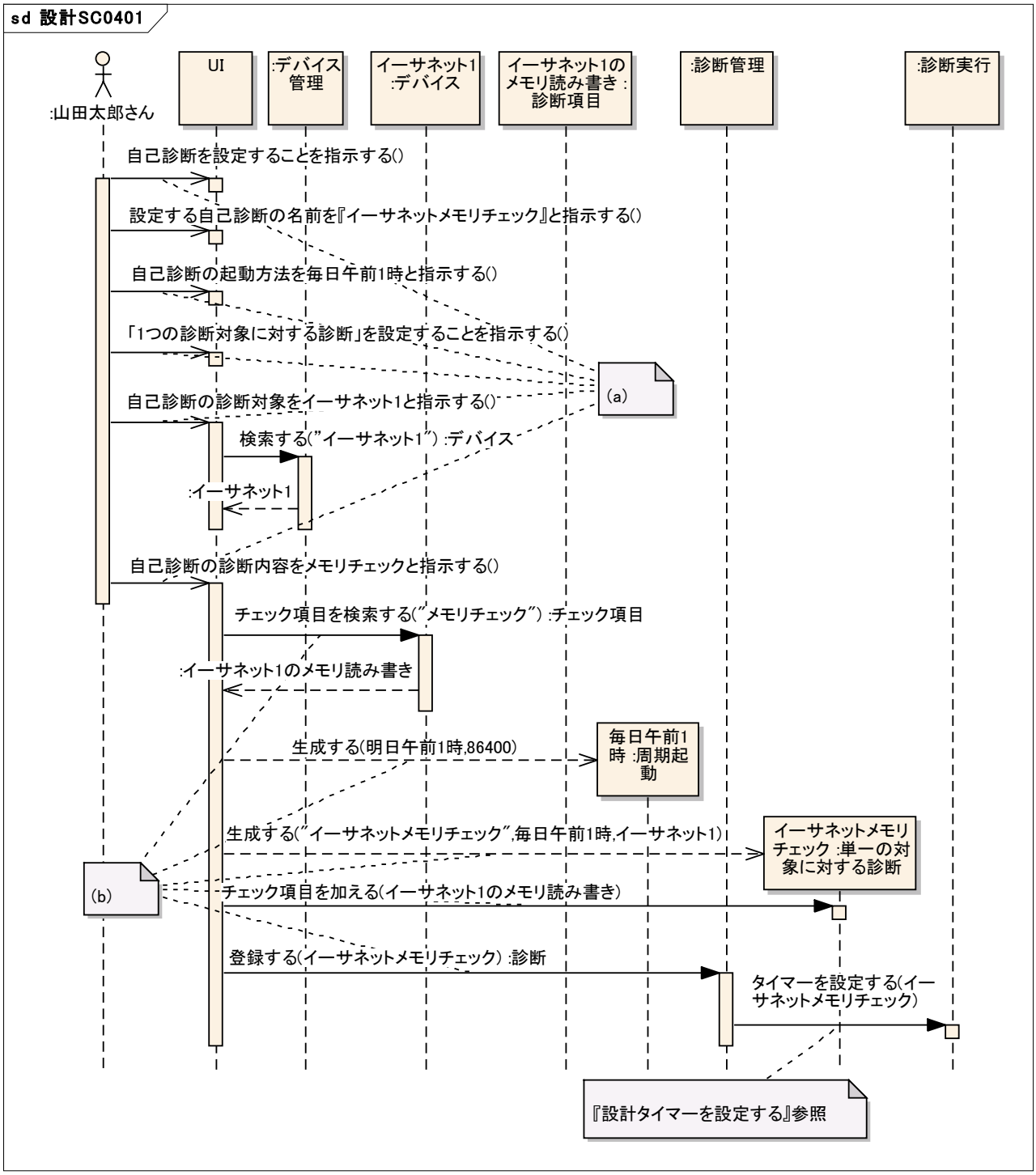
(b)図示できていませんが、「イーサネット 1 簡易:単一の対象に対する診断」からの 2 個の「キャンセルか」メッセージの途中で、「診断を中止する」メッセージを受信しています。

(c)2 個目の「キャンセルか」メッセージの戻り値が真になり、診断を中止します。

(d)レポートを書き込みます。

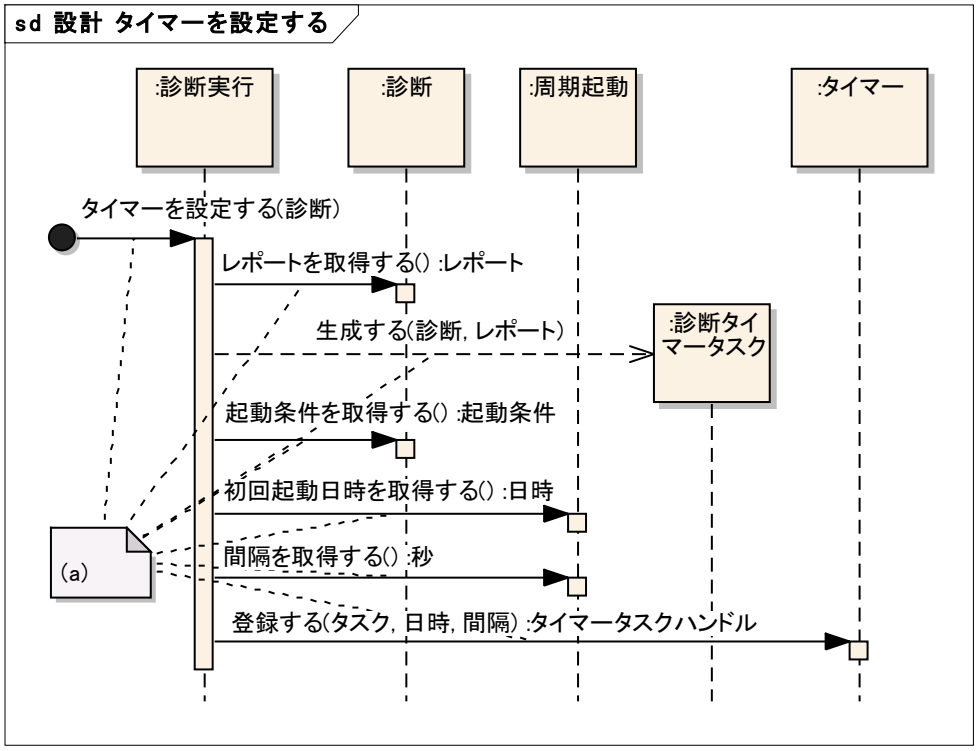
※「キャンセルか」の判断は診断 ID を用いて行います。自身の診断 ID と「診断」クラスの「キャンセルする診断 ID」属性が合致すると、キャンセルとなり、「キャンセルか」メッセージの戻り値が真になります。このモデルでは「キャンセルする診断 ID」属性が 1 個のため、一時点でのキャンセルは一つの診断だけですが、「キャンセルする診断 ID」を複数持つことにより、一時点で複数の診断のキャンセルができるようになります。

