FUJITSU
shaping tomorrow with you

実践MBSE
～SysMLで設計意図を相伝できるか～

2014年8月28日

富士通コンピュータテクノロジーズ
組込みシステム技術統括部アーキテクチャ部
石田 晴幸

1271karch02

自己紹介

FUJITSU

- 氏名：石田 晴幸 (https://www.facebook.com/hernianrunner)
- 経歴

入社以来、ずっとファームウェア・ソフトウェア開発に従事

- 1989～1995
  - 3Dグラフィックアクセラレータ・ドライバ
- 1995～2001
  - Windows用プリンタ・ドライバ　（仕事でC++を使い始めた）
- 2001～2004
  - FOMA端末のアプリ制御　←UMLを使用、実装はC++
- 2006～
  - RAID装置内蔵のWebGUIフレームワーク
    WebGUI本体はC++、コード生成ツール等はJava
- 2011～
  - モデル駆動開発ツールBricRobo
    VDM++のコード生成を試作
- 2014 SysML認定OCSMP-Fund取得

UMTP L3 MODELER

2009～
UMLモデリング推進協会で、
組込みモデルカタログ執筆

本日のメインテーマ

FUJITSU

- 設計書を読んでも、その設計意図は伝わらないことがある
- 設計意図がわからないと変更できない
- うまく変更できないと機能拡張に支障が出る
- 昔、頑張って作った人が定年になったとき、
  ソースコードだけで技術を継承できるのか

- UML/SysMLで書いたモデルは、この問題を解決できるか？

FUJITSU

開発現場にありがちな問題

上流設計の意図通り実装されない

FUJITSU

システムテストで障害を検出しました！

なぜ、今までわからなかった？

相手のモジュールは別部署で作ってました！

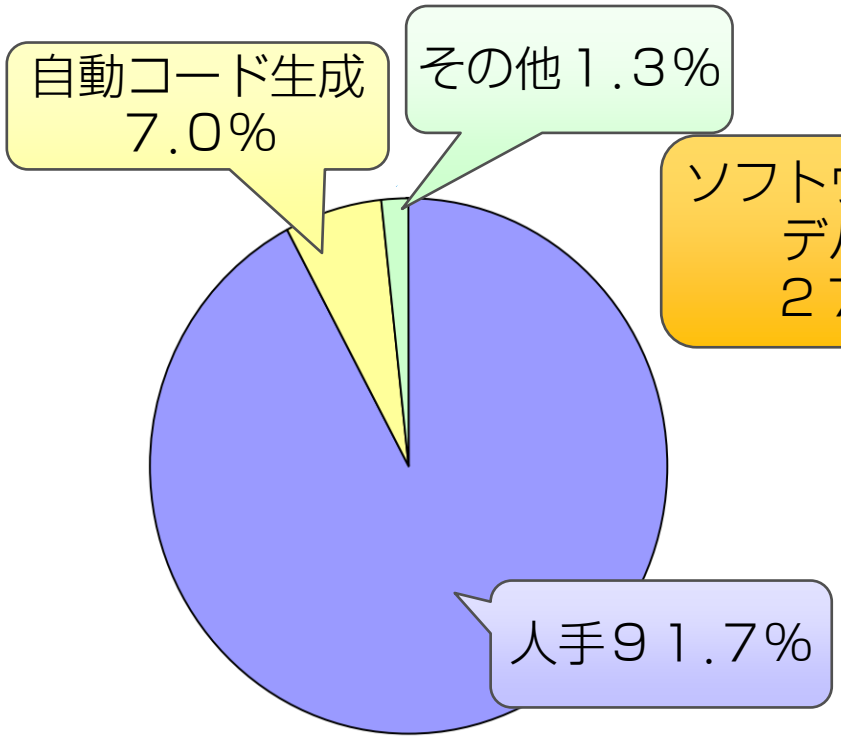
そもそも、どうあるべきなんだっけ？

・・・

他人の書いた資料を理解するのは大変です
上流の資料ほど内容は抽象的で、解釈にブレが出やすいものです

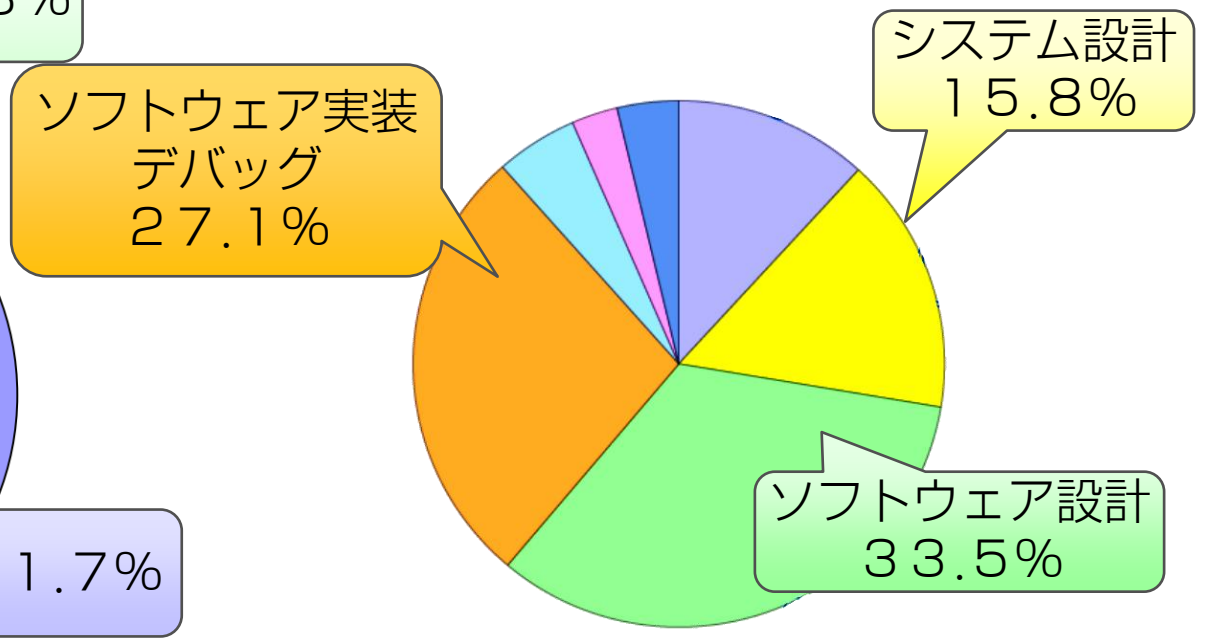
実装工程で混入する障害

９１.７%のソースコードが人手により作られた
２７.１%の不具合がソフトウェア実装で作りこまれた



新規ソフトウェア開発方法の比率

IPA SEC 田丸喜一郎(2013)
「ソフトウェア産業の実態把握に関する調査」による組込み産業の現状 より



不具合の原因工程の比率

IPA SEC 村松昭男(2013)
IPA/SECが提案するソフトウェア品質指標の解説 より

半分でもコーディングを減らせないかなぁ？

それって最適設計なのですか？

FUJITSU

よくある話

他社より性能の高い装置なのに、あまり売れないなぁ

成功した人の話

iPodはソフトウェアだ

by スティーブ・ジョブズ

iPod本体は音楽再生機器
iTunes　というダウンロードソフトウェア
iTunesストア という販売経路
これらが組み合わさって、iPodを構成している

NHKスペシャル メイド　イン　ジャパン　逆襲のシナリオより

仕様変更です

FUJITSU

To:開発者 殿
機能ｘの仕様を変更します。
明日までに影響範囲と対応工数を
教えてください。

トレーサビリティが確保されていれば、
仕様変更による影響範囲はわかります

要件から設計までトレーサビリティを確保することは大変
やるからには、
- 少しでも楽にやりたい
- トレーサビリティの情報を活用したいですね

FUJITSU

モデル

モデルとは

FUJITSU

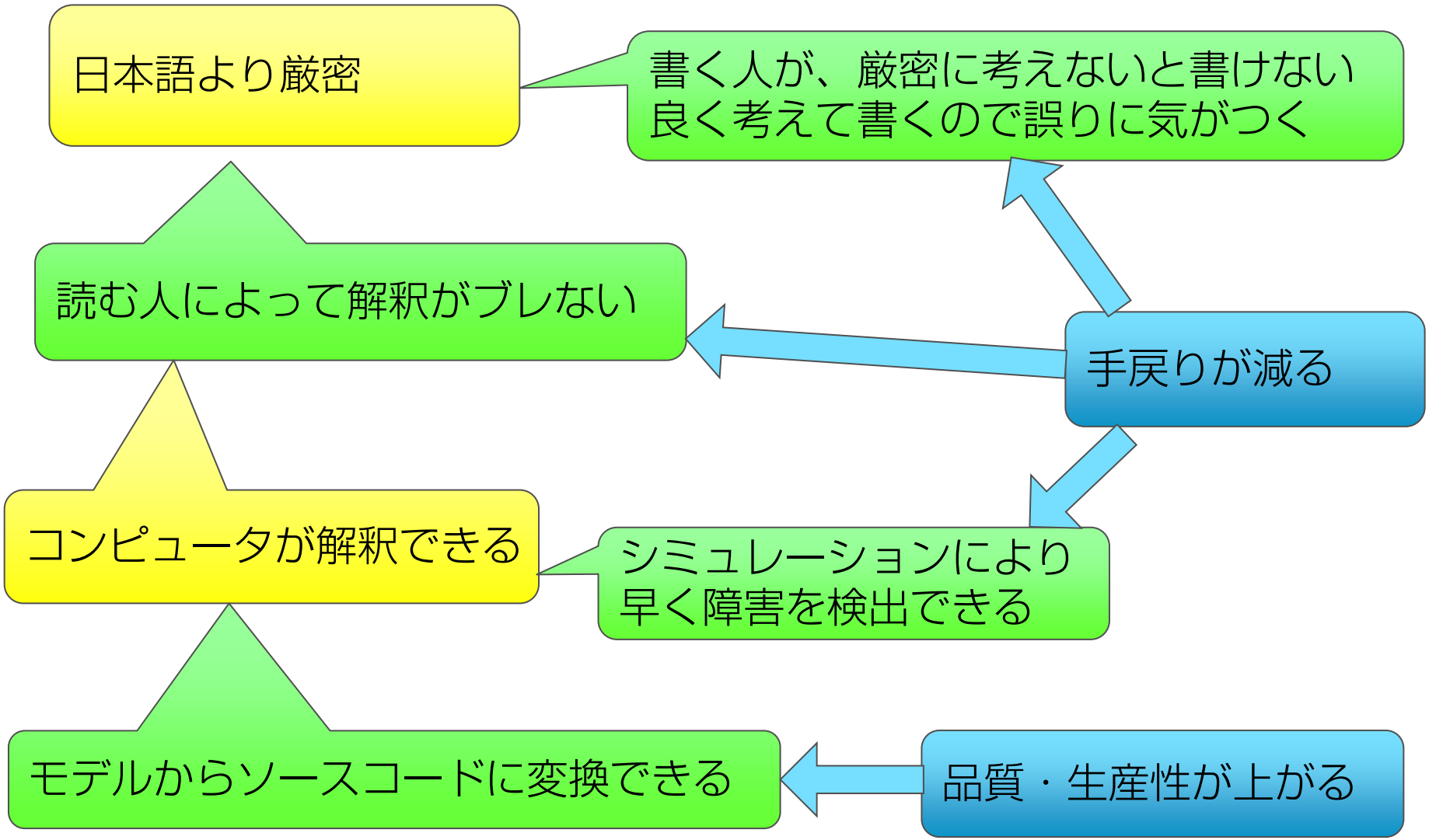
- モデル
  ある事象について、
  諸要素とそれら相互の関係を定式化して表したもの

  デジタル大辞泉より

  「モデル」には、手本・見本という意味がありますが、
  ここではその意味では用いません

- モデルの例
  - UMLやSysML　システムの構造と振る舞いを図で示す
  - MATLAB/Simulink　制御の数式を図で示す
  - VDM　ライトウェイトな形式手法の代表格
  - SPIN（Promela言語）　もともとはプロトコルの検証用
    モデル検査の代表格
  - 数式　モデルの原点
  - etc...