シナリオ「SC0401: 自己診断を設定する(1 つの診断対象に対する診断)」のシーケンス図



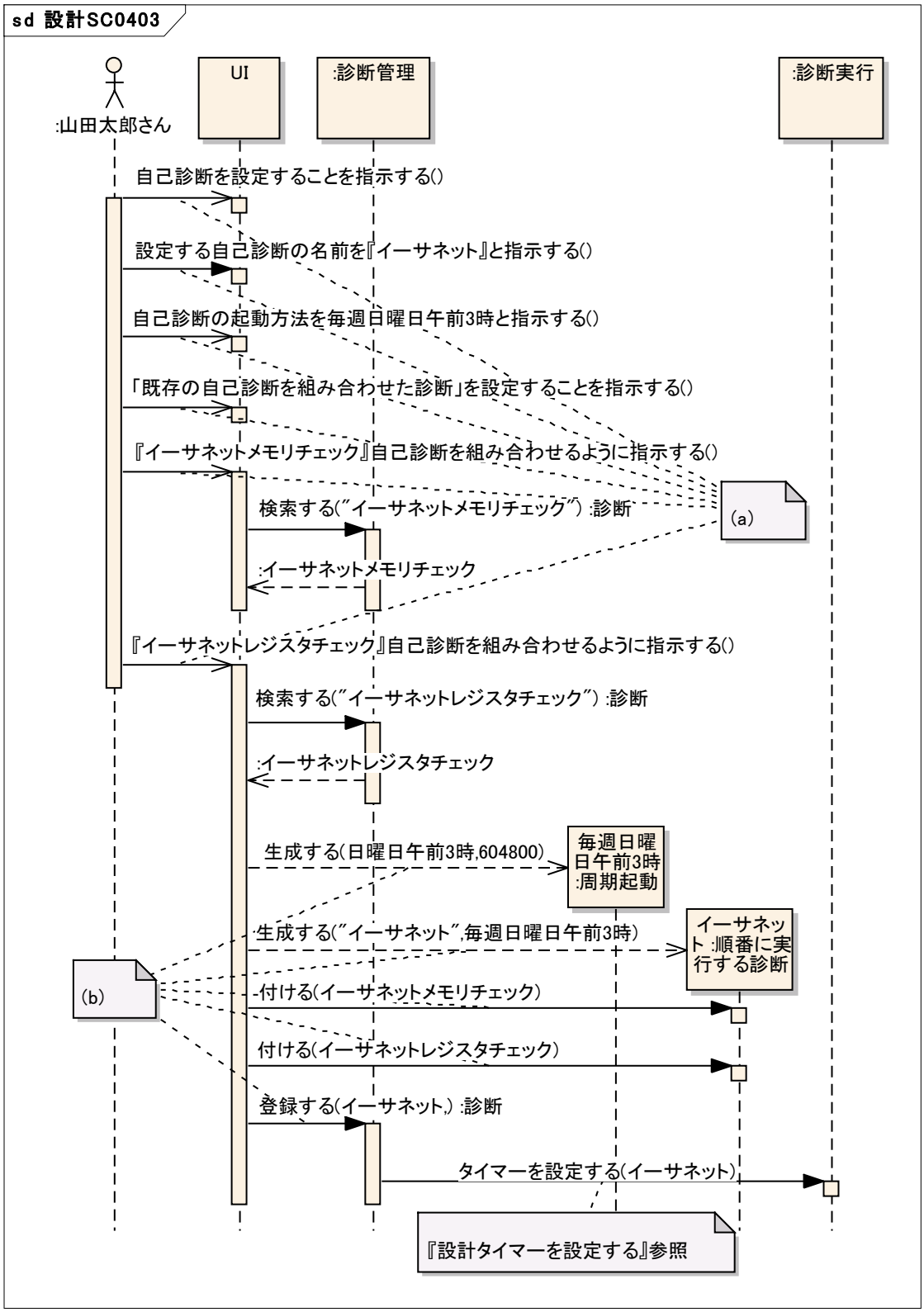
(a)アクターはルーターへ自己診断の名前、起動方法、診断対象、診断内容を指示します。

(b)UI は起動条件と診断を生成し、関連付け、診断を登録します。



(a)「診断実行」は「タイマーを設定する」メッセージを受信すると、必要な情報を取得し、診断タイマータスクを生成し、生成した診断タイマータスクをタイマーに登録します。

シナリオ「SC0403: 自己診断を設定する(既存の自己診断を組み合わせた診断)」のシーケンス図



(a)アクターはルーターへ自己診断の名前、起動方法、組み合わせる診断を指示します。

(b)UI は起動条件と診断を生成し、診断を複製し、関連付けます。

メタファを使って分析したモデル

概念モデル

モデリングのコンセプト

装置の自己診断機能をモデリングするにあたり、まずは診断の意味や目的を考え、日常生活の診断に適用できる概念モデルを作成してみます。診断についての理解を深め、概念を明確にすることで、分析モデルや設計モデルの作成に役立つと思います。

1. 診断って何だろう？

「診断」を国語辞書で調べてみました。(大辞林より)

しんだん 【診断】 （名）スル

(1) 医者が患者を診察し、病状を判断すること。
- 「高血圧と診断する」
- 「診断を下す」

(2) 物事を調べて欠陥がないかなどその状態を判断すること。
- 「企業診断」

つまり、診断とは病気や欠陥など悪いところがないか調べることです。装置の場合は故障箇所や不具合箇所がないか調べることです。悪いところに限らず、悪くなりかけているところや悪くなりそうなところも調べられると、よりよい診断といえるでしょう。

次に、「診断」の類義語を挙げてみます。これらも「診断」とみなしていいでしょう。

◆ 診断の類義語: 検診、点検、検査

ついでに、「自己」とは、国語辞書で調べるまでもなく「自分自身で」ということ。つまり、医者が自分を診断する、企業が自社を診断するということで、装置であれば装置自身で装置内部を診断することです。

2. 診断する目的は何だろう？

診断して悪いところが見つかったらどうしますか？

・医者に言われたなら、病気を治療する
・企業に欠陥があるなら、正す、改善する
・装置であれば、不具合箇所を修理、修復、交換する

悪いところをなおすのはなぜですか？

・人であれば、長生きしたいから。重症になると、介護が必要になったり、お金がかかったりするから。
・企業であれば、企業存続にかかわるから。
・装置であれば、業務に支障が出るから。

悪いところをなおすのは誰ですか？

・他者: 医者、業者
・自己: 自分、自社

以上のことから、問題箇所や予防箇所がないかを診断することで、自己修復あるいは他者に治療・改善・修理依頼するきっかけを知ることができ、診断対象を長生き・長持ちさせるのが目的と考えられます。

3. 身近な診断には何があるだろう？

身近にある診断、点検、検査を探してみました。

◆ 健康診断

・毎年健康診断を受ける。
・Ｆ社では、5 年に 1 回人間ドックに変わる。40 才以上は毎年人間ドック。
・健康診断と人間ドックでは診断項目が異なる。年齢により多少診断項目が異なる。

◆ 車検

・新車は 3 年目、以降は 2 年毎。車検の他に 6 ヶ月点検や 12 ヶ月点検もある。
・点検時期によって点検内容が異なる。

◆ 温水洗浄便座

・自宅の温水洗浄便座のコンセントプラグに「テスト」ボタンがある。
・ボタンを押すと正常に動作するか診断してくれるようだ。
・手洗い時に水が跳ねたり湿気を含んでショートしたりする心配があるからだろうか？

◆ 分電盤

・自宅の分電盤の蓋をあけると「テスト」ボタンがある。
・漏電やショートしていないか診断してくれるのだろう。

◆ 消防設備点検

・自宅マンションでは、年に 2 回、火災報知器が正しく動作するか調べに来てくれる。
・会社でも、年に 2 回くらい消防設備を点検しているのを見かける。

4. 診断に関する共通項目には何があるだろう？

前述の身近な診断(点検、検査を含む)において、共通する項目を挙げてみます。

対象物 / 共通項目	人	車	温水洗浄便座	分電盤	消防設備
診断名	健康診断 人間ドック	車検 定期点検	テスト	テスト	消防点検 防災点検
診断時期	年 1 回 望んだとき	初回 3 年 以降 2 年毎	ボタンを押したとき	ボタンを押したとき	半年毎
診断対象	本人、子供、社員	車	電気設備	電気設備	消防設備
診断依頼者	本人、親、会社	所有者	家人	家人	本人、管理会社、企業
診断場所	病院、健康管理室	車ディーラー、車検場	設置場所	設置場所	設置場所
診断実施者	医者	メカニック	装置	装置	業者
診断項目	血液、血圧、視力、聴力、胃、肺、心臓など	エンジン、ブレーキ、オイル、タイヤなど	ショート	漏電、ショート	熱検知、煙検知、防炎扉、シャッター等
診断結果	健康ランク、問題部位、状態など	問題なし、 要修理箇所、 予防修理箇所	ブザー？	ブザー？	異常なし 異常個所

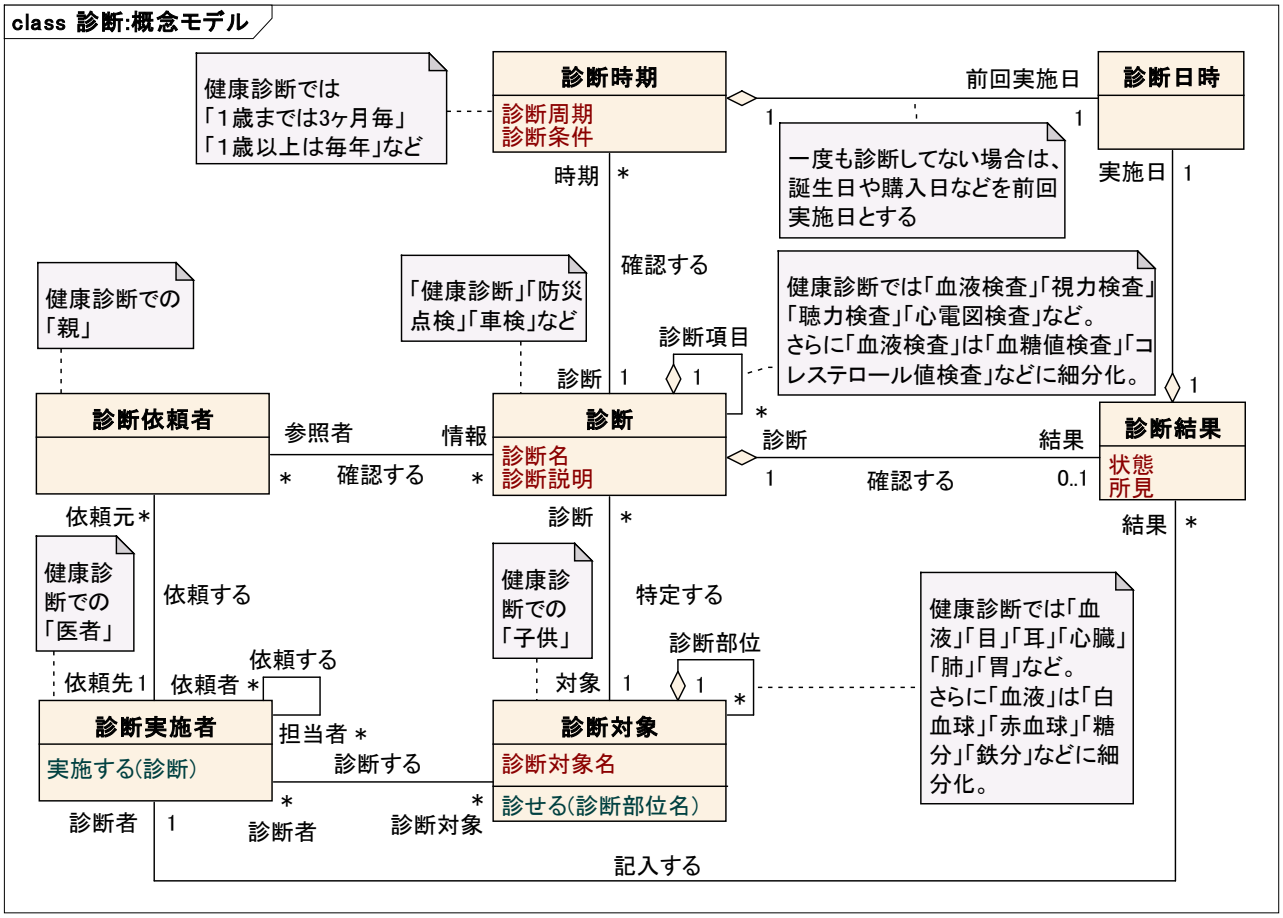
5. 診断シナリオ

共通項目を使って、診断するときの手順(シナリオ)を考えてみます。

- ◆ 登場人物(アクター)は「診断依頼者」「診断実施者」「診断対象」の 3 人。
- ◆ 本人が本人の診断を依頼すれば「診断依頼者」と「診断対象」は同一です。
- ◆ 「診断依頼者」は、「診断時期」になった「診断」がないか定期的に確認し、あれば「診断実施者」に「診断対象」の「診断」を依頼します。
- ◆ 「診断時期」に関係なく、「診断対象」が調子悪くなったり、なんとなく気になったときに「診断依頼者」が「診断実施者」に「診断対象」の「診断」を依頼することもあります。
- ◆ 「診断実施者」は、「診断項目」毎に「診断対象」を「診断」し、「診断結果」を記入して「診断依頼者」に通知します。
- ◆ 「診断対象」は「診断部位」を診せます。
- ◆ 「診断依頼者」は、「診断結果」を確認し、問題があれば治療や修理などを施します。

クラス図

共通項目を使って、概念モデルのクラス図を書いてみます。



- 「診断」には「診断名」があり、いつ頃(診断時期)、誰/何/どの部位 (診断対象) を、どのように診断するのか(診断説明)が分かります。
- 「診断」には「診断時期」があるものとないものがあります。
- 「診断時期」がある場合、「前回診断日」から「診断周期」が経過していたら「診断」する時期です。一度も診断してない場合、誕生日、購入日、設置日、使用開始日などを「前回診断日」とします。
- 「診断時期」には「診断条件」がある場合があります。車であれば「初回は 3 年目」「以降は 2 年毎」といったように「経年」により「診断周期」が変化します。
- 「診断時期」に関係なく、「診断対象」が調子悪くなったときやなんとなく気になるときなどに、「診断依頼者」が「診断実施者」に「診断対象」の「診断」を依頼することもできます。
- 「診断」は「診断項目」に細分化されることがあります。さらに「診断項目」が詳細化されることもあります。
- 「診断実施者」は「診断」を実施し、「診断」に応じた「診断結果」を記入します。「診断項目」がある場合、「診断項目」毎に「診断結果」を記入したあと、それを要約して「診断」の「診断結果」に記入します。

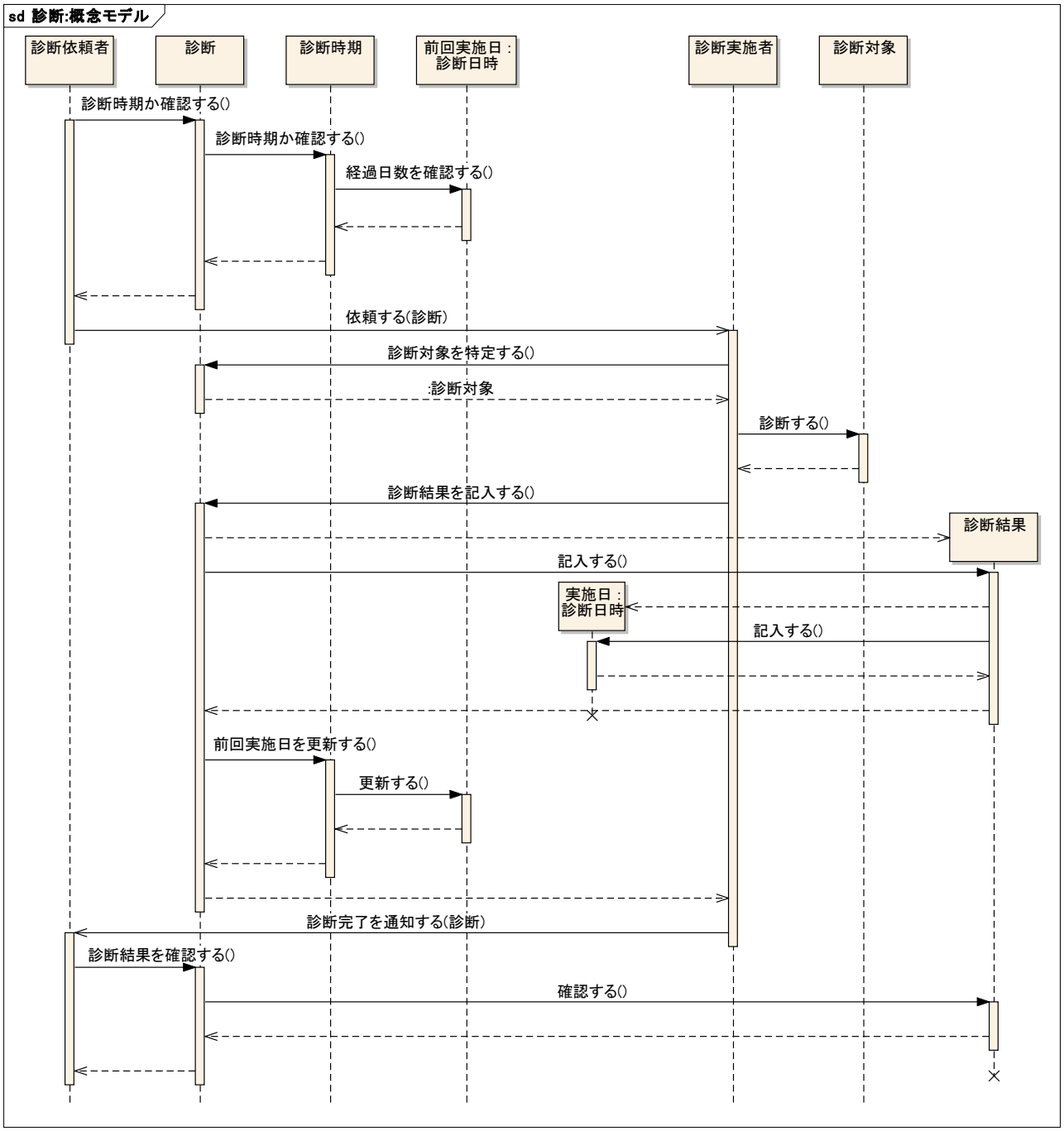
◆ 「診断実施者」は、専門の「診断担当者」に「診断」を依頼することもあります。さらに「診断担当者」は他の「診断担当者」に手伝いを依頼することもあります。