モデルを使うメリット



読む人によって解釈がブレない

FUJITSU

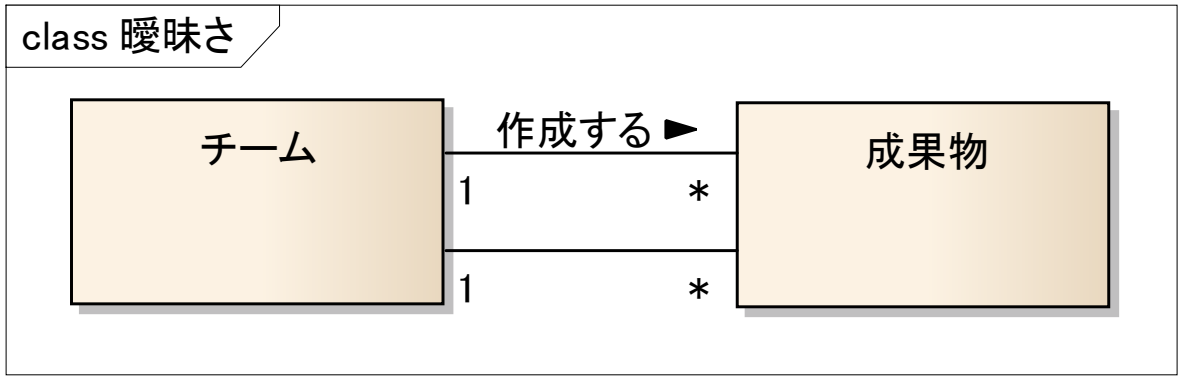
■ あいまいな日本語の例

「各チームが作成する成果物をレビューする。」

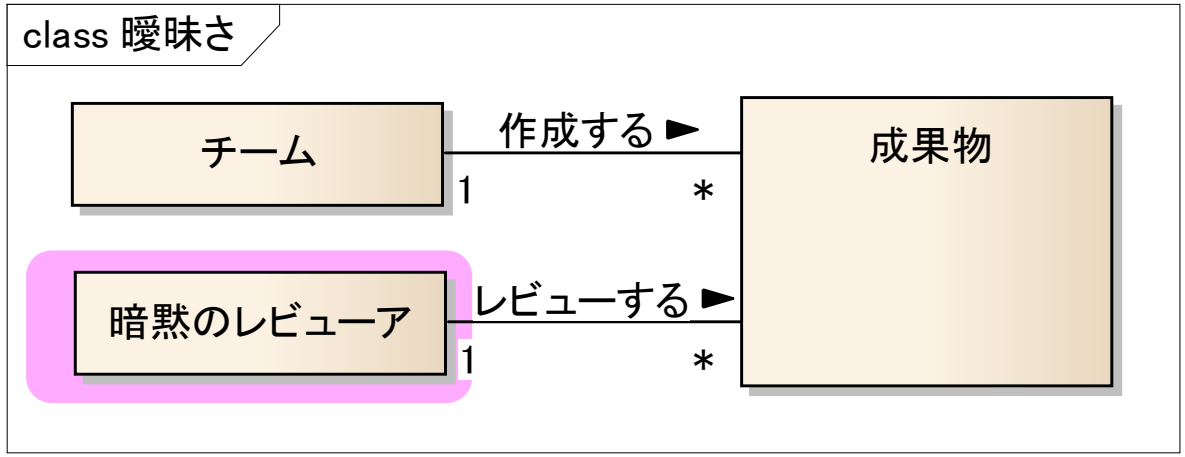
※日経SYSTEMS 2012.04より

解釈の例

例1

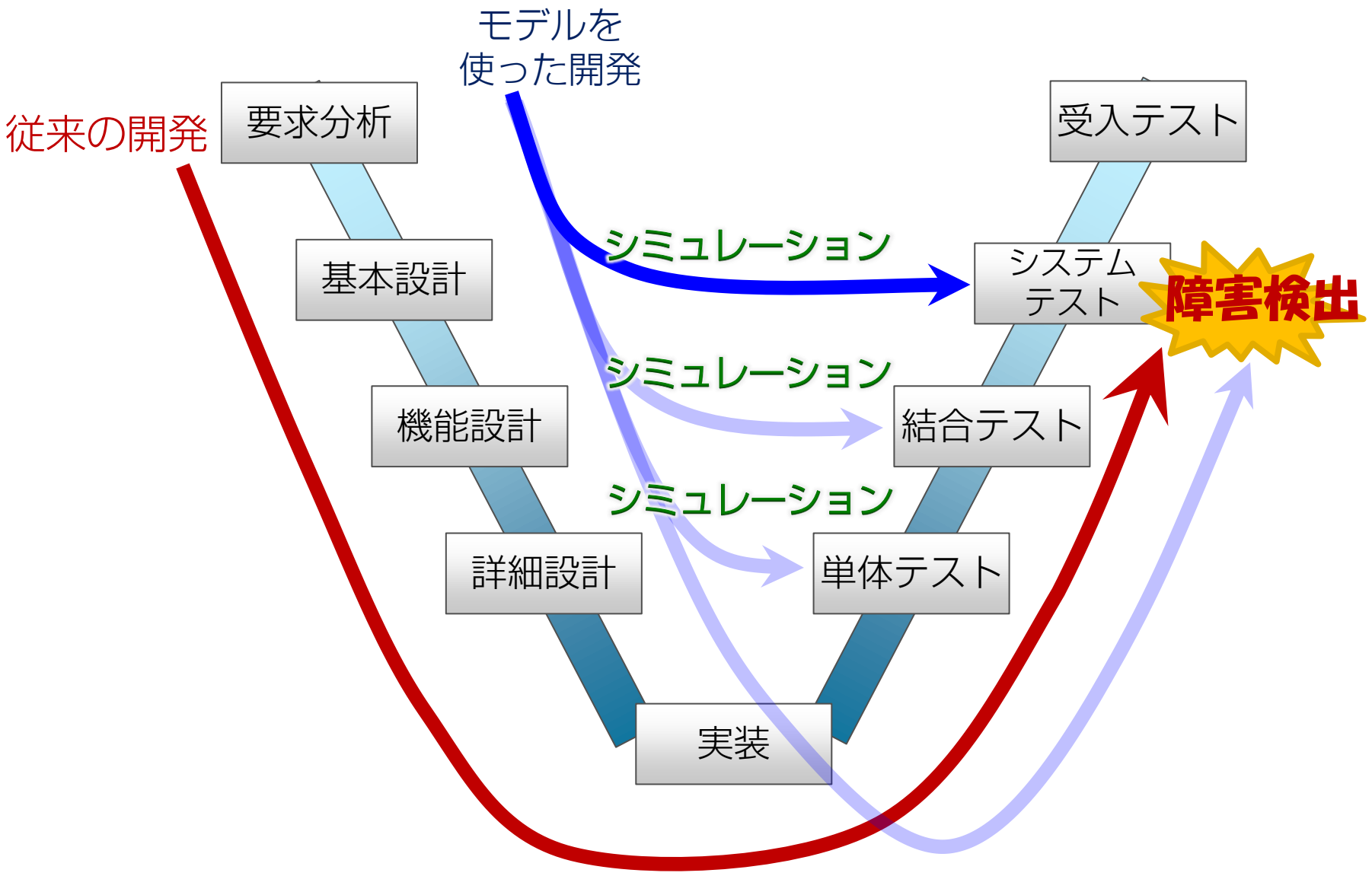


例2



要素や多重度をいじると、
さまざまな解釈の仕方を
生み出すことができます

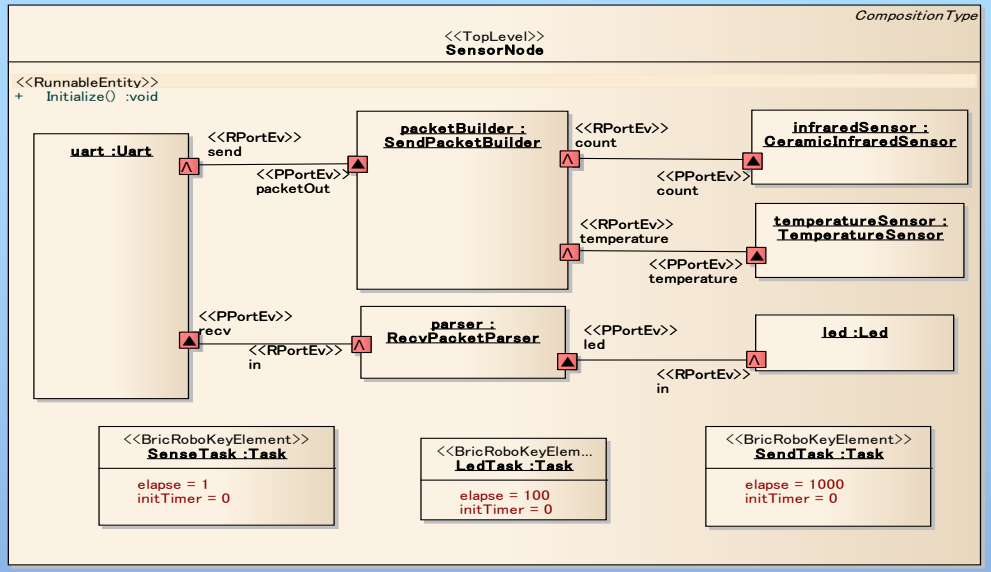
シミュレーションにより早く障害を検出できる

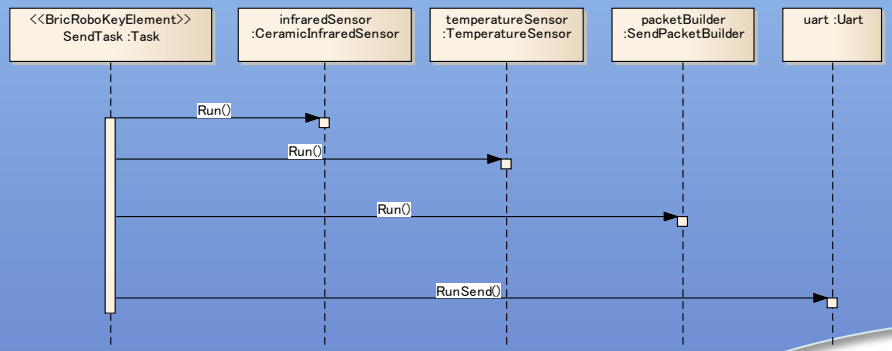


モデルからソースコードに変換できる

FUJITSU

BricRoboモデル





簡単に始められる！
組込みソフト向けモデル駆動開発ツール

BricRobo

BricRobo
コードジェネレータ

ソース
コード

設計と実装が
乖離しません

FUJITSU Embedded System BricRobo

FujitsuBricRobo 検索

FUJITSU

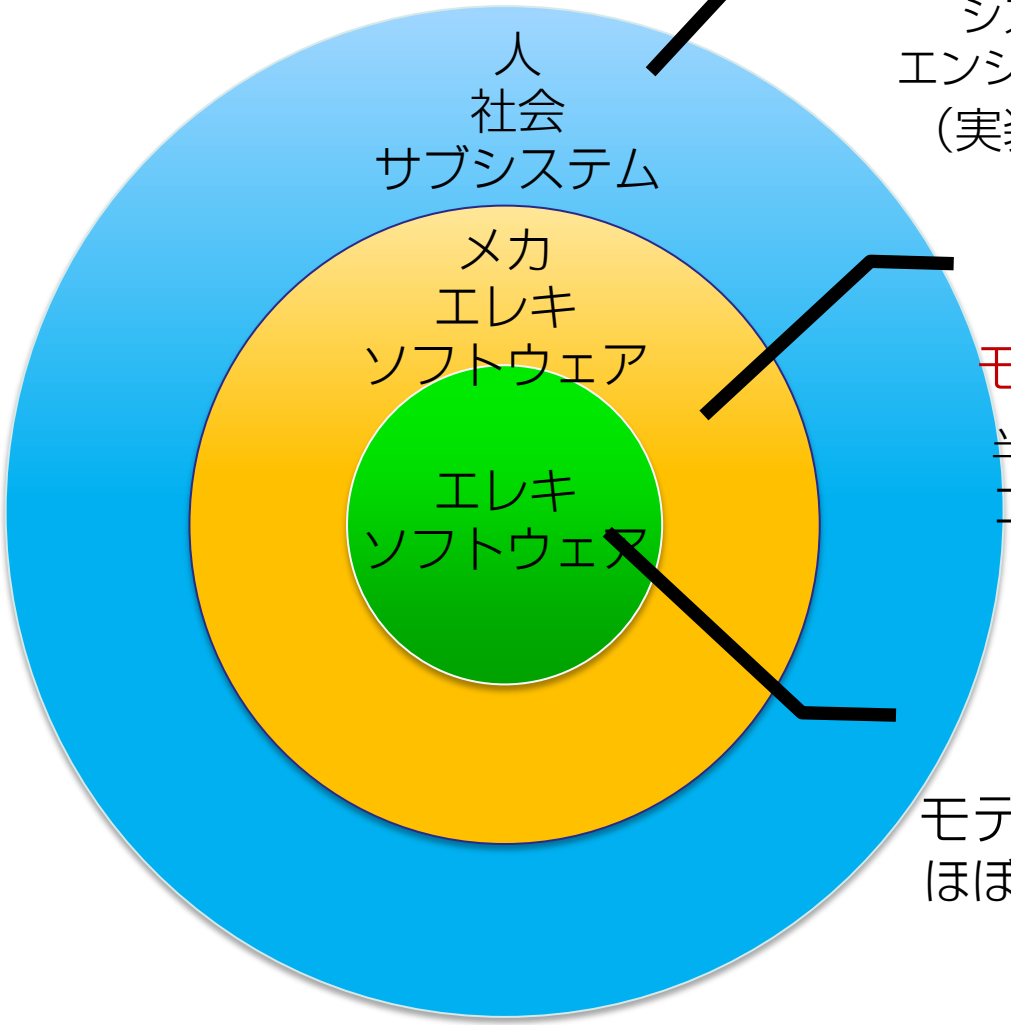
モデル駆動開発

モデル駆動開発

FUJITSU

MBSE、MDD、MBD等、よく似た言葉があります
ここでは、以下の観点で定義します

- 扱う領域
- 自動コード生成

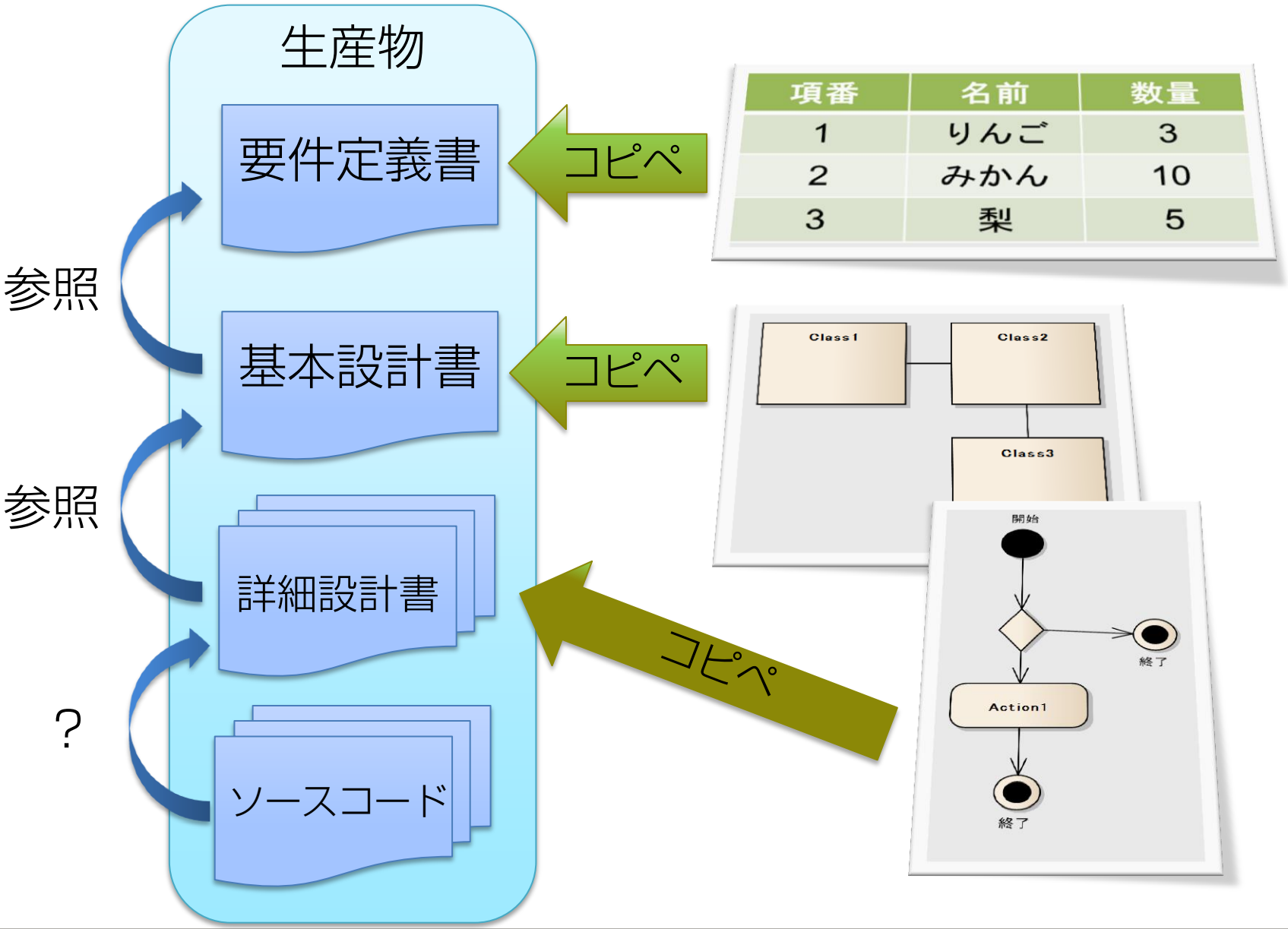


MBSE
モデルベース
システムズ
エンジニアリング
（実装しない？）

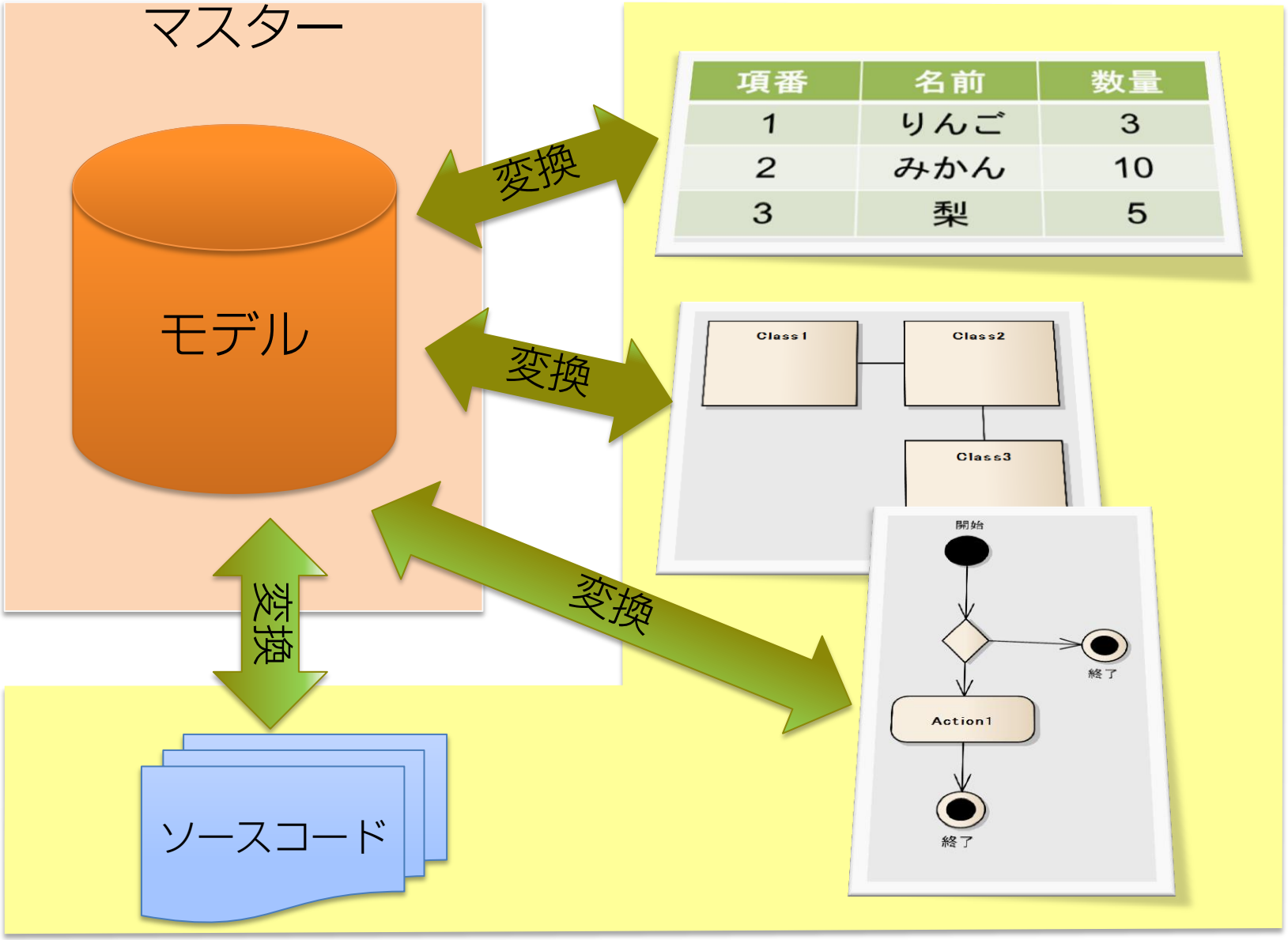
MDD
モデル駆動開発
半自動＋ハンド
コードディング

MBD
モデルベース開発
ほぼ自動コード生成

従来はドキュメント駆動開発



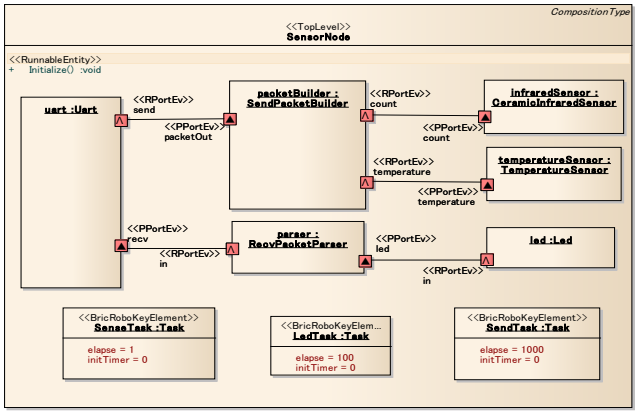
モデル駆動開発はモデルが原本

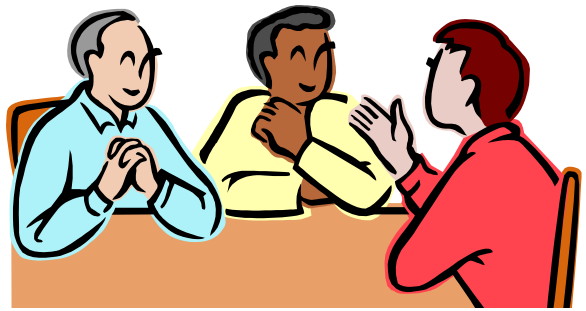


多分野の専門家による議論

- メカ・エレキ・ソフトなど多分野の専門家が議論できる
  - グラフィカルな記法で直感的に理解
  - 全体から詳細まで、任意のスコープでシステムを俯瞰

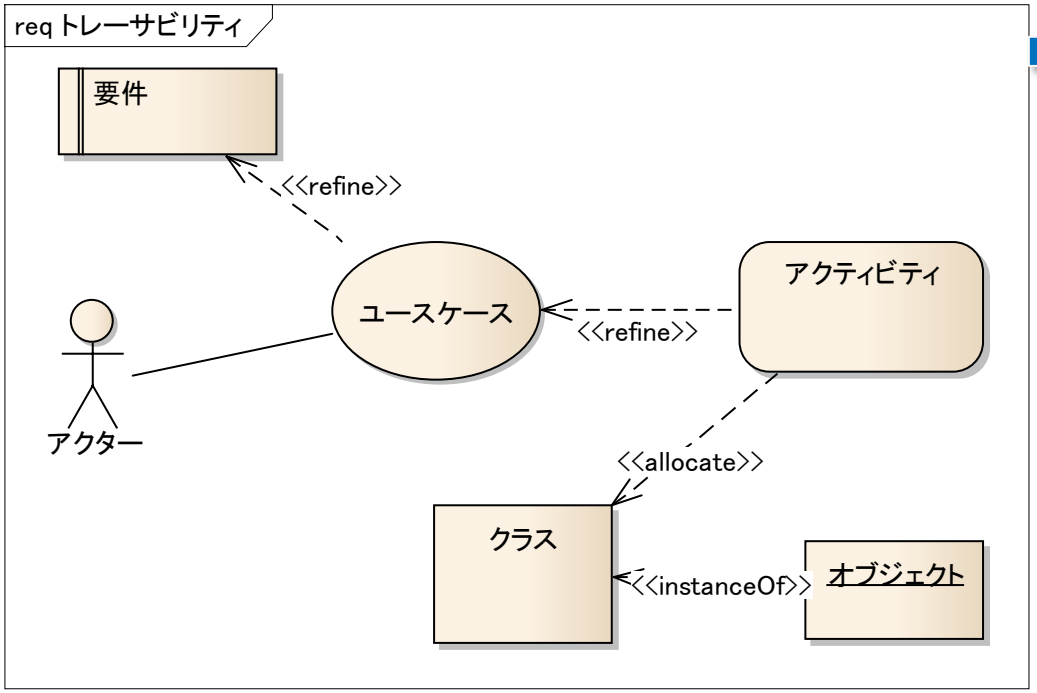
```
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;
using Ionic.Zip;

namespace WebMasterLogCollector
{
    class LogFilter : IDisposable
    {
        private const string ZIP_FILE_NAME_TEMPLATE = "{0:0000}{1:00}{2:00}.zip";

        private Regex _regexHttpUrl = new Regex("^.*?¥"¥¥S+¥¥s+(¥¥S*?)¥¥s");
        private TextWriter _writer;

        public LogFilter(string logFile)
        {
            string logDir = Path.GetDirectoryName(logFile);
            if (Directory.Exists(logDir) == false)
            {
                Directory.CreateDirectory(logDir);
            }
            _writer = new StreamWriter(logFile, false, Encoding.UTF8);
        }

        public void Filter(TextReader reader, Regex urlFilter)
        {
            string line;
            while ((line = reader.ReadLine()) != null)
            {
                Match m = _regexHttpUrl.Match(line);
```