◆ 「診断対象」は「診断部位」を診せることができます。あるいは、診せる方法があります。

シーケンス図

処理の流れをシーケンス図で書いてみます。

シーケンス図により、クラスに追加・変更する操作がわかってきます。



分析モデル

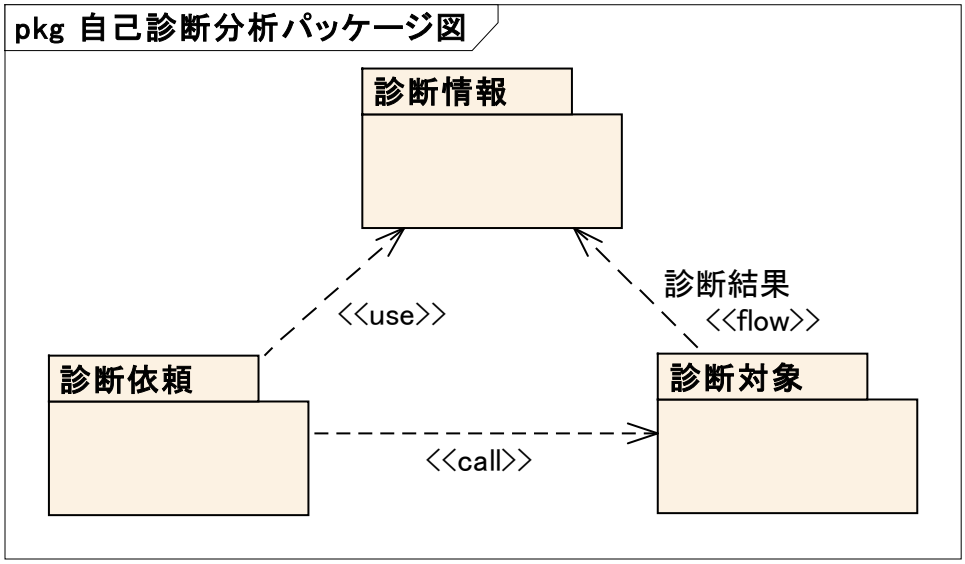
モデリングのコンセプト

要求仕様で出てくる要素として「自己診断」「診断対象」「診断項目」「診断結果」があります。それらの関係に重点を置き、概念モデルを参考にしつつ、ユースケースをできるだけ忠実にモデル化してみます。

なお、診断対象がどのように診断するのかについては関知せず、並行実施するか順次実施するかも診断対象にまかせることにし、診断を実施することによって診断結果がレポートされることに重点を置きます。

パッケージ図

まずは、概念モデルを元にパッケージに分類してみます。



概念モデルで出てきた「診断依頼者」は「診断依頼」とします。ここには、管理者からの「自己診断」の要求を受け付けて、「診断対象」に依頼するまでの部分を担当します。

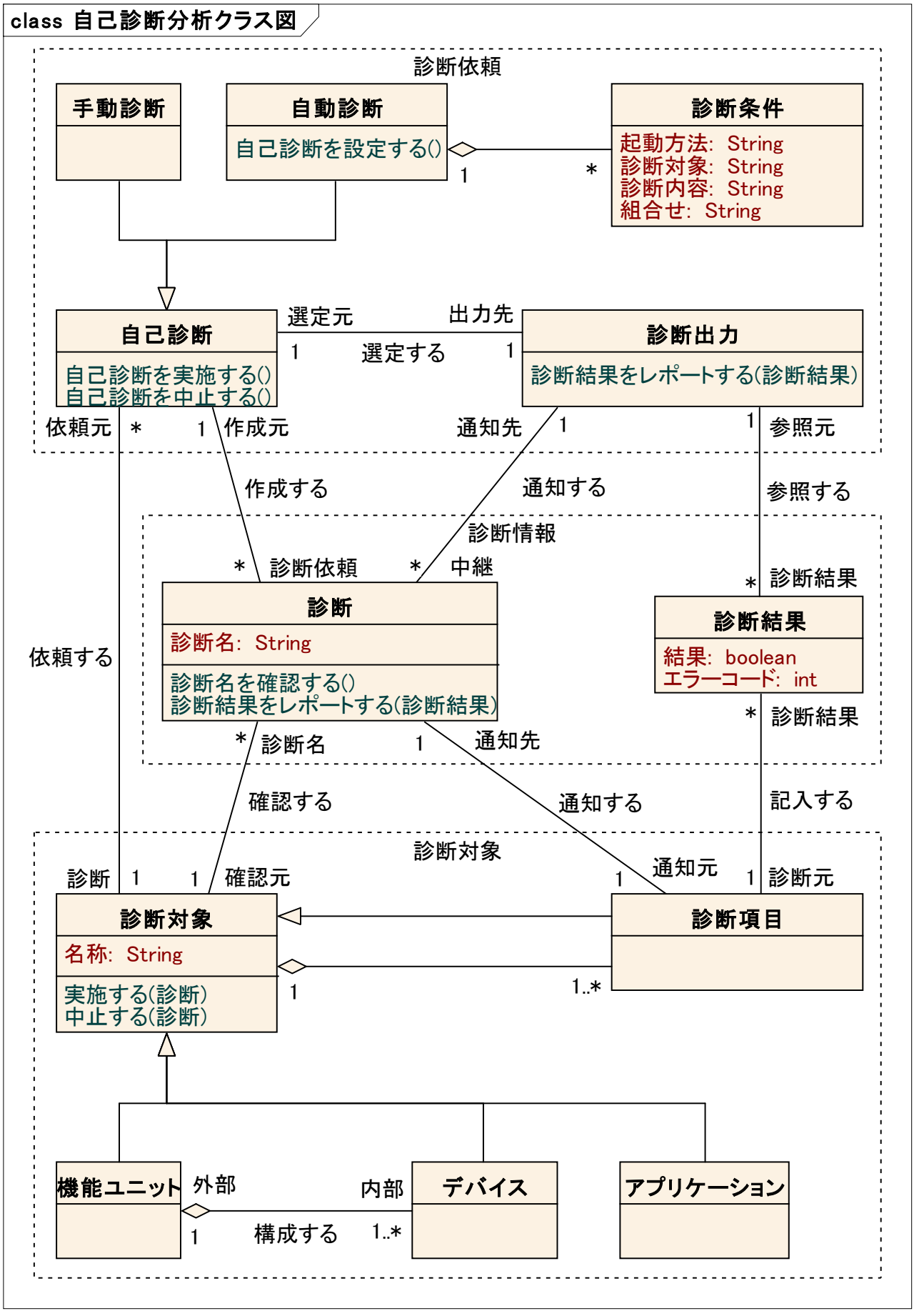
自己診断では、「診断対象」自身が「診断実施」するので、「診断実施」は「診断対象」にまとめます。