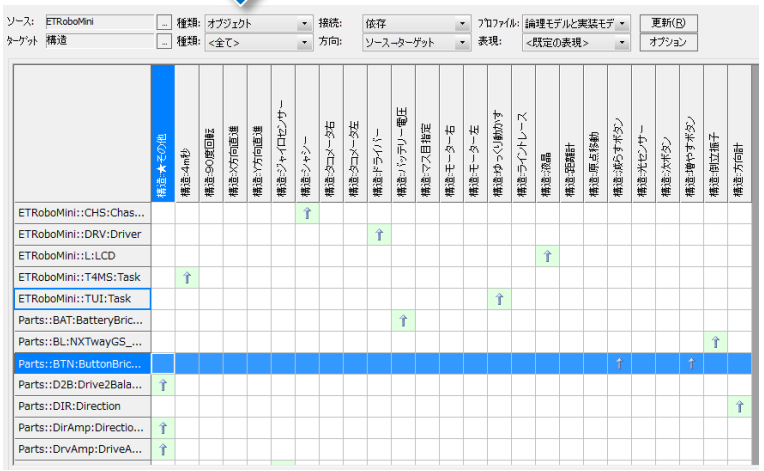


トレーサビリティ

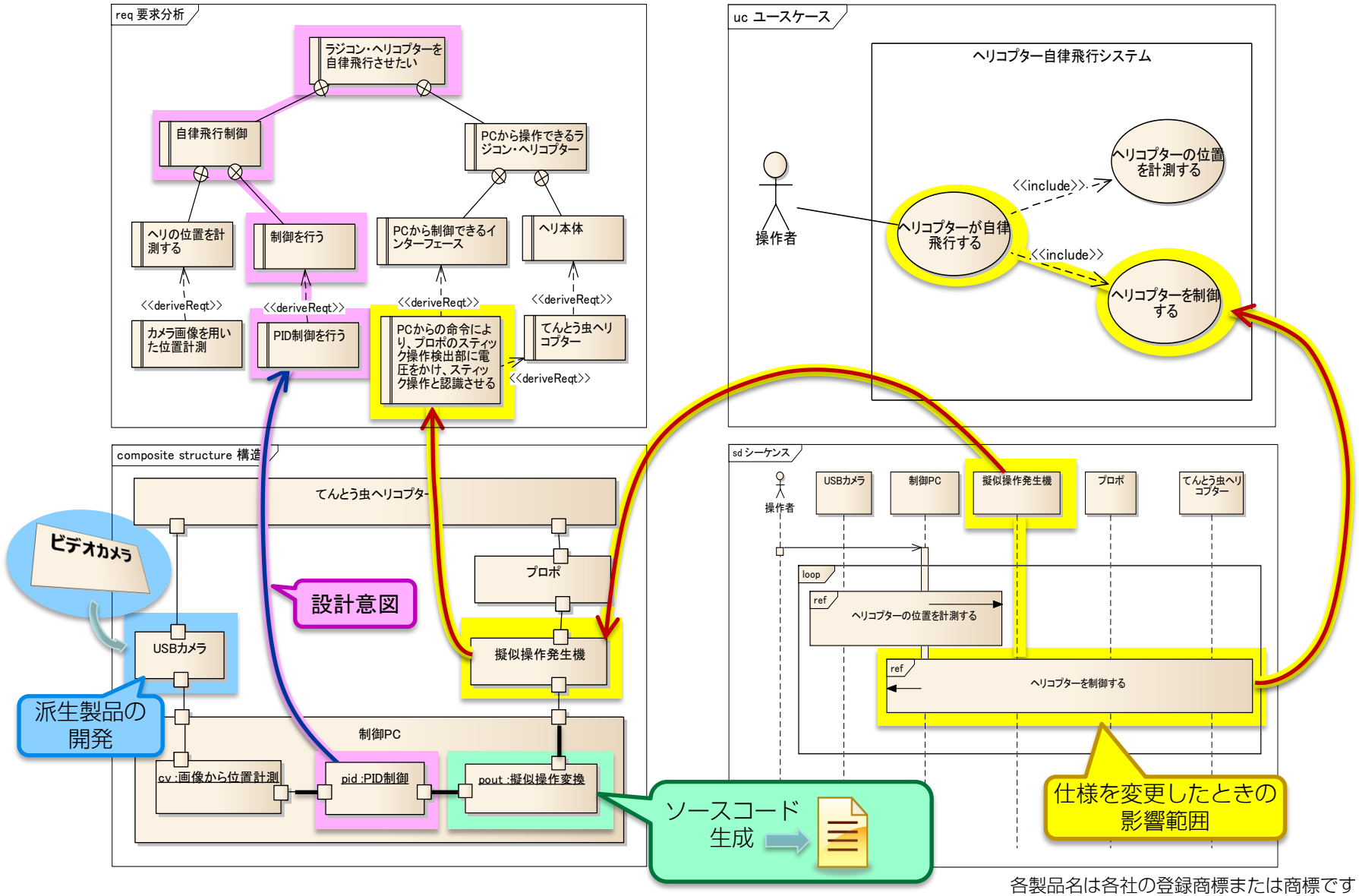
- トレーサビリティ
  - 要件が漏れ無く、どのように設計され、どこに実装されたのか
  - それらの設計・実装がどのようにテストされたのか
  - 第三者にわかる形で説明できるか
- SysMLなら、関連線を使ってトレーサビリティを確保できる



マトリックスに変換して見ることが可能



モデル駆動開発のイメージ

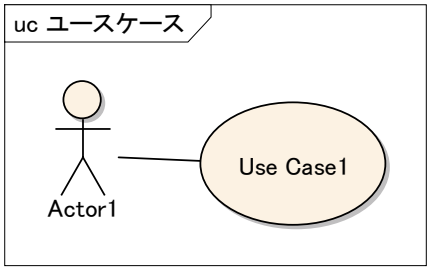


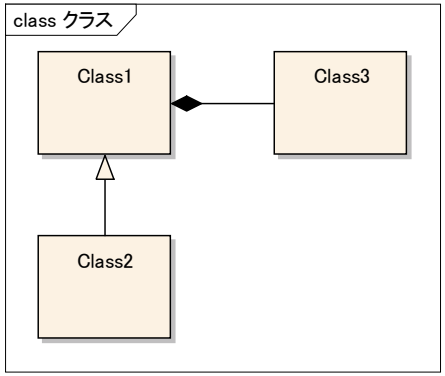
FUJITSU

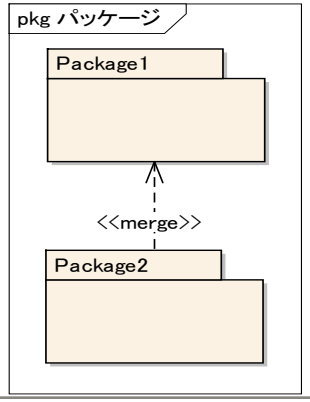
UMLとSysML

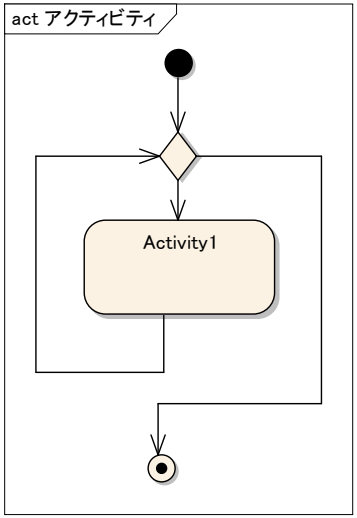
UML (Unified Modeling Language)

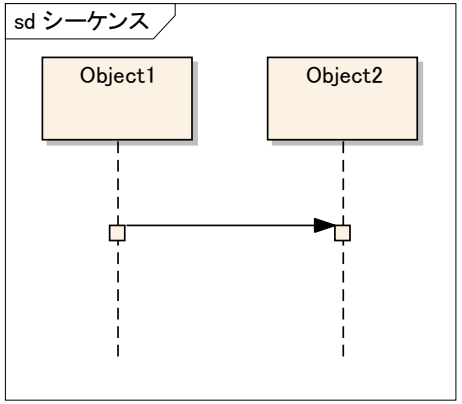
- OMGが策定した、
  ソフトウェア・オブジェクト指向設計のための表記方法
  初版：UML 1.1 (1997)、 最新: UML 2.4.1(2011)
- 矩形や線を使って構造・振る舞いを表す、9種類の図
- UMLツール間でデータを交換するための、XMLデータ形式
- OMG (Open Modeling Group)
  - 国際的な、非営利的なコンピュータ業界の標準策定コンソーシアム





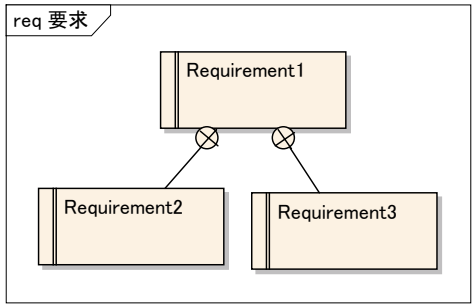


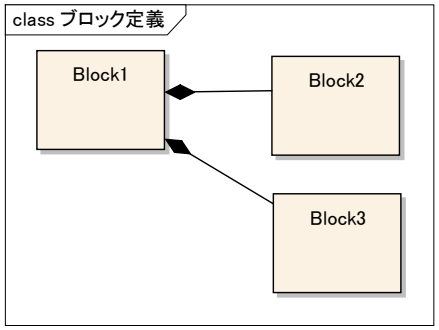


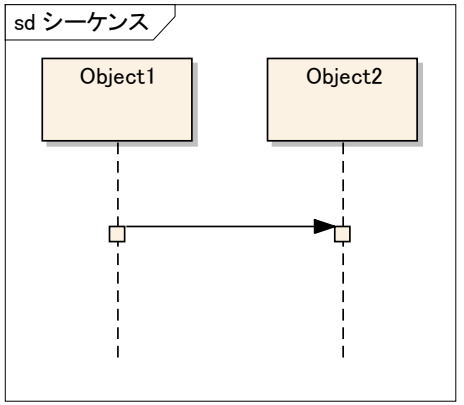


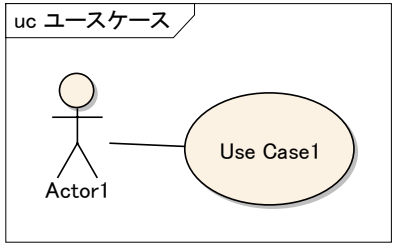
SysML (System Modeling Language)

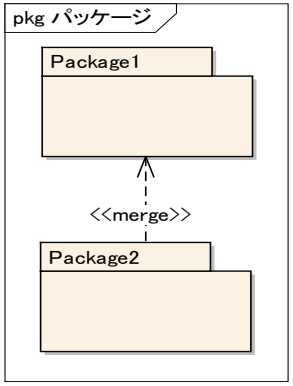
- INCOSEがUMLをシステムエンジニアリング向けにカスタマイズ後にOMGとINCOSEが共同して策定
  初版：SysML 1.0 (2006)　最新：SysML 1.3 (2012)
- 矩形や線を使って要求・構造・振る舞いを示す、9種類の図
- INCOSE (The International Council on System Engineering)
  - システムエンジニアリングの非営利業界団体

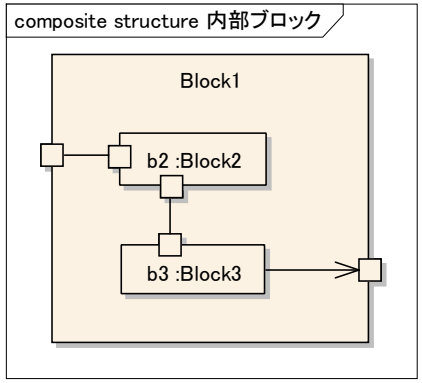








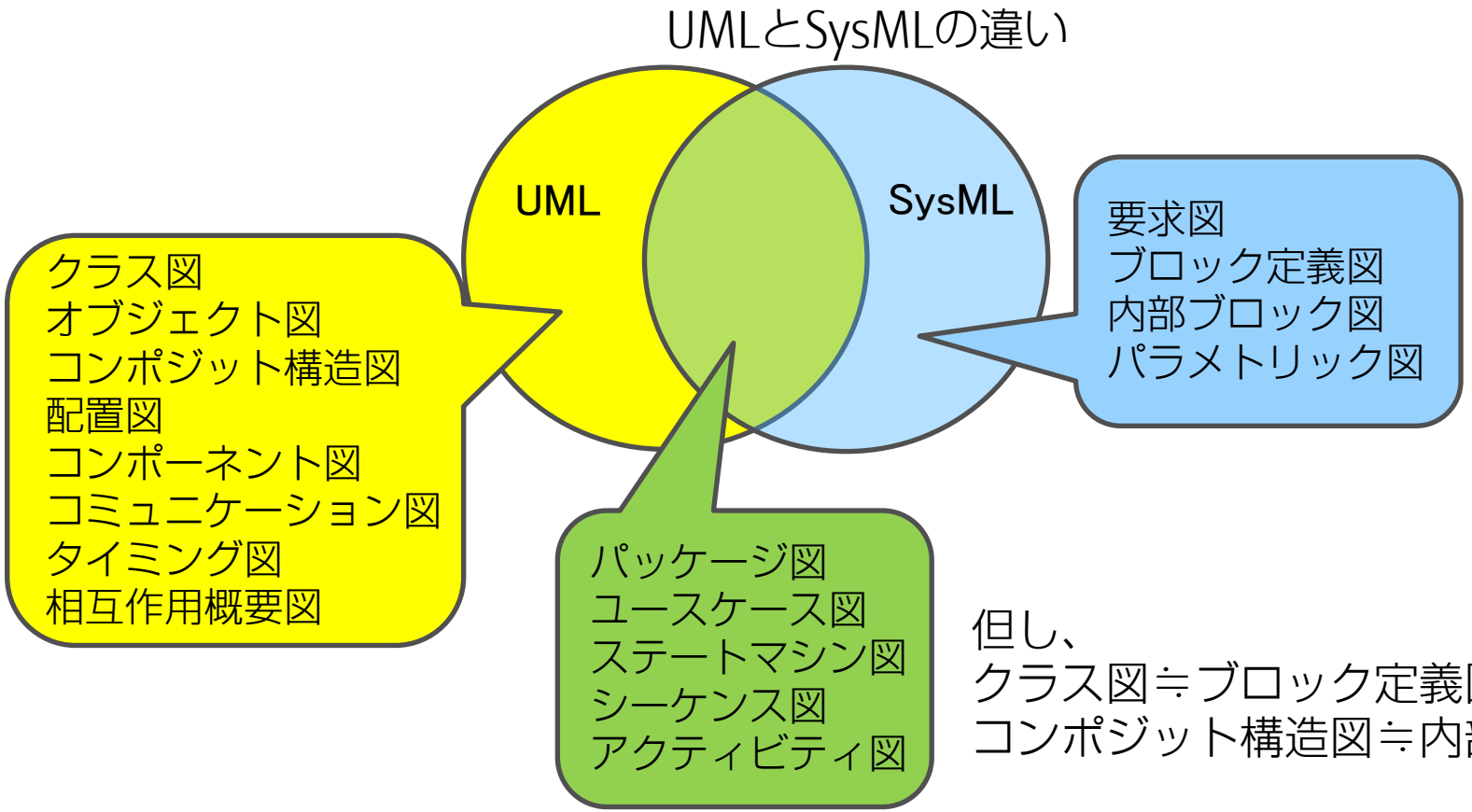




UMLとSysMLの違い

FUJITSU

- SysML
  - OMGの策定したシステムモデリング用言語
  - UMLをベースにして策定



但し、
クラス図≒ブロック定義図
コンポジット構造図≒内部ブロック図

なぜSysMLが求められたのか

FUJITSU

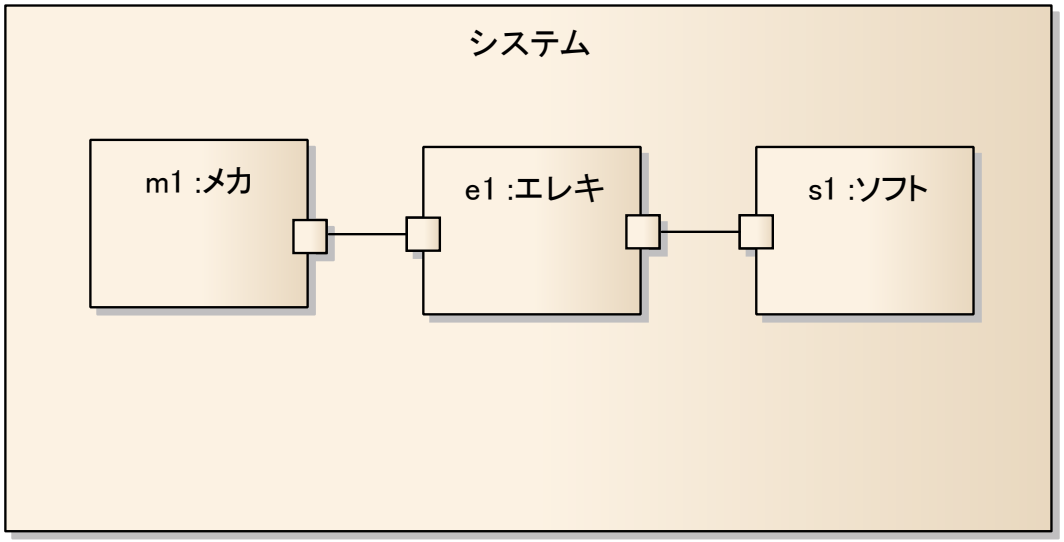
UMLの3大弱点　※石田の主観

- UMLはソフトウェアに寄り過ぎた
- UMLは部品化ができなかった
- UMLで示したシステムで、要件が充足されたか不明だった

UMLが使い物にならないわけではありません
SysMLの利点を知っていれば、
これらのUMLの弱点をうまく回避することも可能でしょう

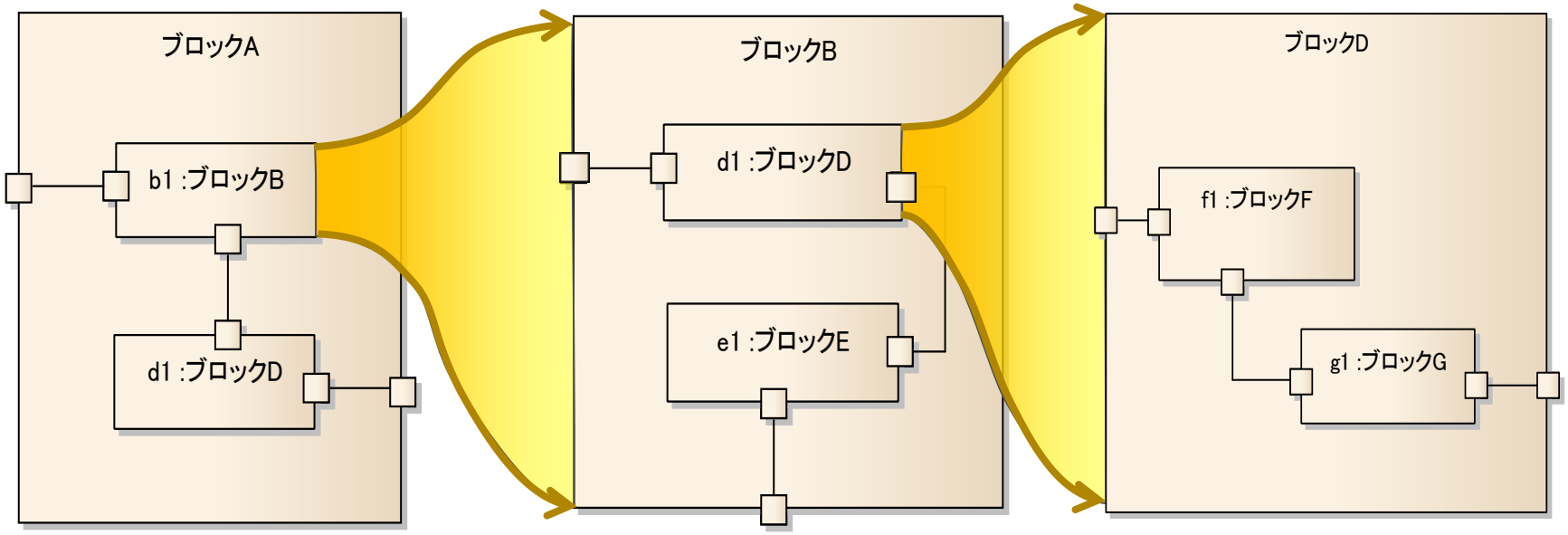
UMLはソフトウェアに寄り過ぎた

UML	ソフトウェアのオブジェクト指向設計を表すのに重宝された
	クラスを格納するコンポーネントがあり、 コンポーネントをサーバー等に配置するための配置図があった
SysML	システムの構造はブロック（定義）とパート（用法）によって示す
	メカ・エレキ・ソフトを同じ表記で扱い、 多分野の専門家が議論する土台にできる



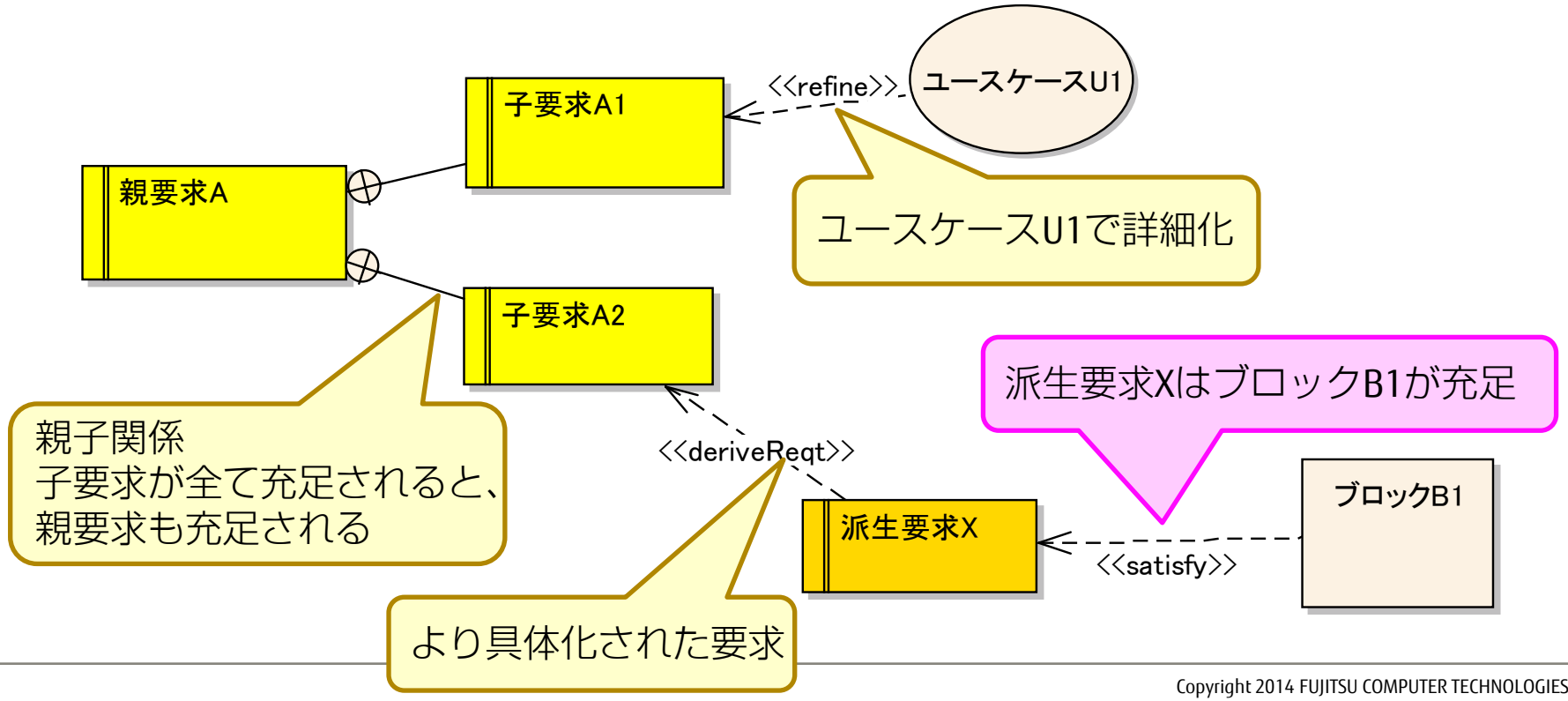
UMLは部品化ができなかった

UML	クラス図が巨大な図になりがちだった
	クラス図では再利用可能なクラスの塊を明確に表現できなかった
SysML	ブロック定義図と内部ブロック図を組み合わせて、何層でも掘り下げられる
	部品と部品を接続できる接続点として、ポートの使い方が明確になった



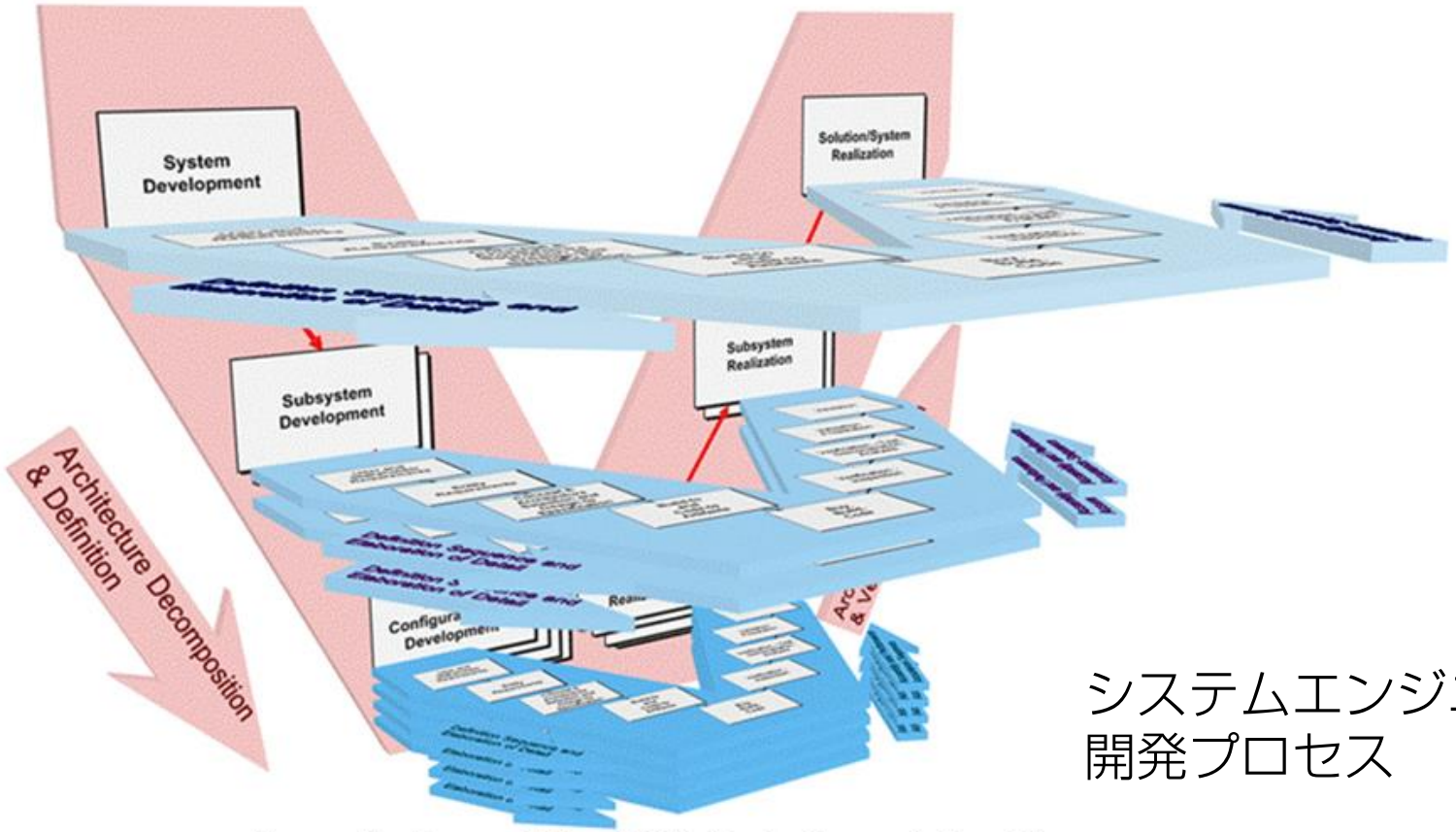
UMLで示したシステムで、要件が充足されたか不明だった

UML	そもそも要件を描く正式な記法がない
	要件の充足性を示す記法もない
SysML	要件図で要件の関係性を示せる 親子関係に限らない
	要件がどのように充足されたかを示す関連線がある



システム・エンジニアリング・プロセス

Dual Vee

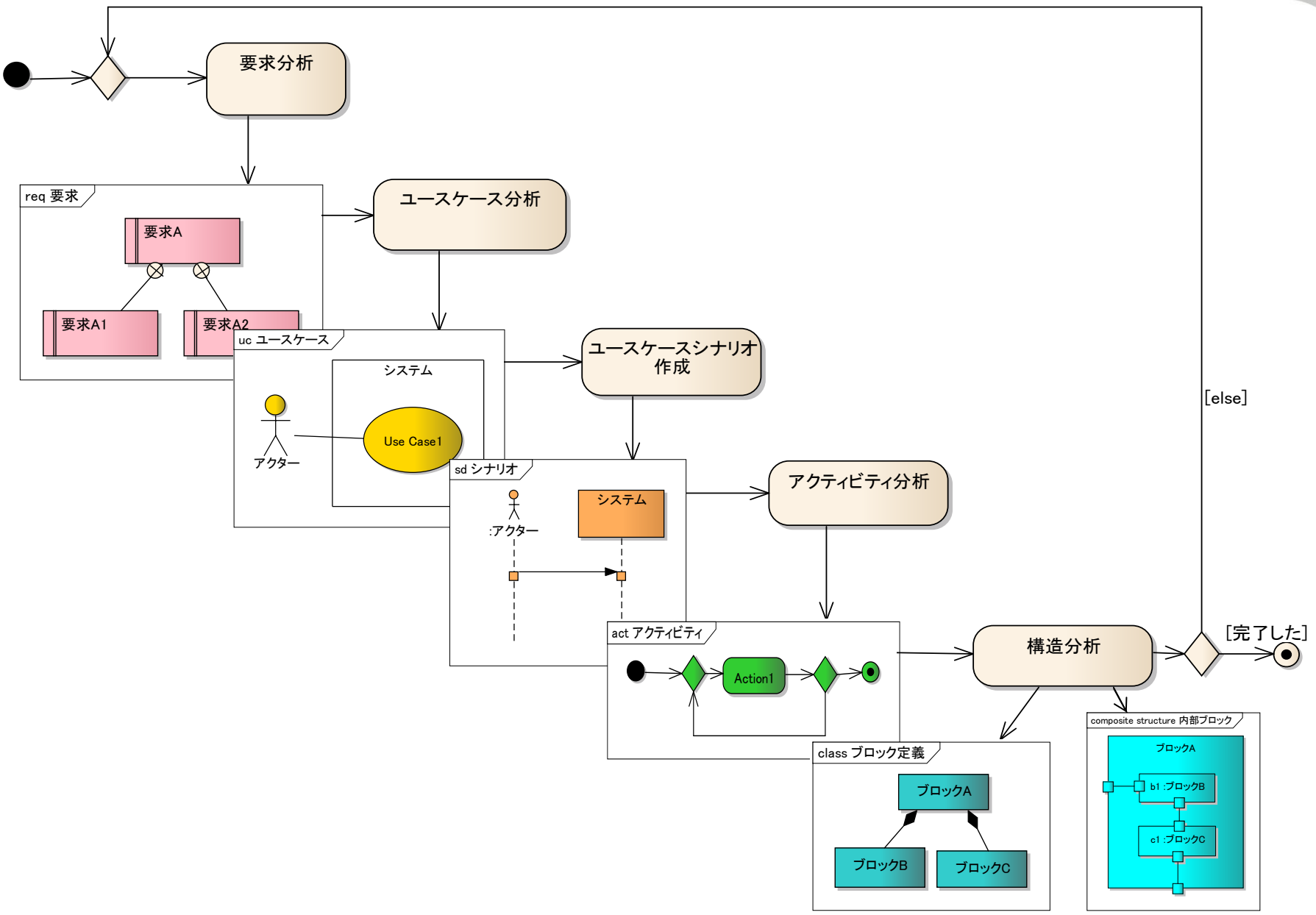


Source: Forsberg and Mooz (2006). Used with permission of the authors under the Creative Commons Attribution 3.0 License..