「診断依頼」と「診断対象」の間でやりとりされる情報を「診断情報」としてまとめます。ここには「診断」と「診断結果」が入ります。

概念モデルでの「診断時期」は、自動診断の設定情報として「自己診断」に含めることにします。「診断日時」は要求仕様には出てこなかったので扱わないことにします。

それらを踏まえてクラス化していきます。

クラス図


class 自己診断分析クラス図
診断依頼
手動診断
自動診断
自己診断を設定する()
診断条件
起動方法: String
診断対象: String
診断内容: String
組合せ: String
1
*
自己診断
自己診断を実施する()
自己診断を中止する()
選定元
出力先
1
選定する
1
診断出力
診断結果をレポートする(診断結果)
依頼元
*
1
作成元
通知先
1
1
参照元
作成する
通知する
参照する
診断情報
*
診断依頼
*
中継
*
診断結果
診断
診断名: String
診断名を確認する()
診断結果をレポートする(診断結果)
診断結果
結果: boolean
エラーコード: int
依頼する
*
診断名
1
通知先
*
診断結果
確認する
通知する
記入する
診断対象
診断
1
1
確認元
1
通知元
1
診断元
診断対象
名称: String
実施する(診断)
中止する(診断)
1
1..*
診断項目
機能ユニット
外部
内部
デバイス
アプリケーション
1
構成する
1..*

「自己診断」には、機器管理者の要求により起動される「手動診断」と、予め設定された「診断条件」に合わせて起動される「自動診断」があります。いずれも「自己診断」のサブクラスです。

「自己診断」を実施する際には、「診断結果」をレポートするための「診断出力」を用意しておきます。コンソール出力、ロギング出力、LED 点滅など、いろいろな「診断出力」の実装が考えられます。

「診断対象」には「機能ユニット」「デバイス」「アプリケーション」があります。いずれも「診断対象」のサブクラスです。「機能ユニット」はいくつかの「デバイス」から構成されます。

「診断対象」には「診断項目」が決まっています。「自己診断」を実施する毎に「診断結果」がレポート(通知)されます。「診断項目」に応じた診断が行われるので「診断項目」も「診断対象」のサブクラスにして診断の実施と中止ができるようにしておきます。

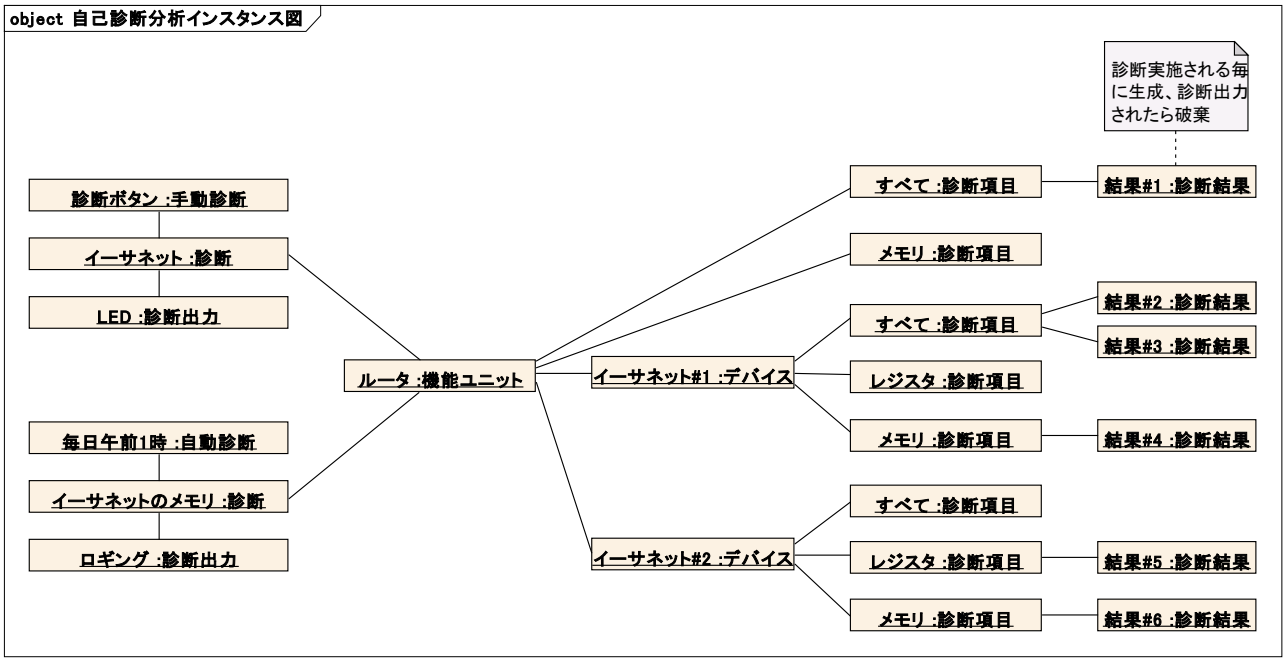
「診断対象」で診断を実施するときは、何の診断なのかを表す診断名と、「診断結果」をレポートできる「診断」を渡すことにします。「診断」は並列動作可能なので、「診断結果」は非同期で通知できるようにしておきます。「診断結果」が通知されたら診断完了したとみなすことができます。つまり、「診断」には診断実施中なのか診断完了したのかといった状態が存在することになります。