システムエンジニアリングの開発プロセス

- ピンクのV字　アーキテクチャ観点からシステムを分割するプロセス
- 水色のV字　　各システム要素を設計・実装・評価するプロセス
- 二つのV字が直交している

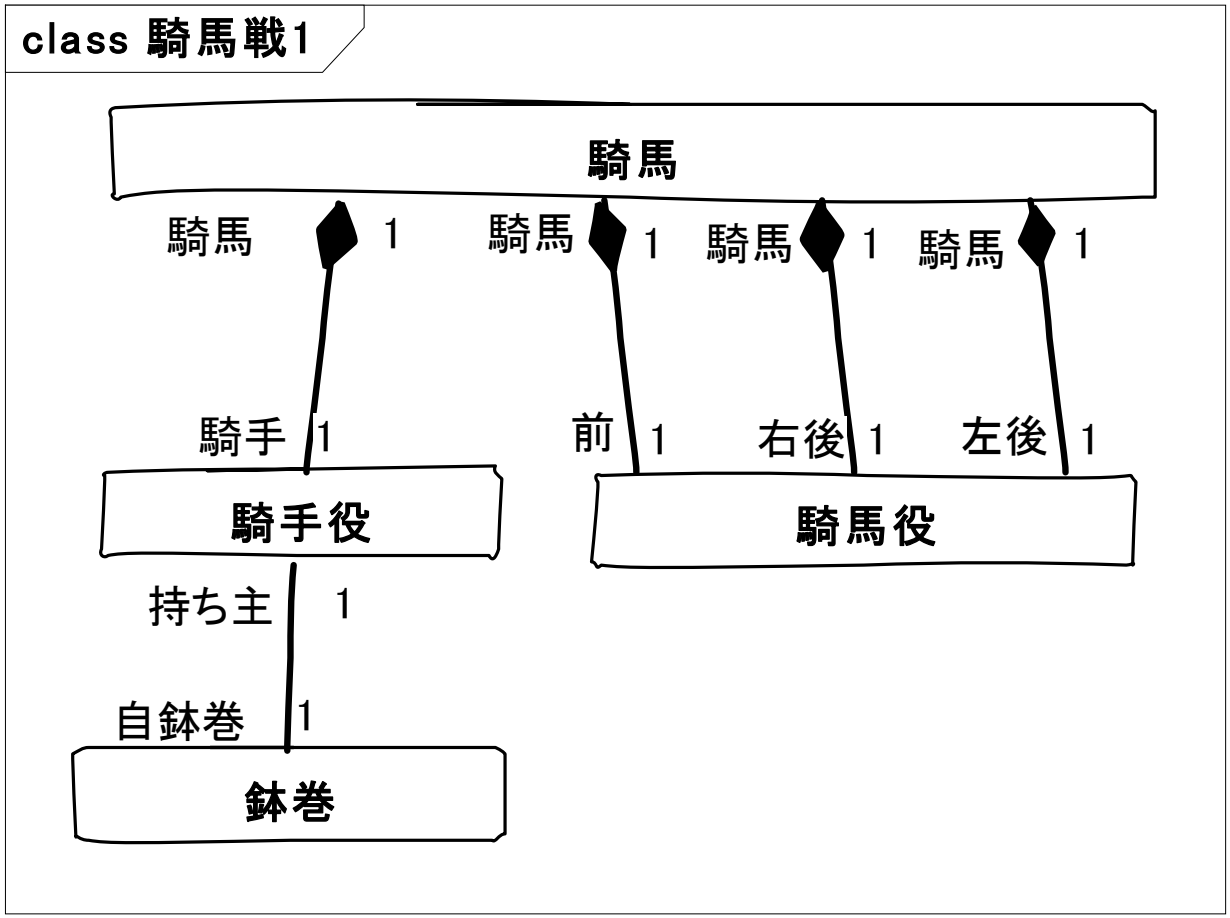
要求から構造を抽出するプロセス例



FUJITSU

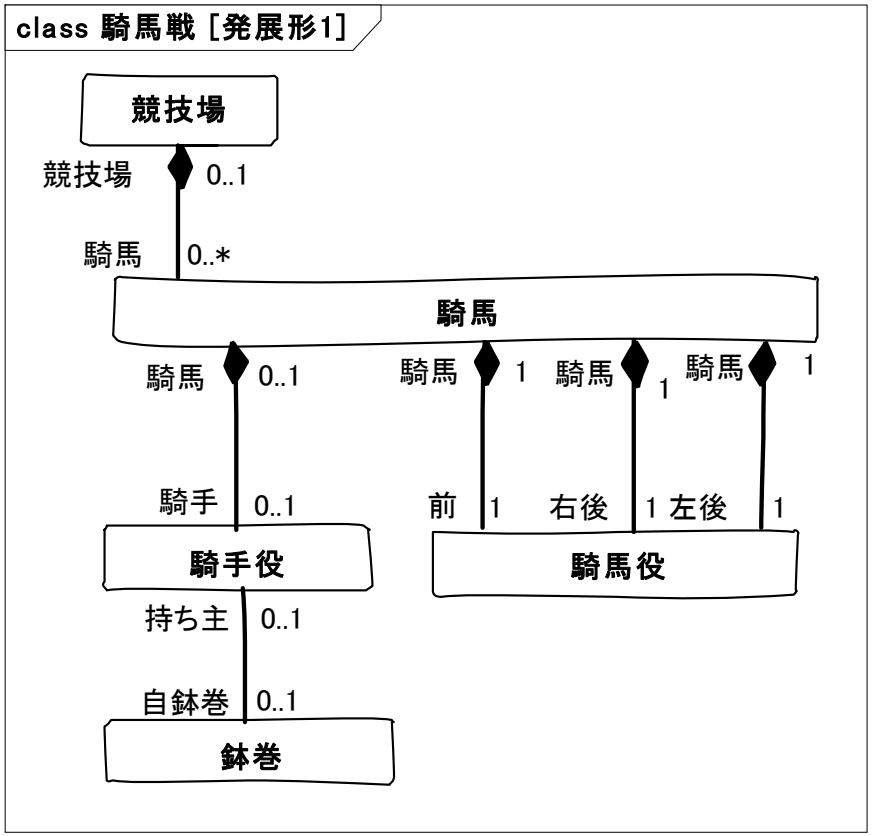
モデルに意図を込める#1
騎馬戦

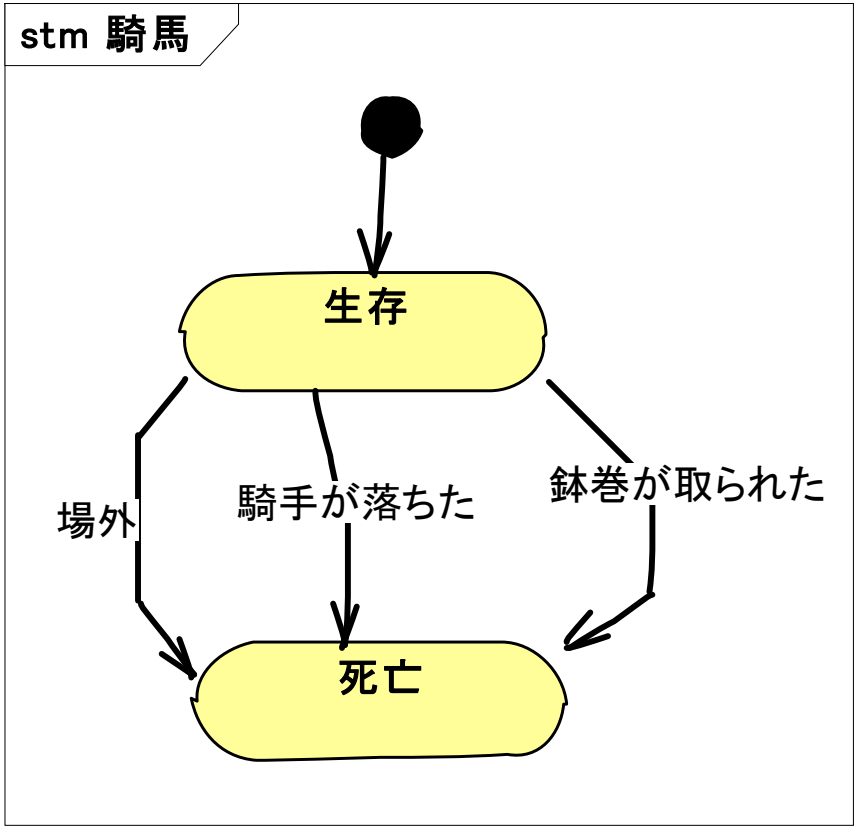
運動会の騎馬戦をモデリングしてみる



- この図は誤りではないが「騎馬戦」とは言いがたい
- 何かが欠けている

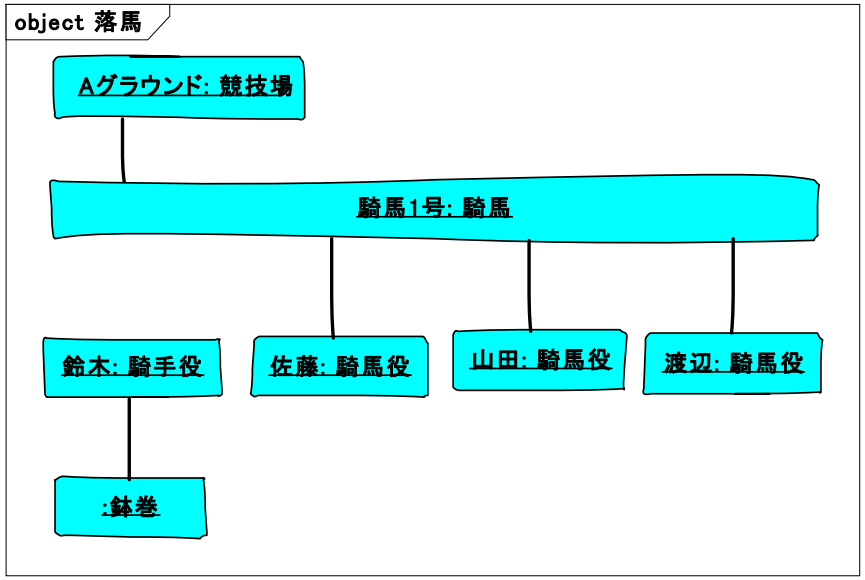
騎馬戦（発展形1）

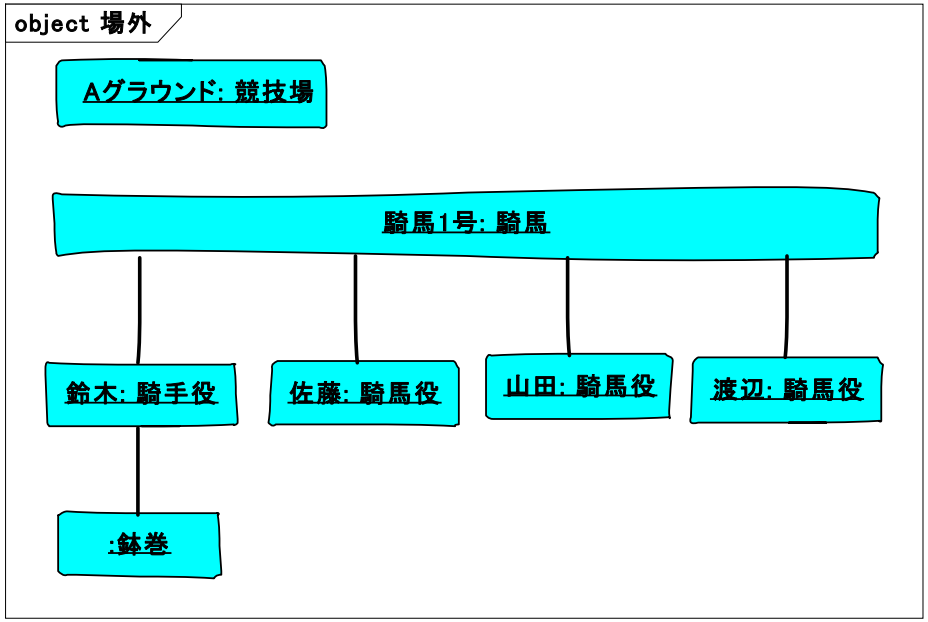


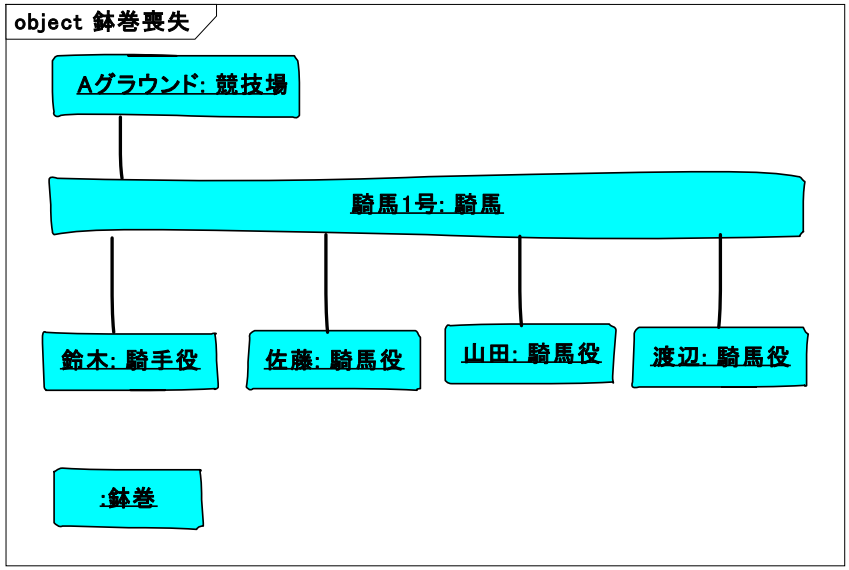


- 騎馬について掘り下げてみたもの
  - 騎手が騎馬から落ちるかもしれない