要求仕様にできるだけ忠実にしたためこのようなクラス図になりましたが、「診断」自身が診断実施できた方がいいなどの見直しは PIM 設計で行うことにします。

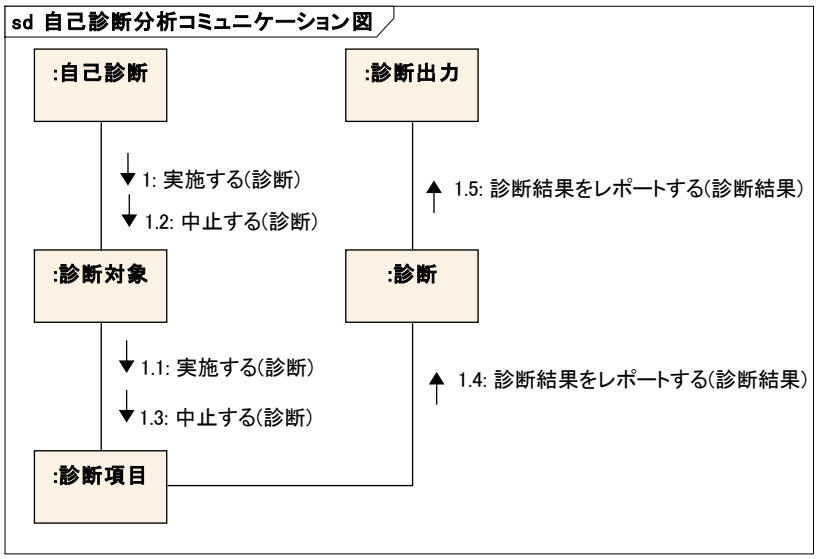
オブジェクト図

ユースケース記述に記載されたインスタンスに、関連しそうなインスタンスを加えたインスタンス図を以下に示します。

診断毎に診断結果が生成され、診断出力されることになります。

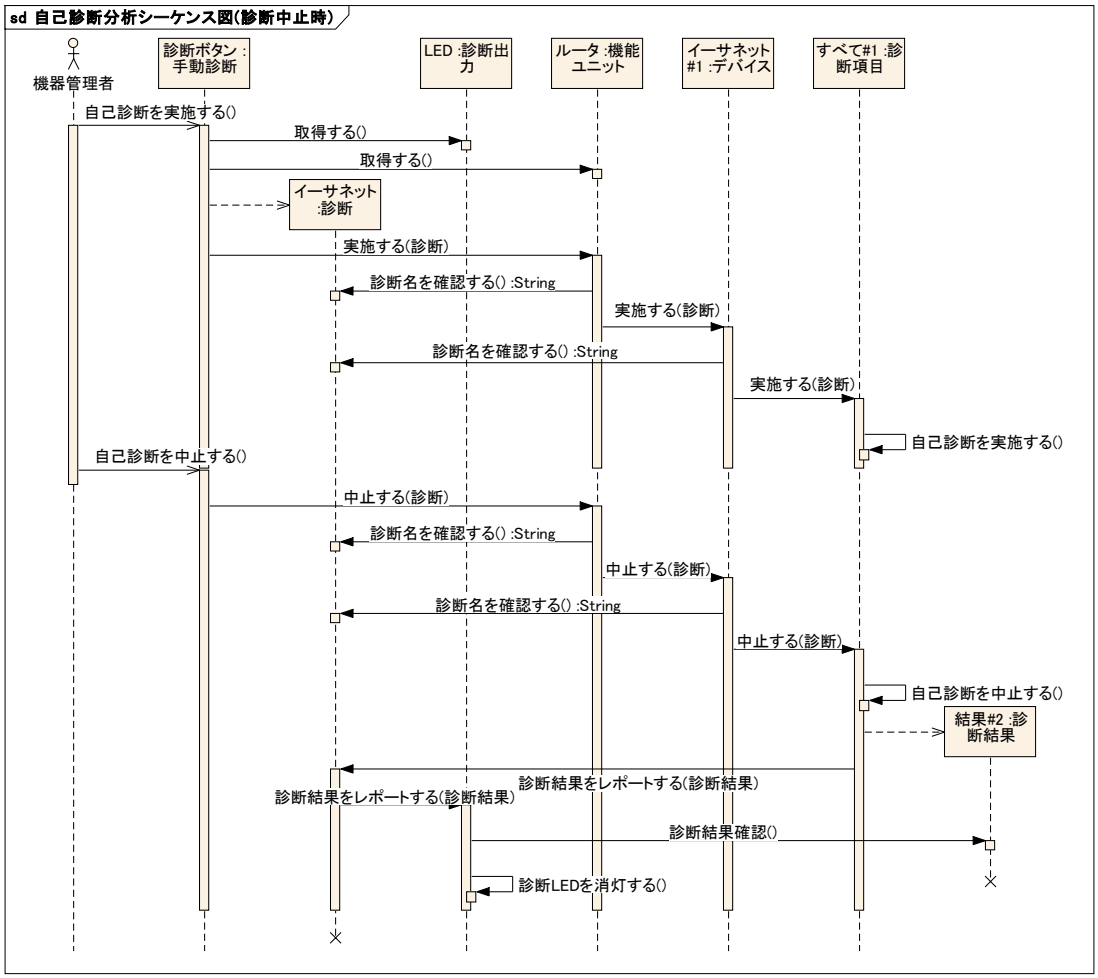


診断の実施および診断を中断した場合のコミュニケーション図を以下に示します。



シーケンス図

管理者の指示によりイーサネットの自己診断を実施し、診断を中止ときのシーケンス図を示します。

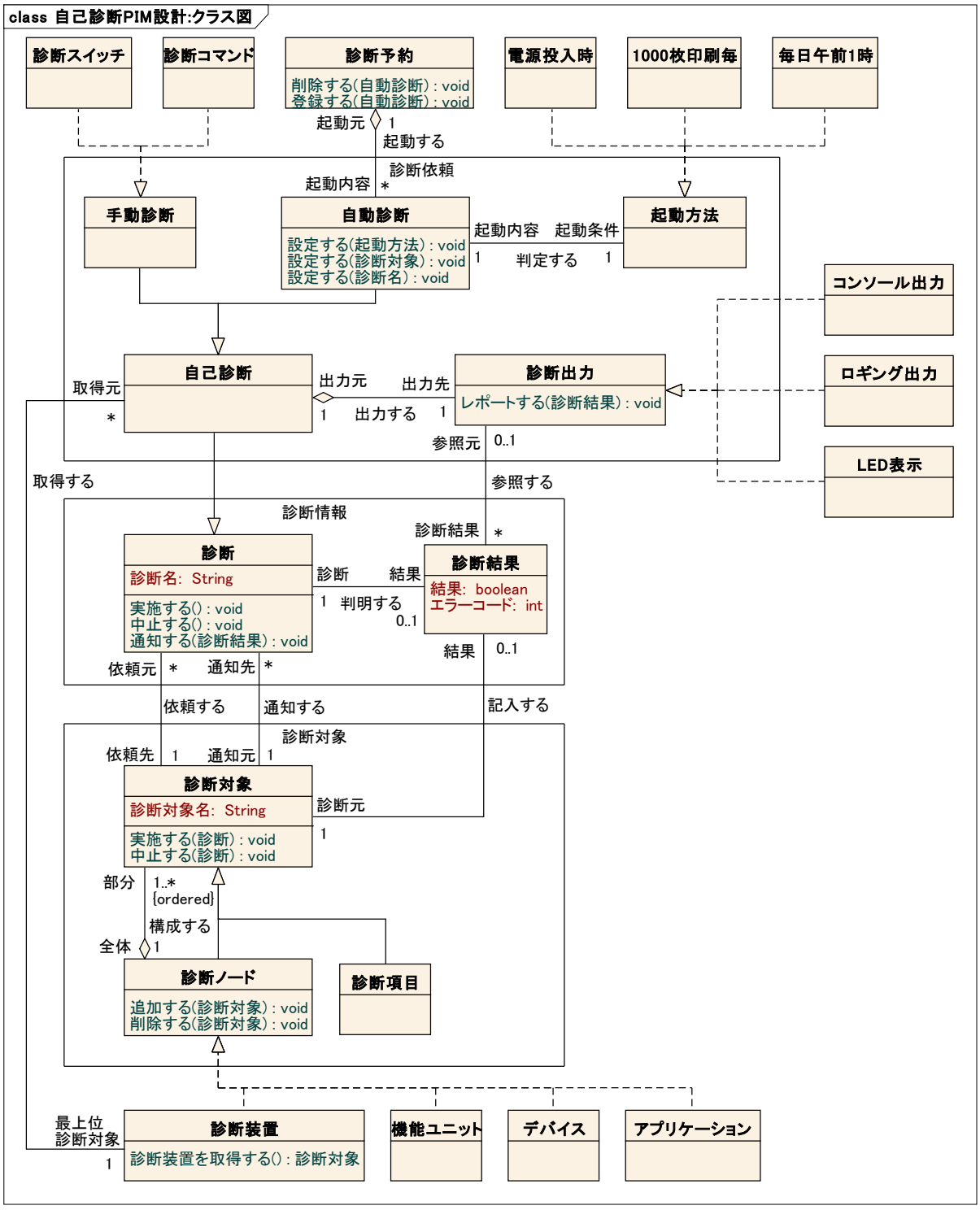


PIM 設計モデル

モデリングのコンセプト

分析モデルの関係性を崩さないように、依存性をできるだけ排除したコンパクトなモデルにします。

クラス図



「自動診断」は、「診断予約」として複数登録できるようにしました。ただし、「診断予約」は大抵の装置に備わっていると思われるスケジュール機能を利用することを想定し、本モデルでは詳しく取り上げないことにします。

分析モデルでは「自己診断」から「診断対象」に診断実施を指示していましたが、パッケージの独立性を高めるため、「診断」で診断実施できるようにし、「診断依頼」パッケージと「診断対象」パッケージを完全分離させました。

「診断」は、生成したあと 1 度だけ診断実施して破棄することができますが、「診断」をひとつ生成したあと何度も診断を実施して最新の診断結果を受け取ることもできます。

分析モデルでの「診断対象」は「診断項目」を保持していましたが、「機能項目」自体も「診断対象」のサブクラスであり、その構造がファイルシステムの「ディレクトリ」と「ファイル」のツリー構造に類似しているので、「ディレクトリ」相当を「診断ノード」とし、「ファイル」相当を「診断項目」としました。これにより、診断対象がファイルのディレクトリ名のように指定でき、以下のように診断対象を文字列で指定できるようにします。

- すべての診断: "*"
- メイン基板のすべての診断: "main/*"
- メイン基板のメモリ診断 : "main/memory"
- すべての基板のメモリ診断: "*/memory"