  - 騎手の鉢巻を取られるかもしれない
  - 競技場の外に出るかもしれない

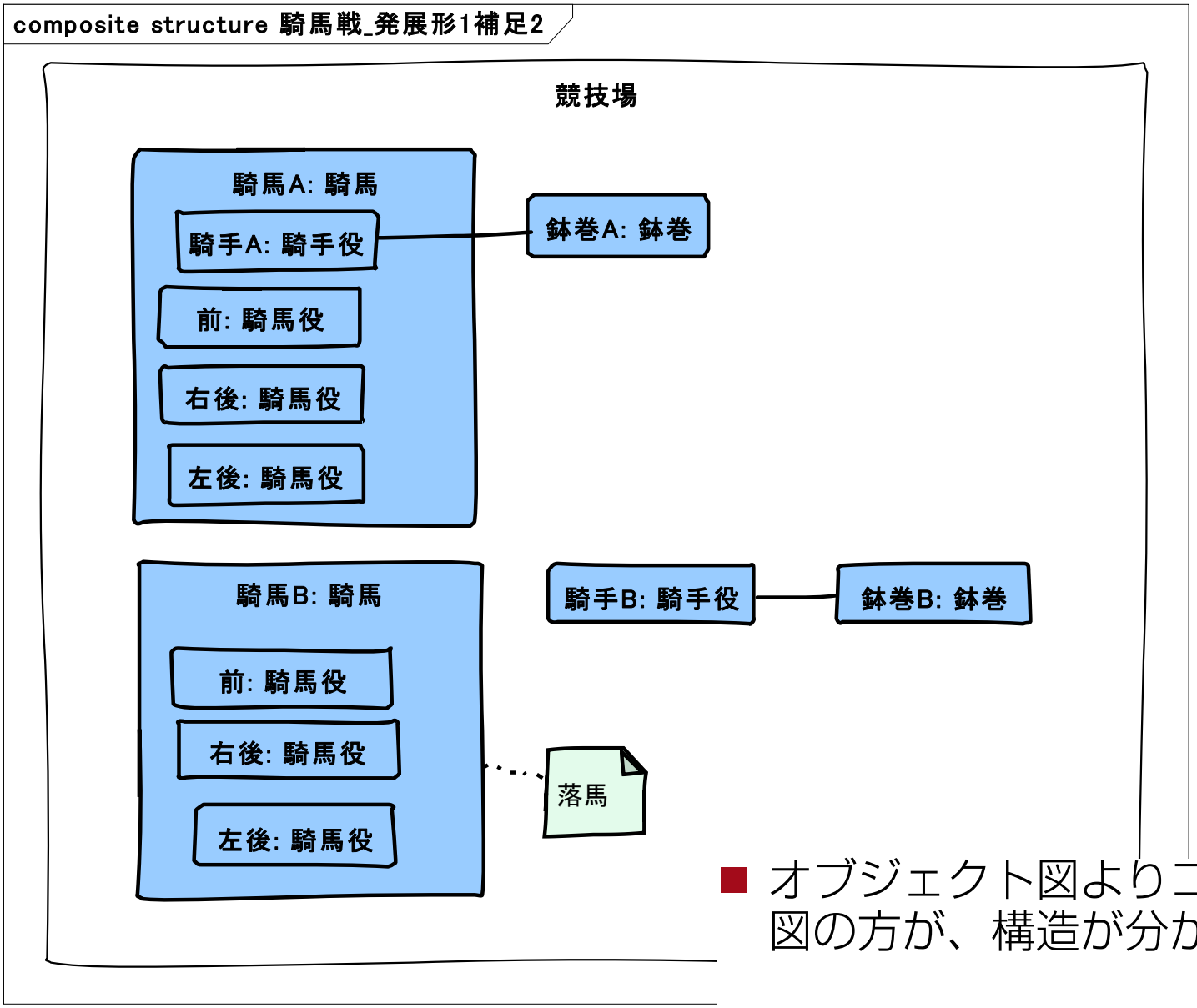
騎馬戦（発展形1補足）





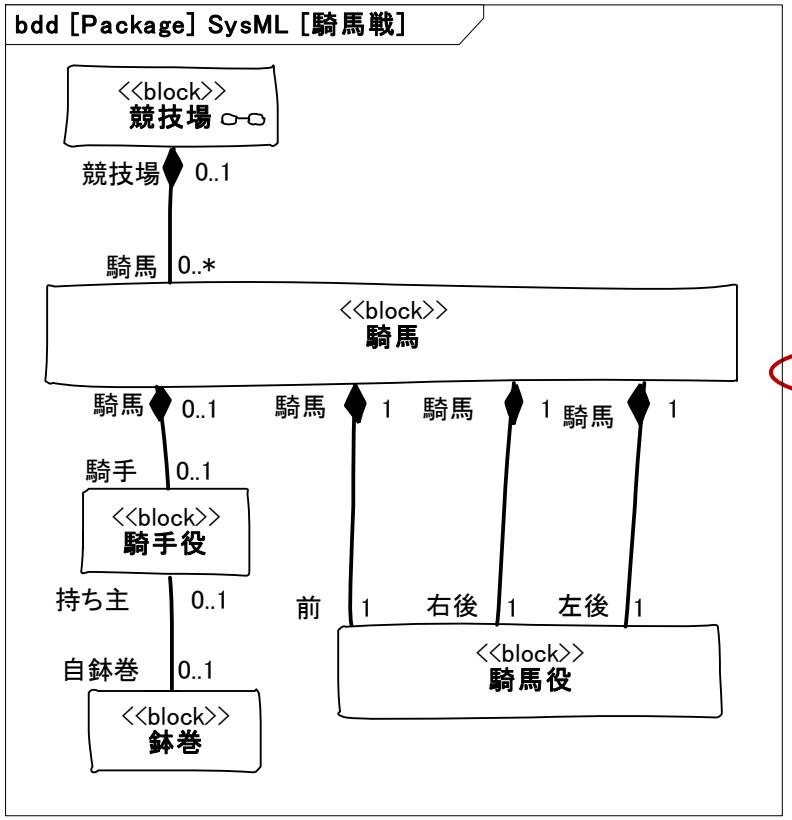


騎馬戦（発展型1補足2）



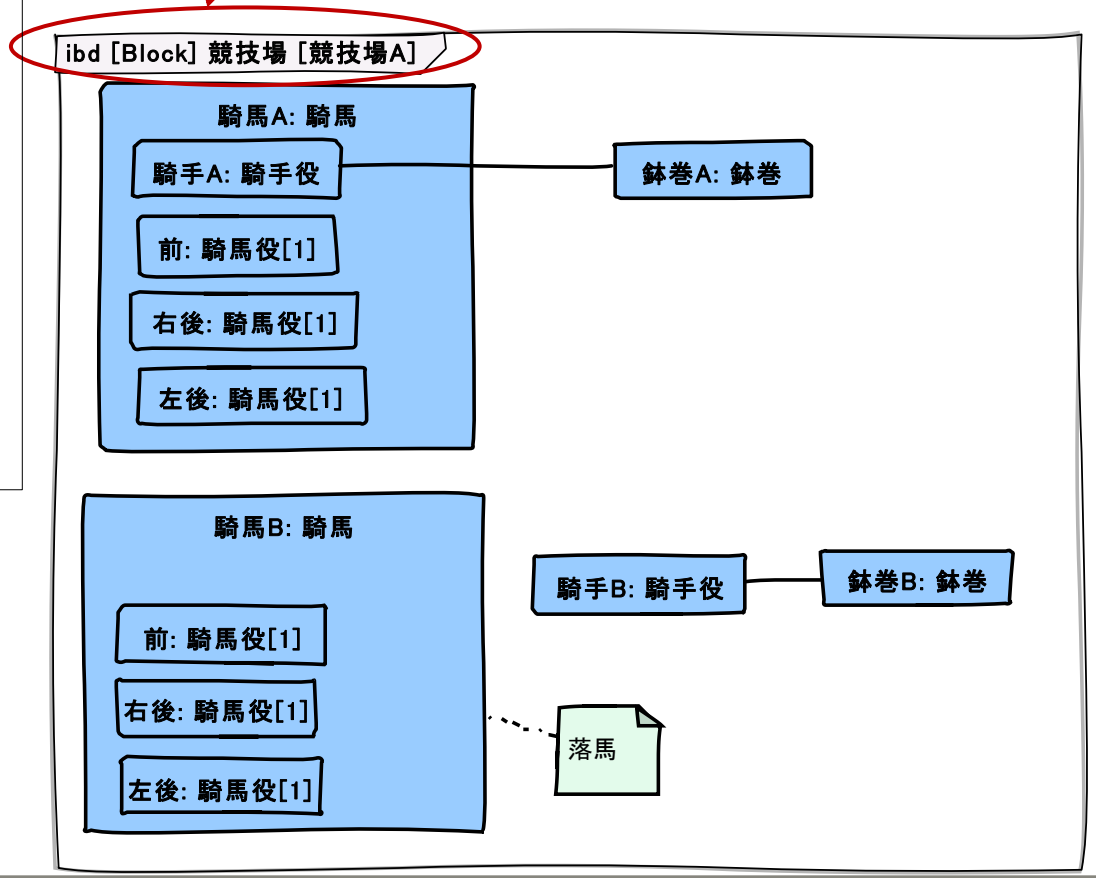
- オブジェクト図よりコンポジット構造図の方が、構造が分かりやすい

SysMLで表すと

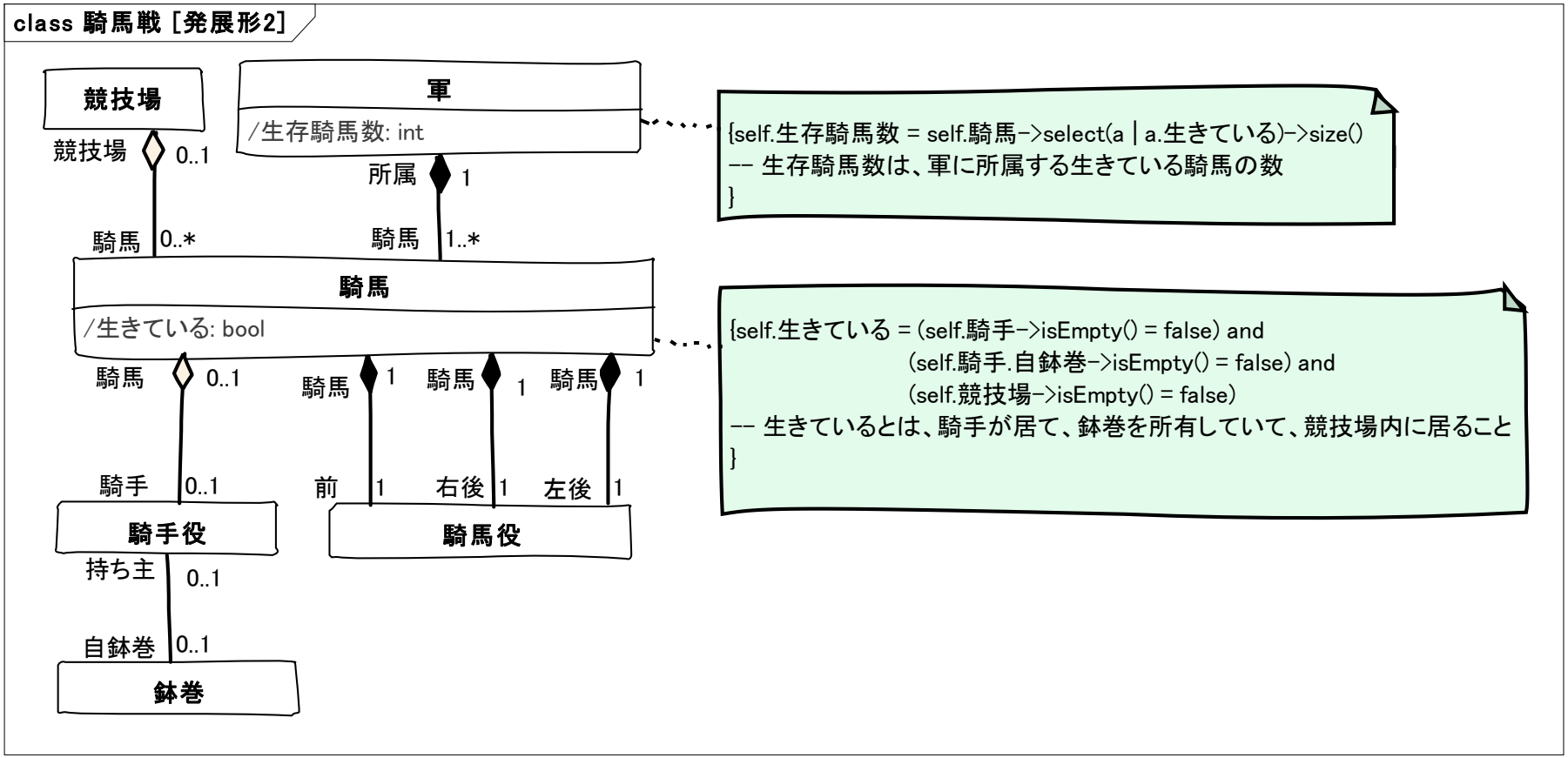


- このくらいの記述内容では、UMLとの差は小さい？

- ダイアグラムフレームに注目！
  ここを見るとモデルのどの部分の図なのか分かる

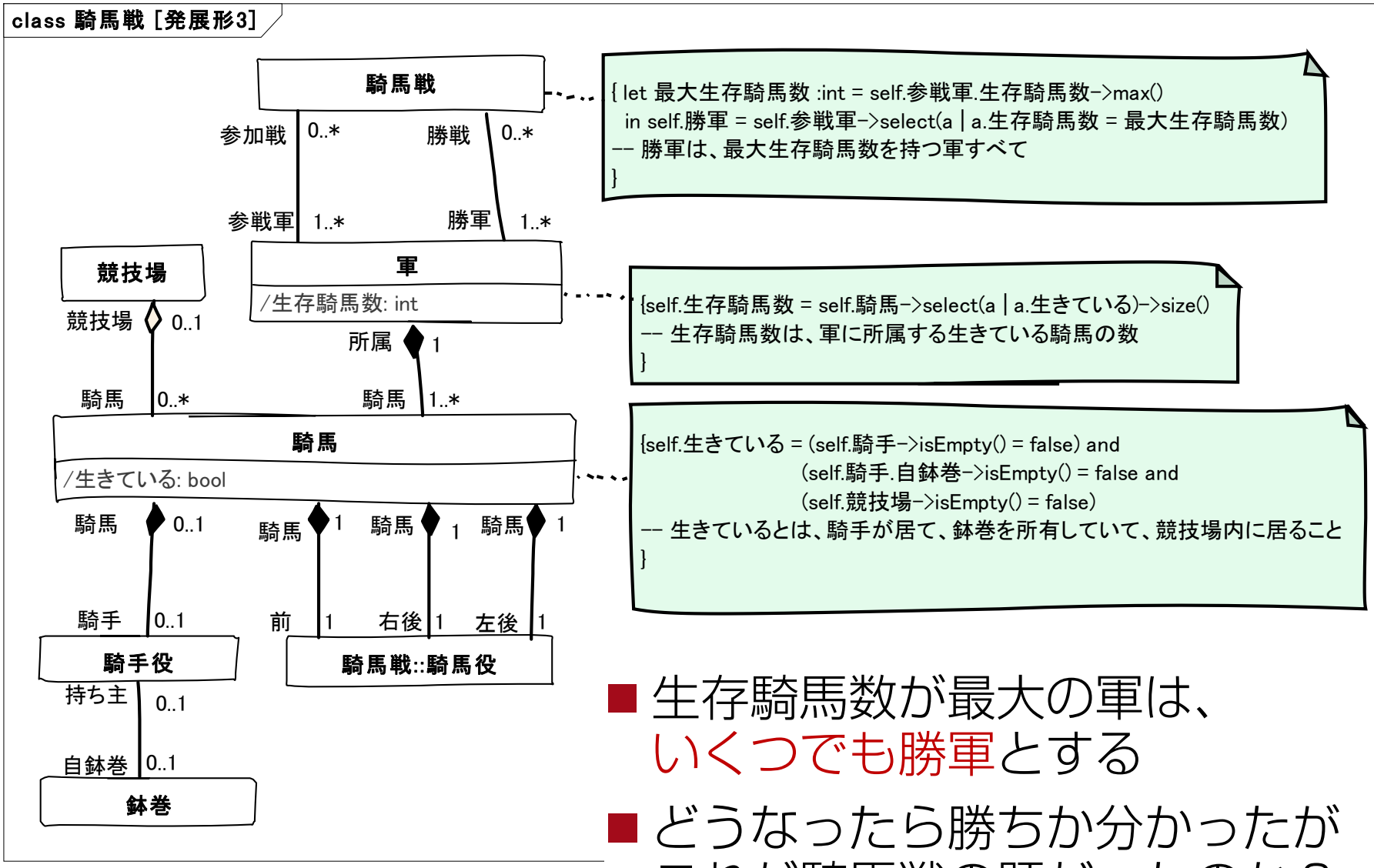


騎馬戦（発展形2）



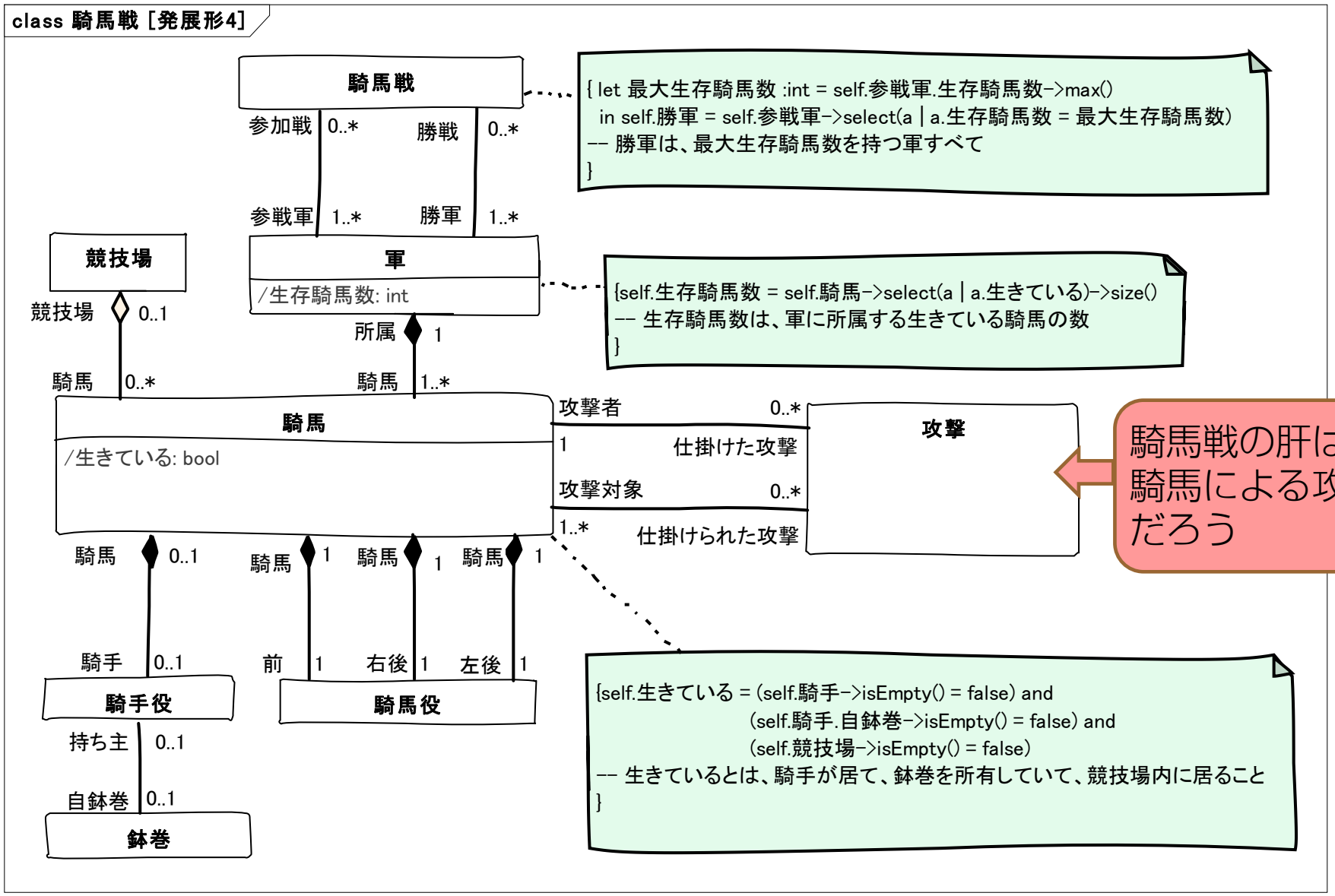
- 騎馬の集合体として「軍」について考えたもの
- 騎馬戦とは、複数の軍で争い、生存騎馬数を競うものだが…