なお、「診断ノード」から下位の「診断対象」に診断を依頼するには、「診断」オブジェクトが必要になります。依頼元から渡ってきた「診断」をそのまま渡すと、複数の「診断結果」が「診断出力」に渡ることになり、診断経過が出力されるような動作になります。ひとつの診断毎に「診断結果」を待ち合わせるような場合には、一時的な「診断」オブジェクトを生成して診断依頼し、診断結果を待ち合わせるようにする必要があります。

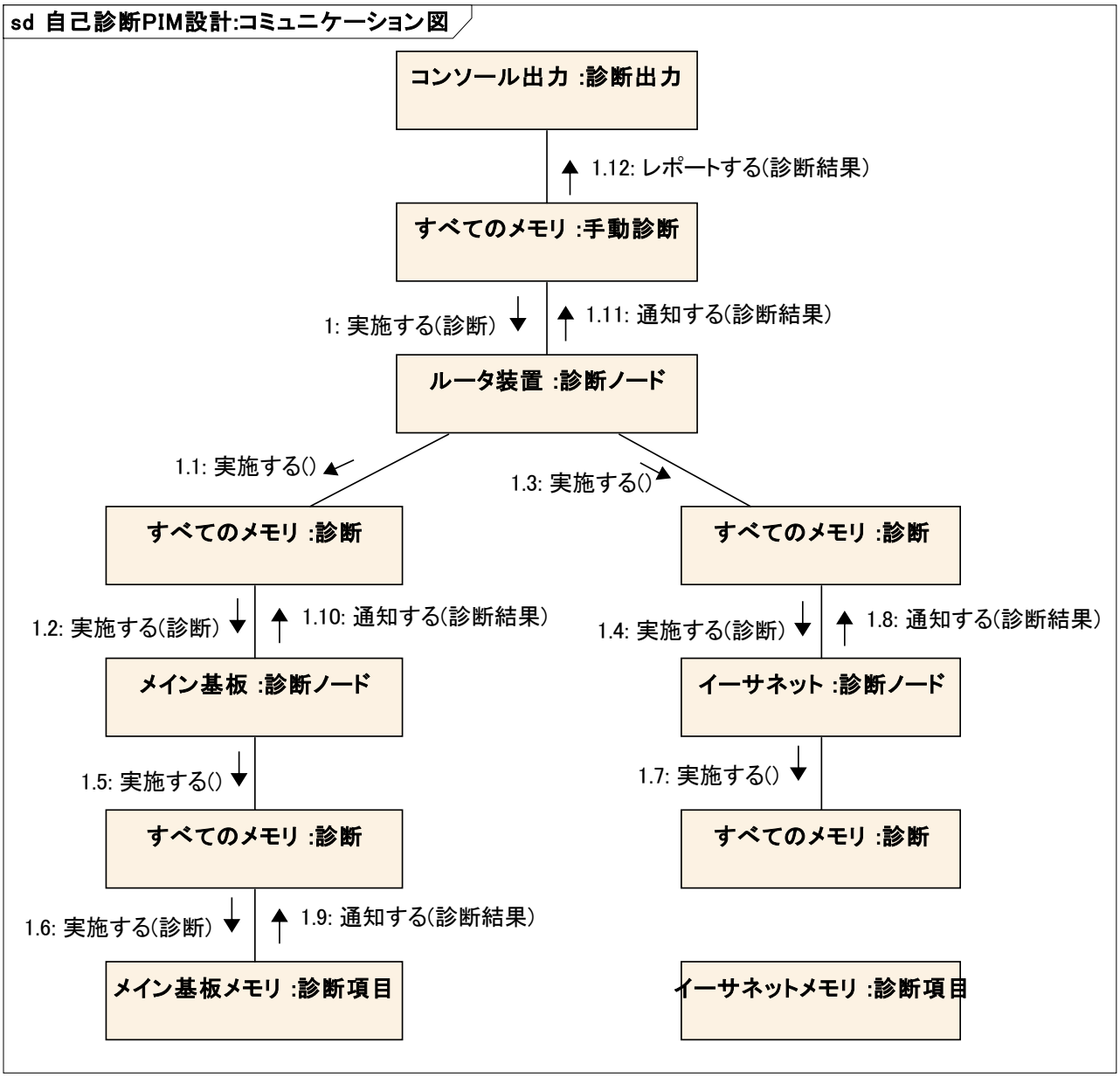
オブジェクト図

オブジェクト図は分析モデルと同等です。

コミュニケーション図

以下に、すべてのメモリ診断を実施する場合のコミュニケーション図を示します。

メイン基板のメモリ診断とイーサネットのメモリ診断は並列動作し、イーサネットのメモリ診断の方が先に終わる順序にしてあります。



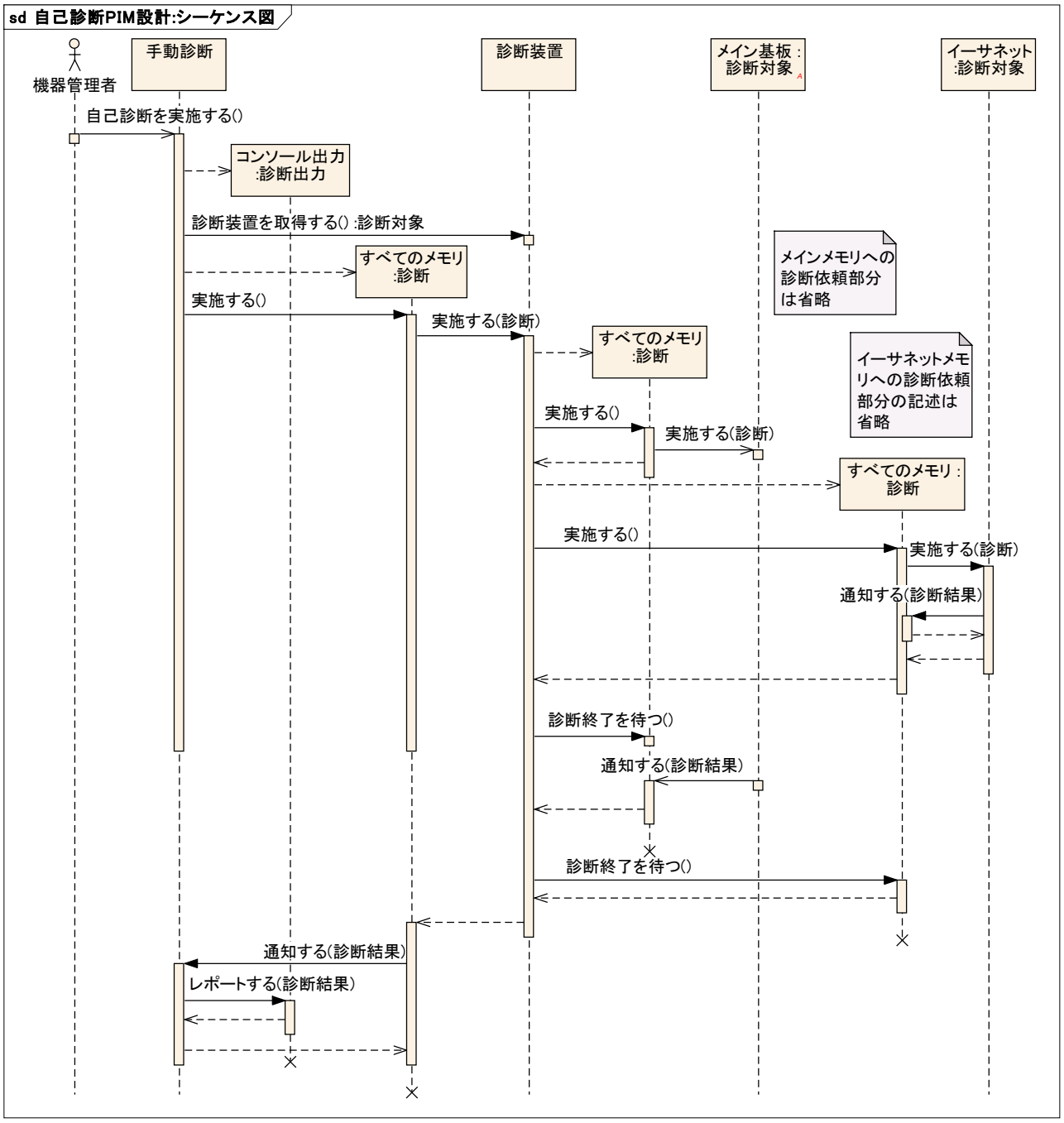
シーケンス図

メイン基板とイーサネット基板があり、それぞれにメモリが搭載されていて、メモリを診断する場合のシーケンス図を以下に示します。

メイン基板での診断はハードウェアで行われ、診断依頼は即時復帰し、診断完了は割り込みで通知される非同期型を想定しました。ソフトウェアで行う場合でも、スレッド生成して非同期型にすることもできます。