騎馬戦（発展形3）



- 生存騎馬数が最大の軍は、いくつでも勝軍とする
- どうなったら勝ちか分かったが これが騎馬戦の肝だったのか？

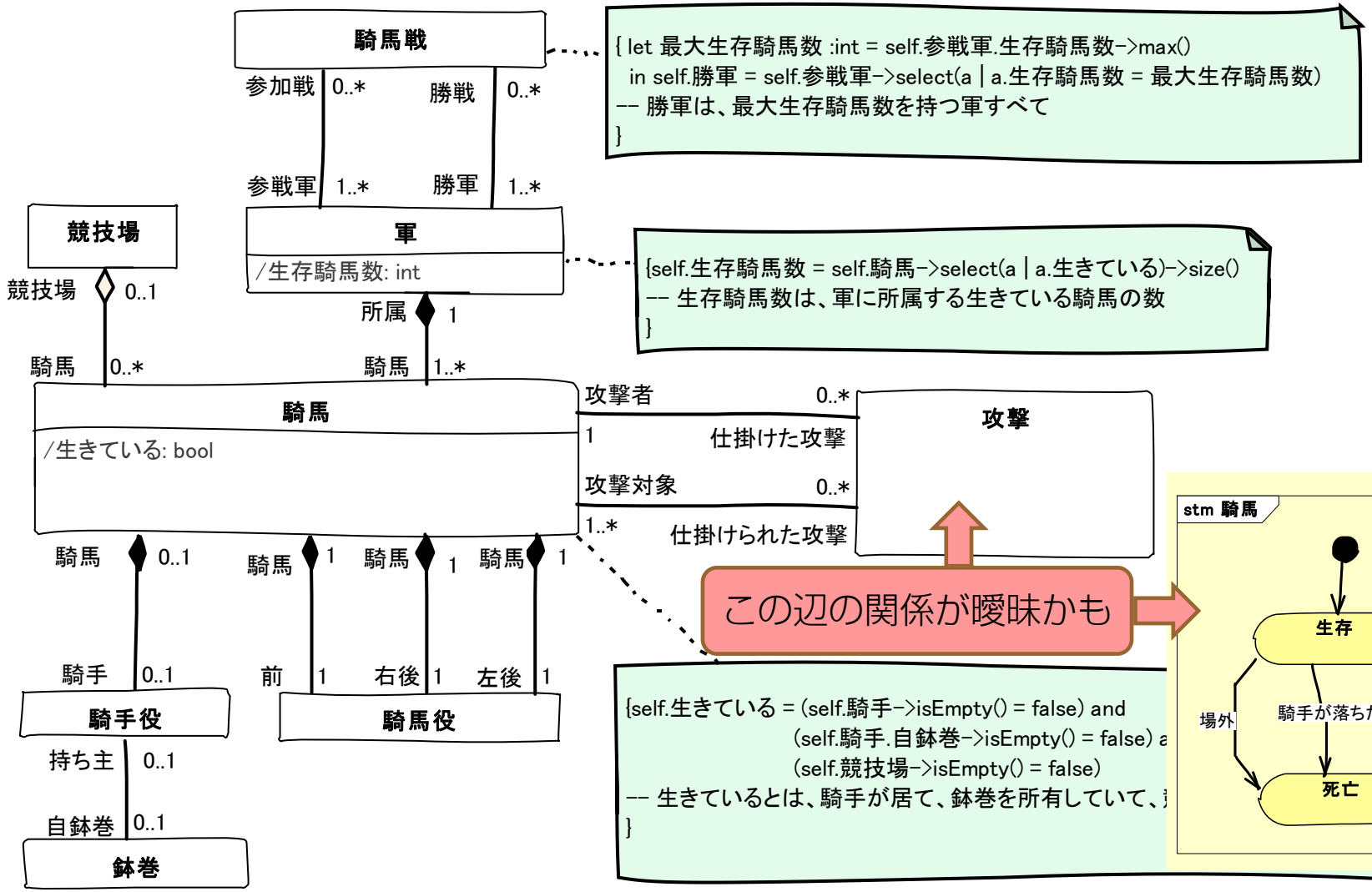
騎馬戦（発展形4）



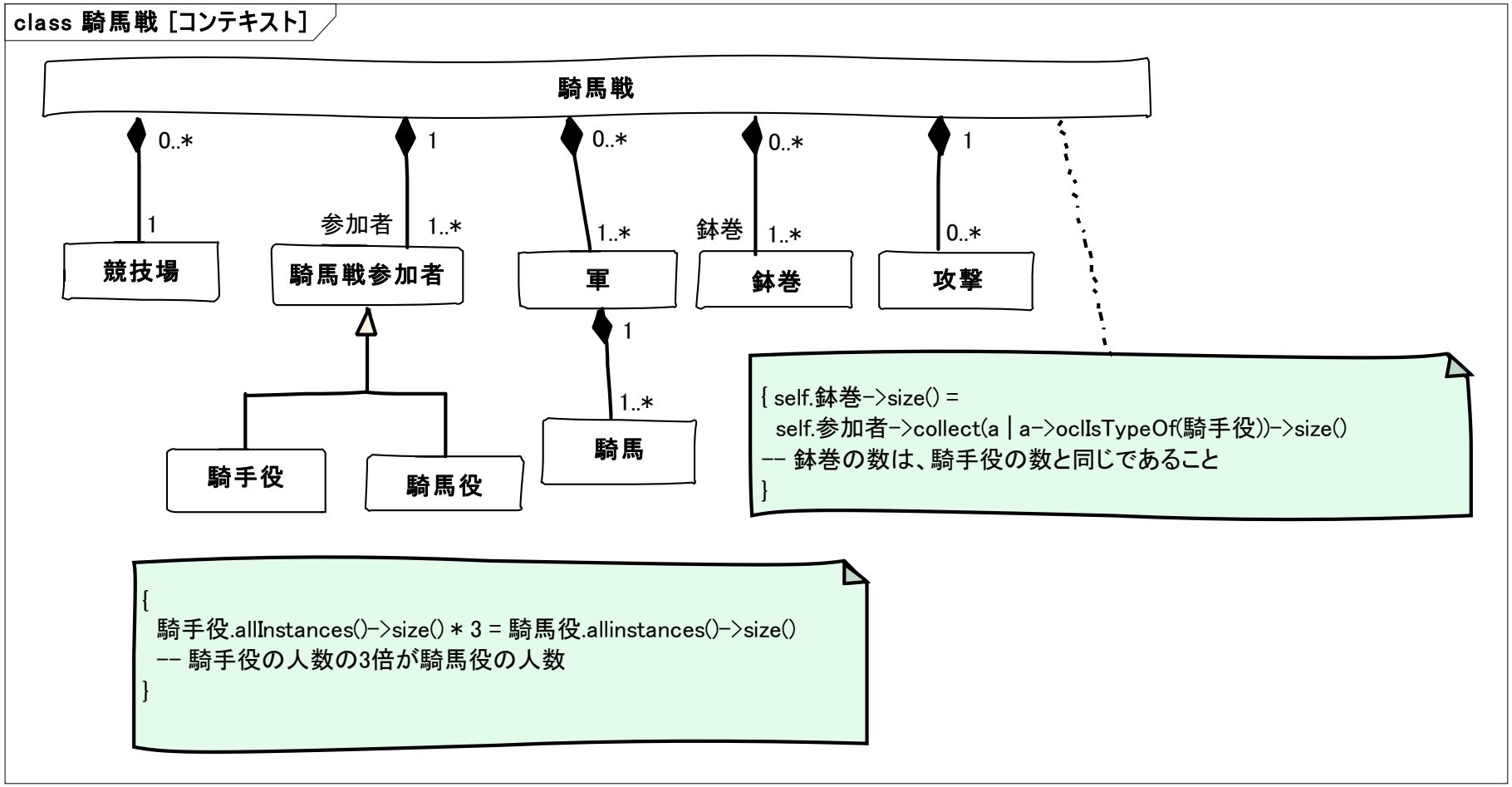
騎馬戦（発展形4補足）

FUJITSU

class 騎馬戦［発展形4］

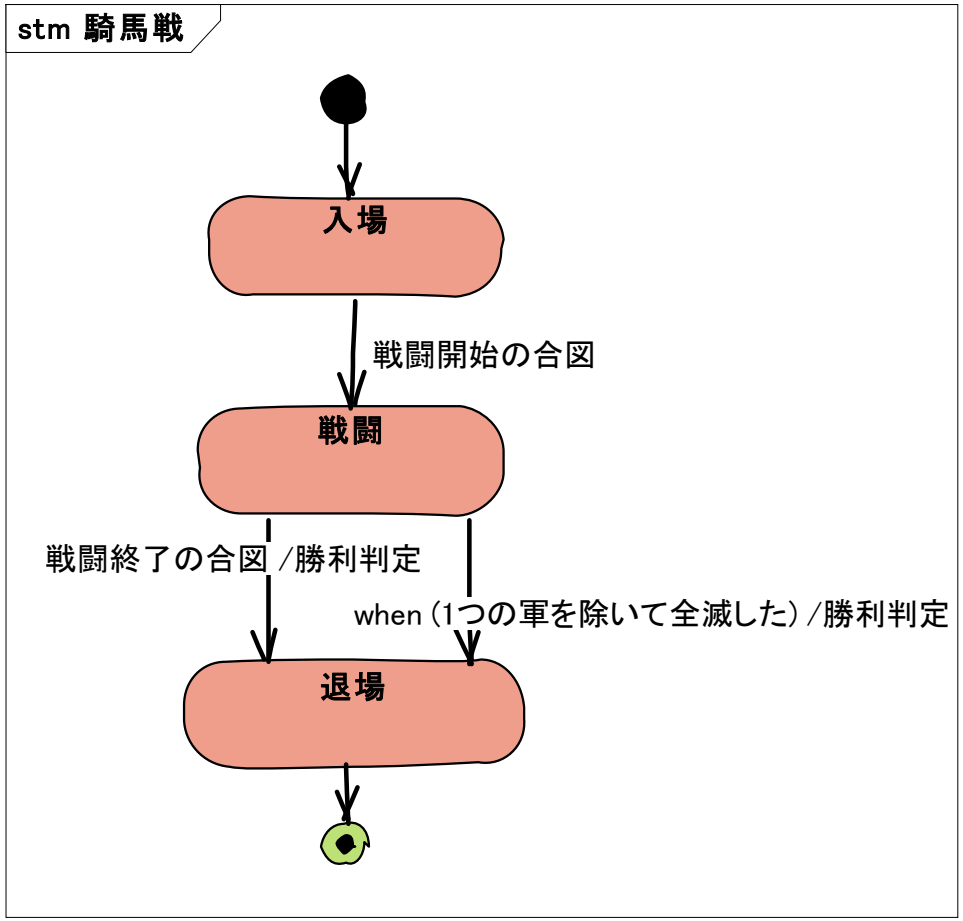


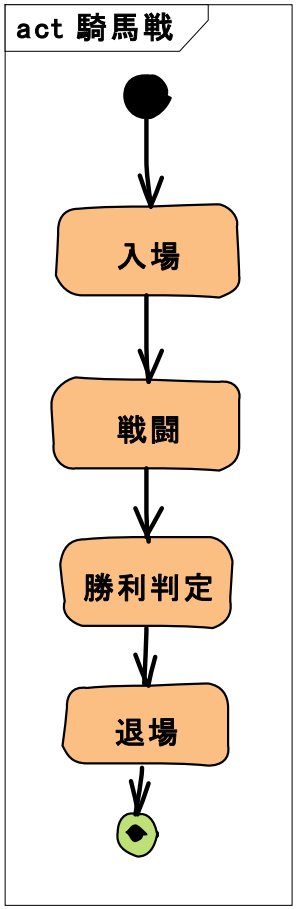
騎馬戦（コンテキスト図）



- 最初に登場人物などをざっくり説明すると良い

騎馬戦（振る舞い）





- ステートマシン図、アクティビティ図、シーケンス図
  表現力の違いを使い分けて

ここまでのまとめ

FUJITSU

- 最初に登場人物などを説明しよう
  - コンテキスト図
- 価値を生み出すものをしっかり抽出しよう
  - 騎馬戦の「攻撃」クラス
- 考える順と説明する順はちがいます
  - コンテキスト図を最初に書けるとは限らない

上流工程で、
問題領域をしっかりモデリングすることが大切です

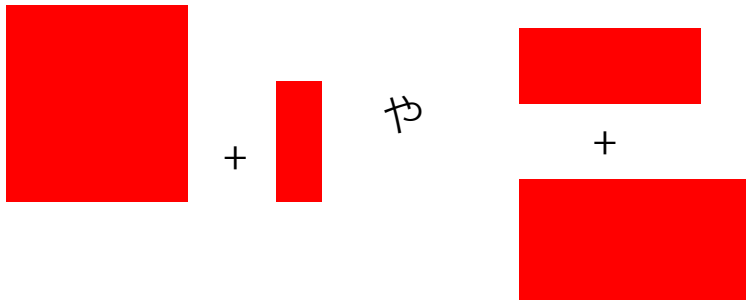
OMG Object Constraint Language (OCL)