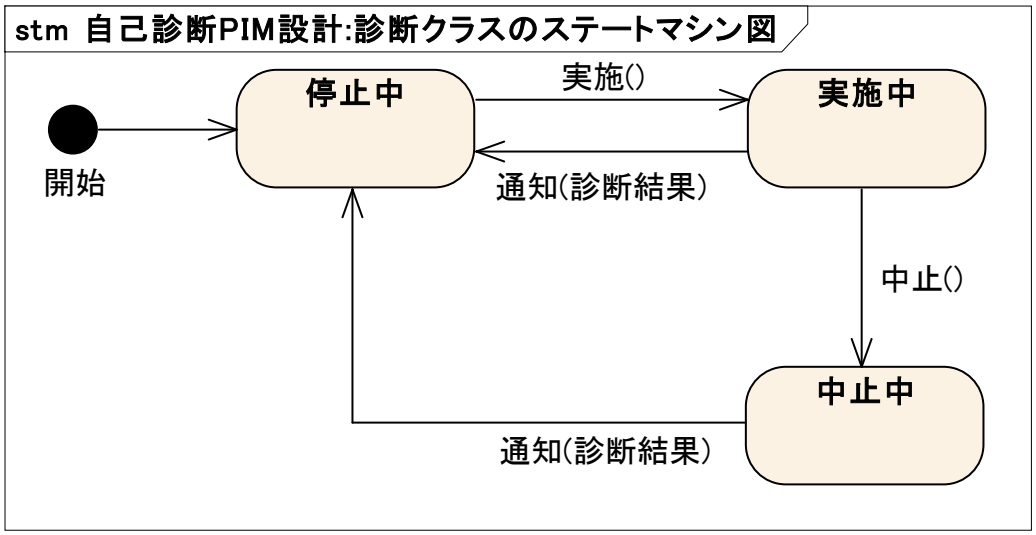
サブ基板での診断はソフトウェアで行われ、診断実施は診断終了まで復帰しない同期型を想定しました。診断装置もソフトウェア実装なので同期型にしました。

診断対象が診断結果を生成する記述は省略しました。診断結果をオブジェクト(クラス)にせずにエラーコードを直接診断に通知するように仕様変更してもいいと思います。

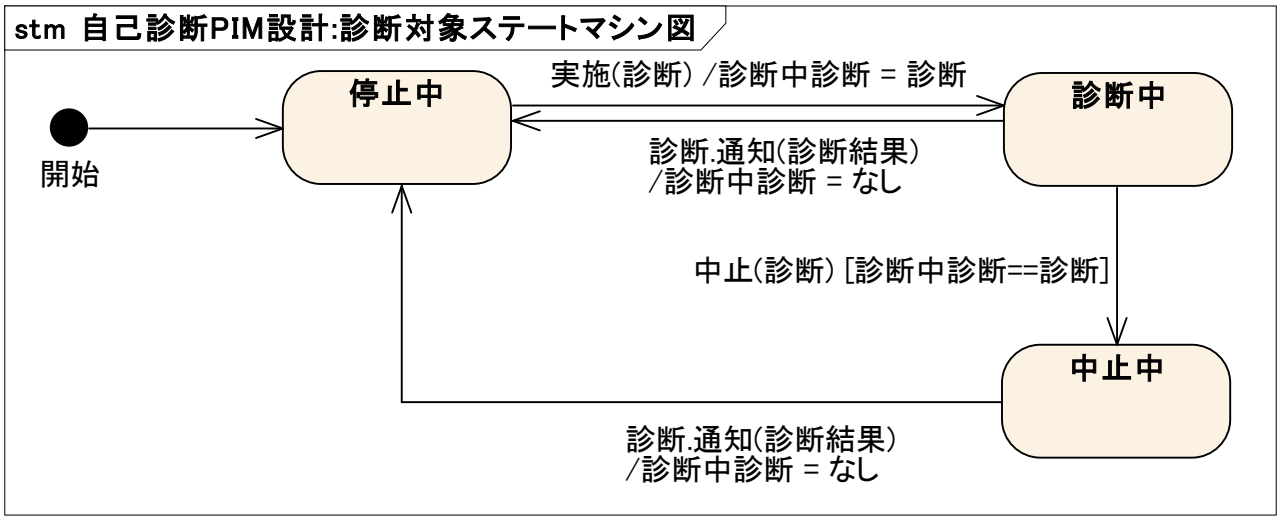


ステートマシン図

診断対象クラスのステートマシン図



診断対象クラスのステートマシン図



おわりに

2009 年 11 月のキックオフで活動を開始してから、約 10 ヶ月。

所属はもちろんのこと、仕事の内容や経験も異なる約 20 名のメンバが、業務の合間を縫いながら月一ベースでの定例打ち合わせを重ね、当初の目標である「モデルカタログの第一版」をなんとかリリースすることが出来ました。

モデルカタログに対する意義や期待についてはメンバ全員で共通していたものの、実際に作るとなると、その前提となるプロセスやモデリング手法、対象とする分野等々、いろいろな検討事項がありましたが、今回は、立場や経験の異なるメンバが限られた時間で活動するということを前提に、「まずは各自の思い描くモデルを描いて、それをカタログにしてみよう」というところからスタートしました。

少ない活動の中、モデルカタログを利用する方のために、その作成においては下の 4 点を考慮しました。

カタログの使い方をきちんと明示すること

複数の分野・対象についてのモデルを掲載すること

同じ分野・対象については、複数の視点からの異なるモデルを掲載すること

モデルについては、それを導くまでの思考過程やモデルの変化を明記すること

今回リリースするカタログは、扱っている分野・対象が少なかったり、モデルの記法が多少揃っていなかったり等々、カタログと呼ぶにはまだ質量ともに不足していますが、何とか上記 4 点については遵守できたのではないかと自負しています。ぜひ、開発の現場において活用して頂くとともに、みなさんの率直なご意見を頂ければ幸いです。

なお、今回のカタログはあくまでも「第一版」です。今後も、利用した方からのフィードバックを反映しながら、少しずつカタログの質量を充実させていく計画です。

「第二版」については、まずは「量」の充実を図りたいと考えており、組込みソフトウェアのモデリングに関わる多くの方の参加を期待しています。

ご意見やご参加のお問い合わせについては、以下の URL をご覧ください。

UMTP 組込みモデリング分科会メンバ（初版作成フェーズ：2009 年 11 月～2010 年 10 月）

あいうえお順、敬称略

石滝　智洋　（オリンパスソフトウェアテクノロジー　株式会社）

石田　晴幸　（株式会社　富士通コンピュータテクノロジーズ）

大森　淳夫　（パイオニア　株式会社）

小黒　登行　（株式会社　豆蔵）

栗田　道広　（スター精密　株式会社）

品川　正彦　（株式会社　日立インフォメーションアカデミー）

白神　一久　（富士通　株式会社）

新家　了訪　（株式会社　テイジイエル）

鈴木　茂　　（株式会社　オージス総研）

高原　大樹　（アドソル日進　株式会社）

時田　大介　（スター精密　株式会社）

西口　敦子　（NEC ラーニング　株式会社）

畑　理介　　（株式会社　オージス総研）

松嶺　恭守　（理想科学工業　株式会社）

山本　裕行　（株式会社　日立インフォメーションアカデミー）

芳村　美紀　（株式会社　エクスモーション）：副主査

渡辺　博之　（株式会社　エクスモーション）：主査

組込み分野のための UML モデルカタログ

発行日　２０１０年（平成２２年）１０月１日
発行者　UMTP, Japan
編　著　組込みモデリング分科会
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/