FUJITSU

- UMLに制約をつけるときに使う言語
- 仕様書のありか ：http://www.omg.org/spec/OCL/
- なぜ制約が必要なのか？

定義： 赤い矩形、
ただしグレーの部分を除く

書きやすい範疇だけの
モデルから、意図を読み
取るのは困難

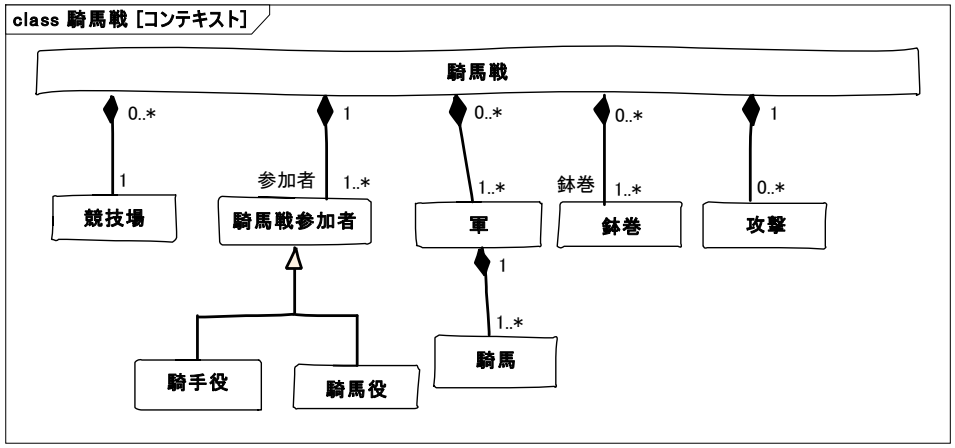


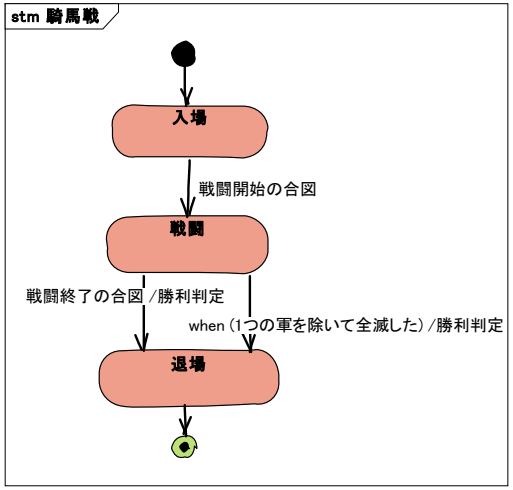
では、直感的でないことがある

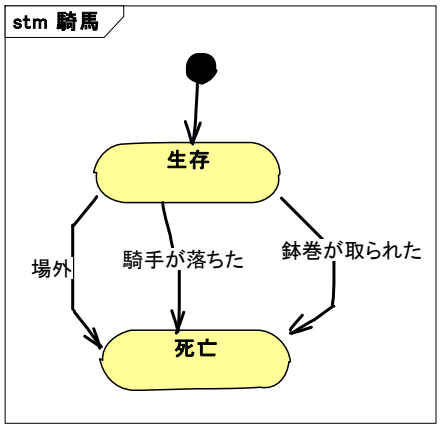
FUJITSU

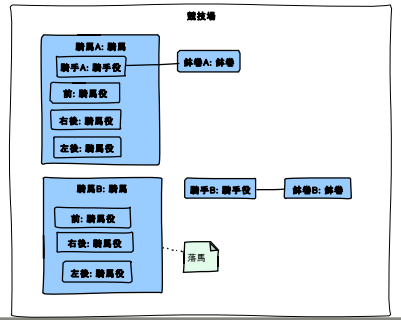
モデルに意図を込める#2
騎馬戦

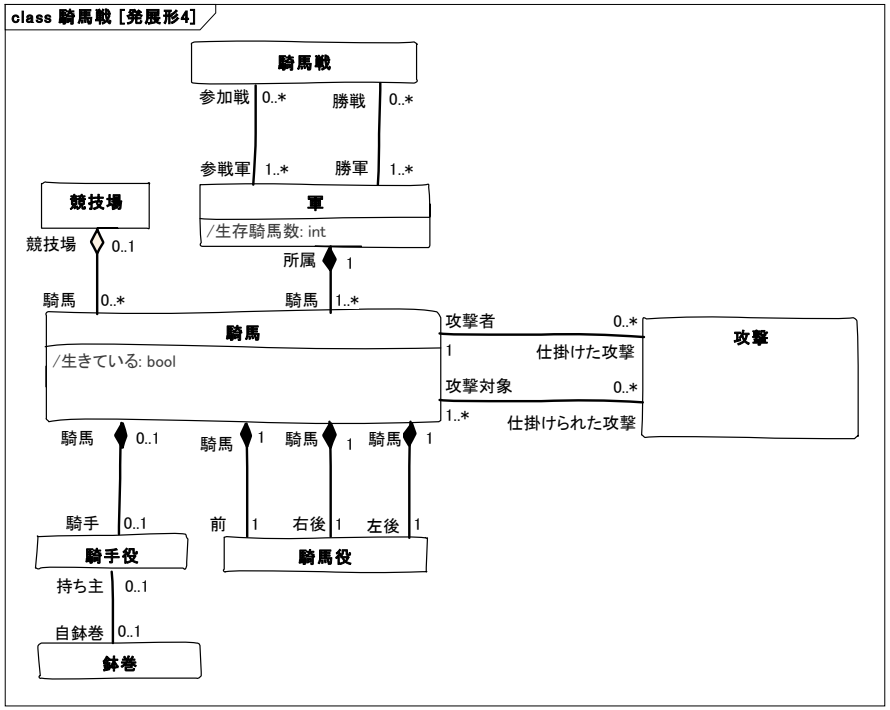
これで良かったのか？

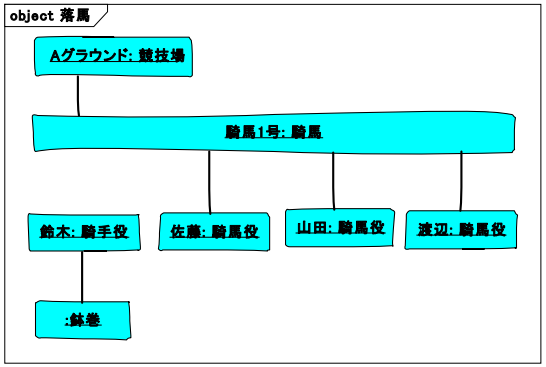


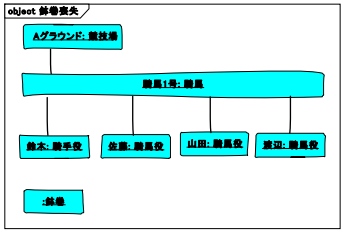


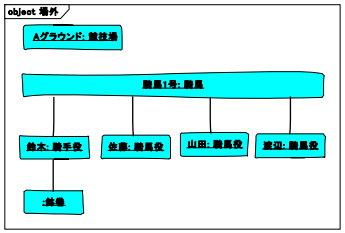






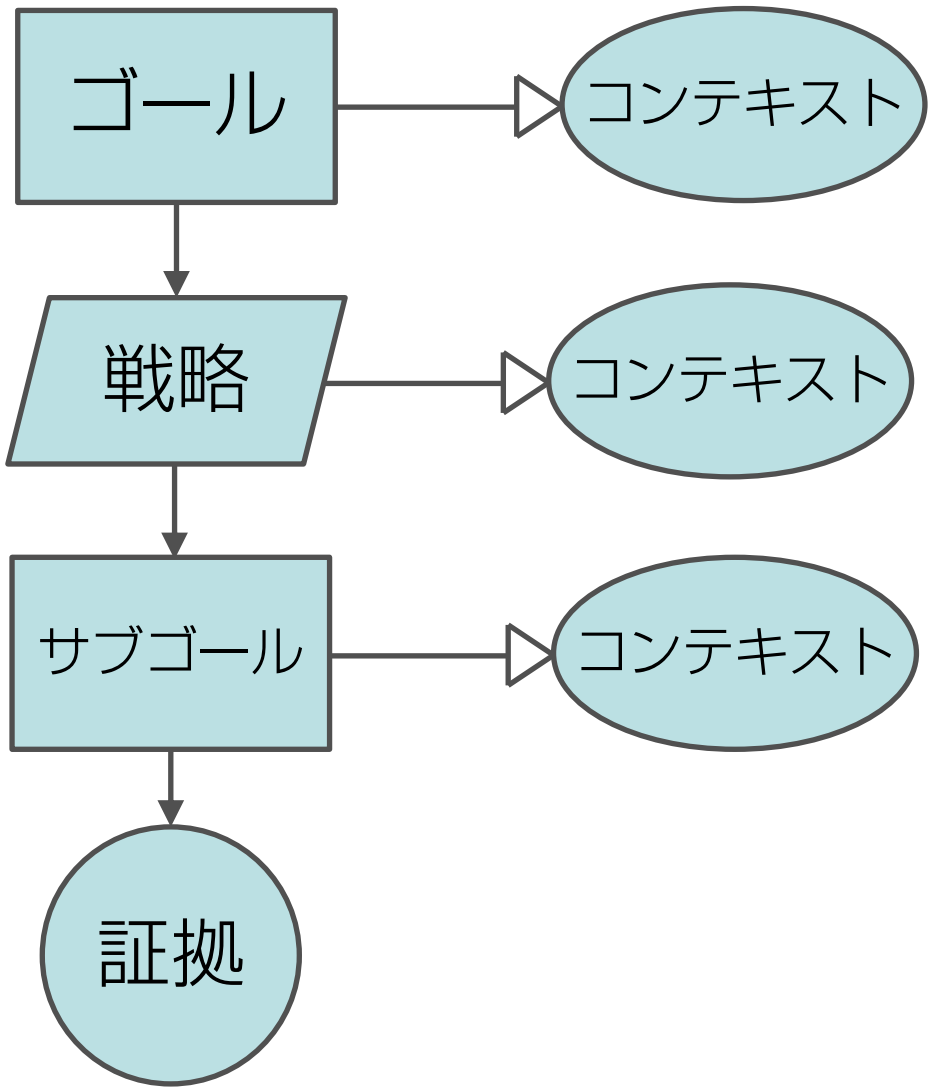






Goal Structuring Notation (GSN)で説明する

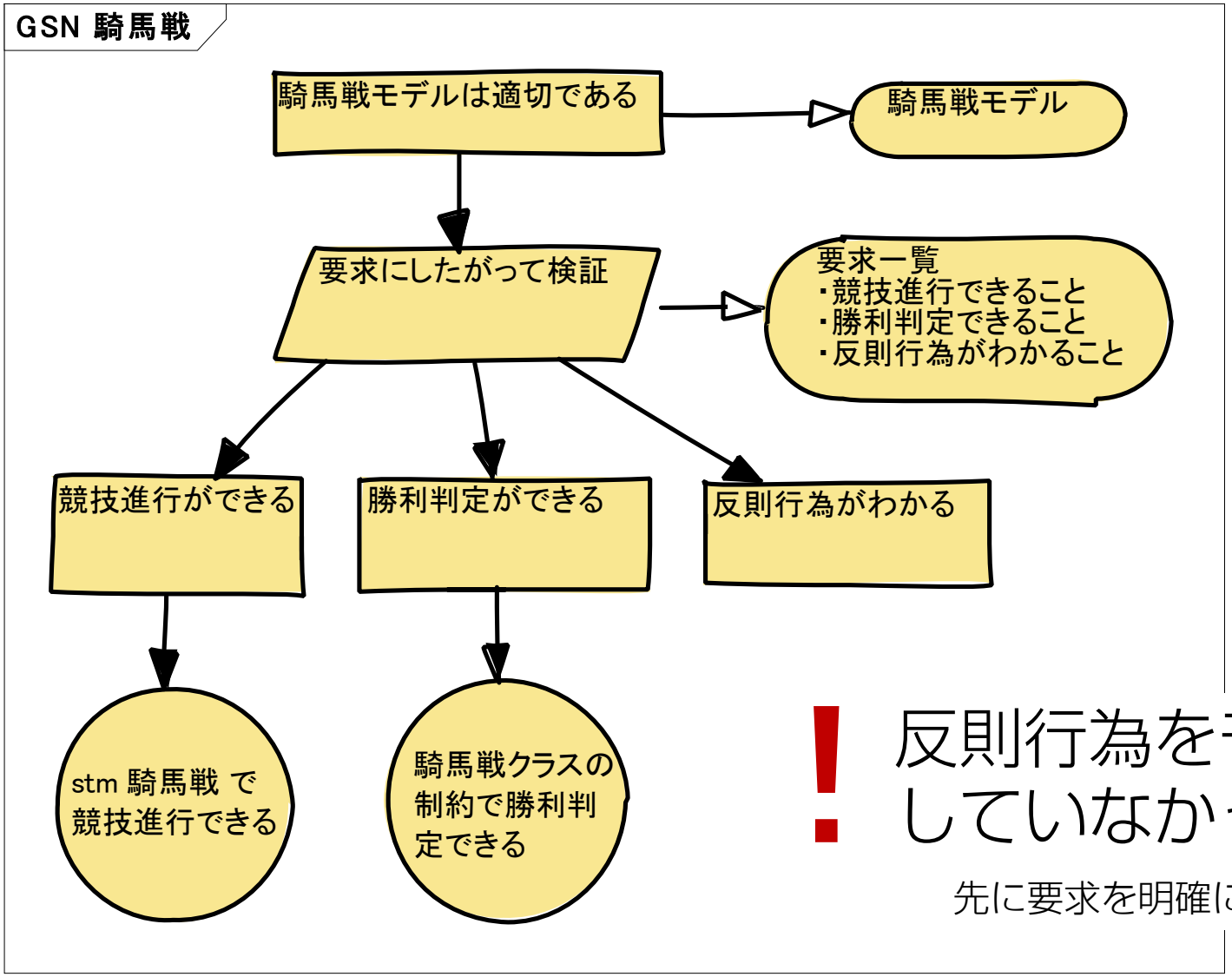
FUJITSU



- 議論をモデルにする書式
- ゴールを戦略で複数のサブゴールに分解
- 再帰的に分解して証拠までつなげる
- 下記URLで仕様を配布中

  http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf

騎馬戦モデルをGSNで説明する

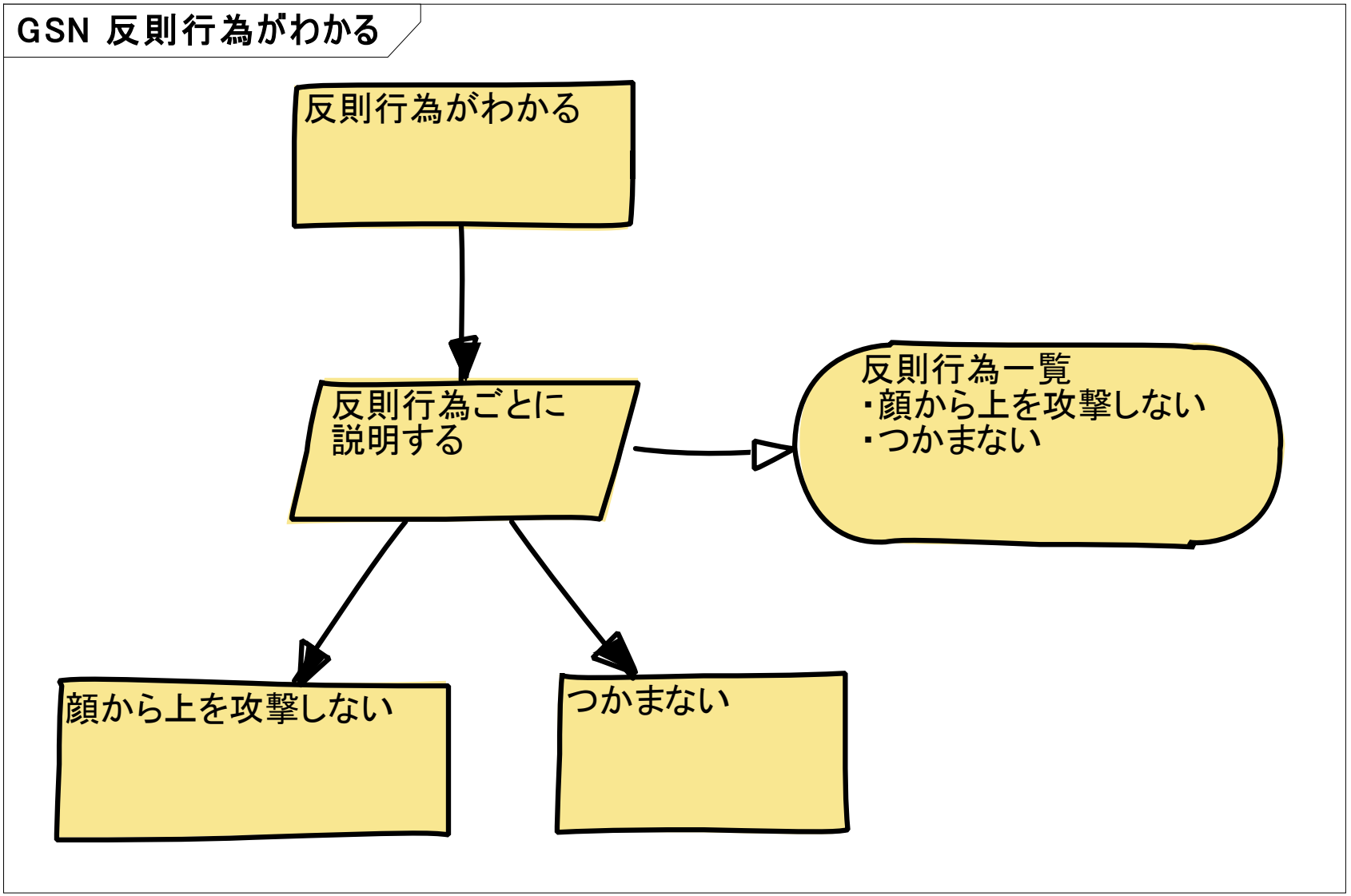


! 反則行為をモデルにしていなかった

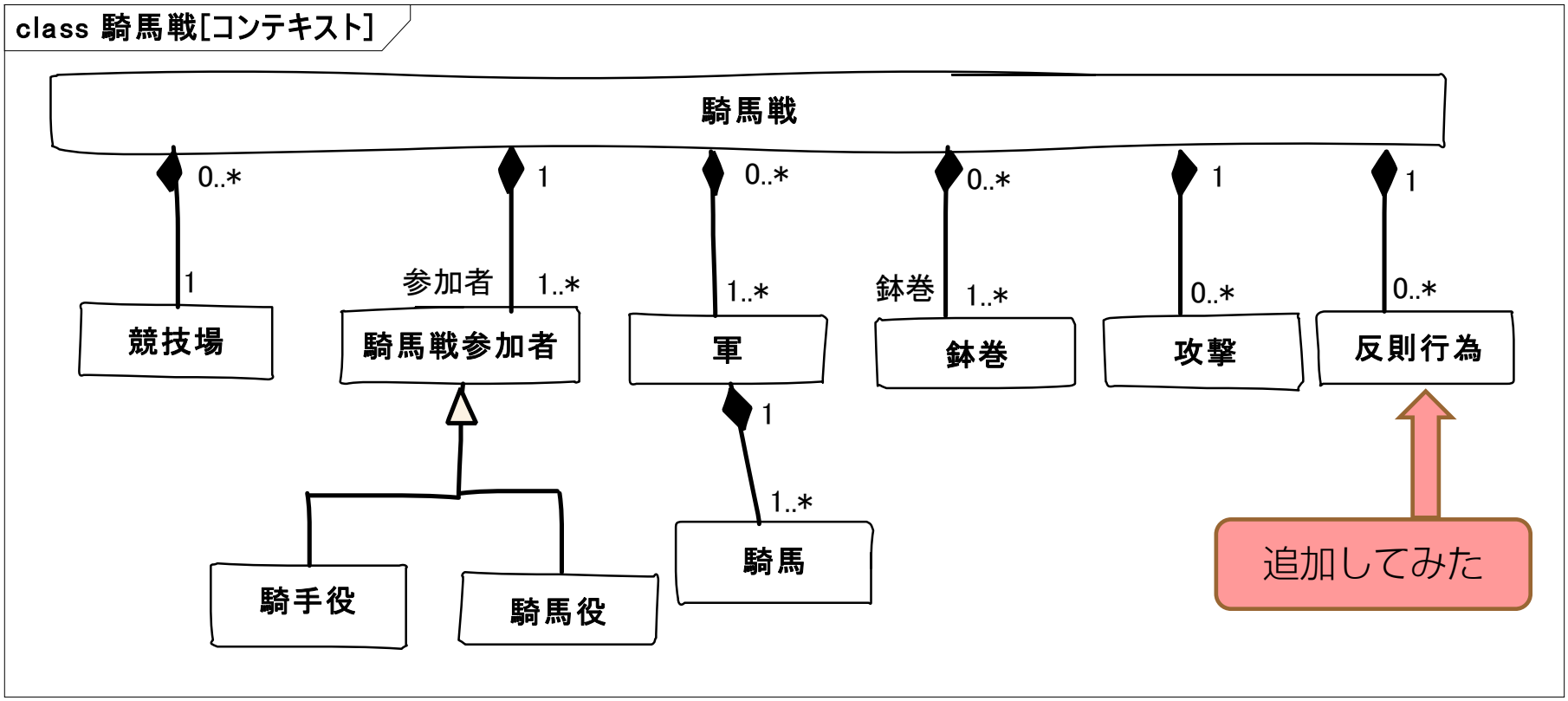
先に要求を明確にしないから…

先に反則行為を説明しよう

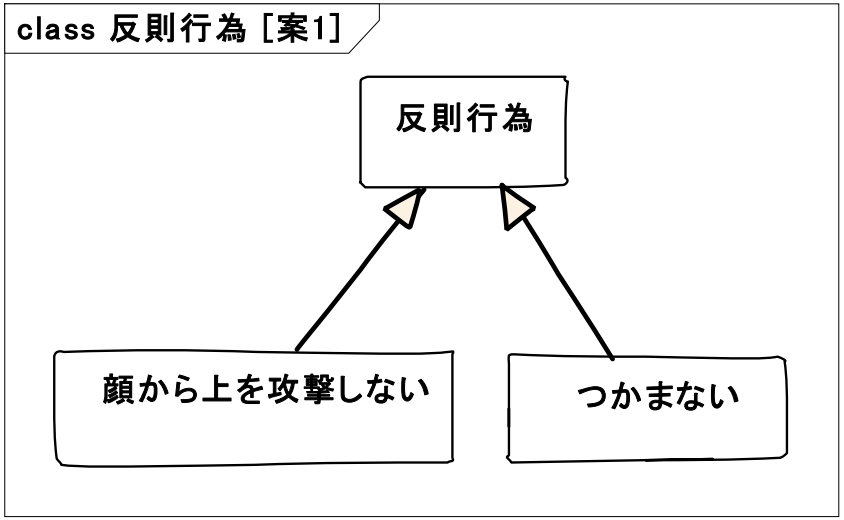
FUJITSU



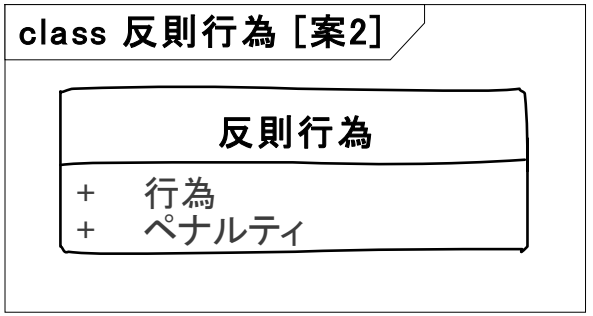
反則行為の追加

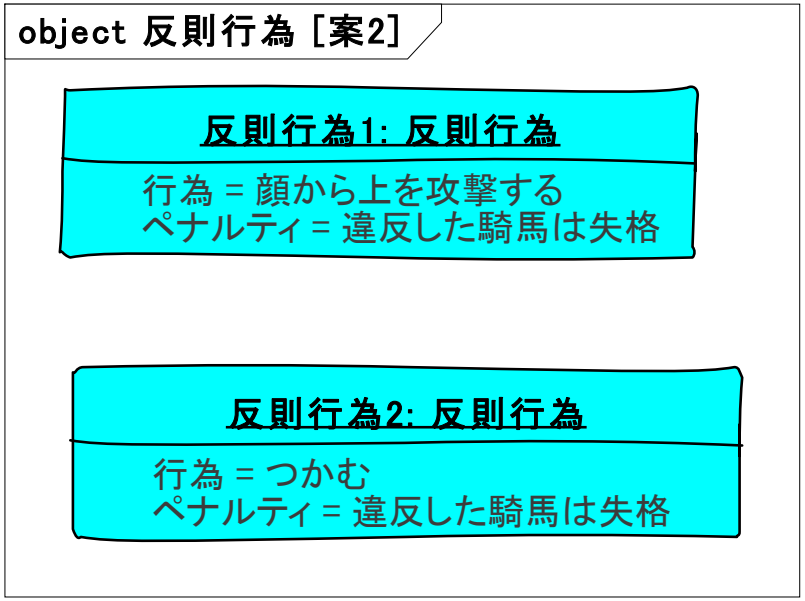


クラス or オブジェクト？



ペナルティを
うまく表せない…





騎馬に「失格」属性を足さないと…

ここまでのまとめ

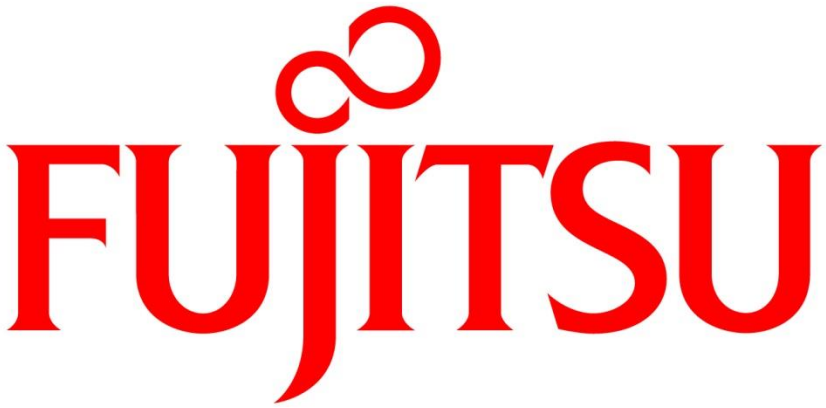
FUJITSU

- GSNでモデルの正しさを議論（説明）しよう
  - 偉い人に見てもらえば良い、というのは甘え
    - 正しさの議論の中に、「権威が同意した」という手法はある
      しかし、それだけではお寒い。
- 説明がまとも
  モデルが説明通り
  ⇒モデルはまとも
- 正しいモデルを残すだけでなく、
  どう説明できるか、が必要になります
  （第三者検証の時代）
- 考える順序と、説明の順序は異なることがあります

最後に

FUJITSU

- モデルに意図を込めるには引き出しが必要
  - 多重度に気を遣う
  - 制約を付ける
  - クラス or オブジェクト
  - etc...
- GSNを使って自分でモデルの正しさを説明しよう
  - 他人をまきこむ前に自己レビュー
- なんでもモデルにしてみよう
  - 製品から離れてモデリングしてみると、
    意外な気づきを得られます
    （私も騎馬戦モデルのためにOCLを再勉強しました）



shaping tomorrow with you