

モデル駆動開発と ドメイン特化言語

2013年10月16日

九州大学
大学院システム情報科学研究所
鵜林 尚靖



九州大学
KYUSHU UNIVERSITY

内容

- 第一部 ソフトウェア工学とモデル駆動開発(MDD)
- 第二部 MDD概論
 - MDDとは
 - コード生成の仕組み
 - ドメイン特化開発とDSL
- 第三部 研究事例（私たちの研究紹介）
 - 事例1：組込みソフトウェア開発のための外部環境分析
 - 事例2：Yet Another MDD ～コード生成を仮定しないMDD～
- 第四部 今後の動向

第一部 ソフトウェア工学と モデル駆動開発(MDD)

アイスブレイク: NATO会議覚えていますか？


NATO Software Engineering Conferences

homepages.cs.ncl.ac.uk/brian.randell/NATO/

SIGSOFT Resources NATO Software Engineering Conferences


[Home Page](#)

The NATO Software Engineering Conferences



P. Naur and B. Randell, (Eds.). Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO (1969) 231pp.

[\[Download PDF version \(2.3Mb\)\]](#)



B. Randell and J.N. Buxton, (Eds.). Software Engineering Techniques: Report of a conference sponsored by the NATO Science Committee, Rome, Italy, 27-31 Oct. 1969, Brussels, Scientific Affairs Division, NATO (1970) 164pp.

[\[Download PDF version \(3.2Mb\)\]](#)

[Introduction](#)

[The Writing of the NATO Reports](#)

[Includes [Masterpiece Engineering](#)]

[Photographs - Software Engineering 1968](#)

[Photographs - Software Engineering Techniques 1969](#)

Last updated 13 Aug 2001 - B. Randell

NATO Software Engineering Conference 1968

homepages.cs.ncl.ac.uk/brian.randell/NATO/N1968/index.html

SIGSOFT Resources NATO Software Engineering Conference 1968

[The NATO Software Engineering Conferences](#)

NATO Software Engineering Conference

Garmisch, Germany, 7-11 Oct 1968

Photographs provided by Robert McClure and Brian Randell.

				
E. Austin	J.D. Babcock	R.S. Barton	F.L. Bauer	R. Bemer
				
J. Berghuis	E.E. David	E.W. Dijkstra	M. Engeli	L.K. Flanigan
				
B. Galler	S. Gill	H.R. Gillette	A.E. Glennie	R.M. Graham
				
D. Gries	J.A. Harr	I. Hugo	J.N.P. Hume	H.A. Kinslow
				

<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/>

報告書の目次

4

4 5

CONTENTS

HIGHLIGHTS	3
CONTENTS	5
PREFACE	9
1. BACKGROUND OF CONFERENCE	13
2. SOFTWARE ENGINEERING AND SOCIETY	19
3.1. THE NATURE OF SOFTWARE ENGINEERING	19
3.2. SOFTWARE ENGINEERING MANAGEMENT AND METHODOLOGY	24
3.3. DESIGN AND PRODUCTION IN SOFTWARE ENGINEERING	32
4. DESIGN	35
4.1. INTRODUCTION	35
4.1.1. Sources of techniques	35
4.1.2. Need for hardware based on program structure	35
4.1.3. Relation to mathematics	37
4.2. DESIGN CRITERIA	38
4.2.1. General design criteria	38
4.2.2. User requirements	40
4.2.3. Reliability and design	44
4.2.4. Logical completeness	44
4.3. DESIGN STRATEGIES AND TECHNIQUES	45
4.3.1. Sequencing the design process	45
4.3.2. Structuring the design	50
4.3.3. Feedback through monitoring and simulation	53
4.3.4. High-level languages	55
4.4. COMMUNICATION AND MANAGEMENT IN DESIGN	59
5. PRODUCTION	65
5.1. INTRODUCTION	65
5.1.1. The problems of scale	65
5.1.2. The problems of reliability	70
5.2. PRODUCTION — MANAGEMENT ASPECTS	72
5.2.1. Production planning	72
5.2.2. Personnel factors	83
5.2.3. Production control	86
5.2.4. Internal communication	89
6. SERVICE	103
6.1. INTRODUCTION	103
6.1.1. The virtue of realistic goals	103
6.1.2. Initial system release	103
6.1.3. Frequency of releases	104
6.1.4. Responsibility for modified systems	106
6.2. REPLICATION, DISTRIBUTION AND MAINTENANCE	107
6.2.1. Replication	107

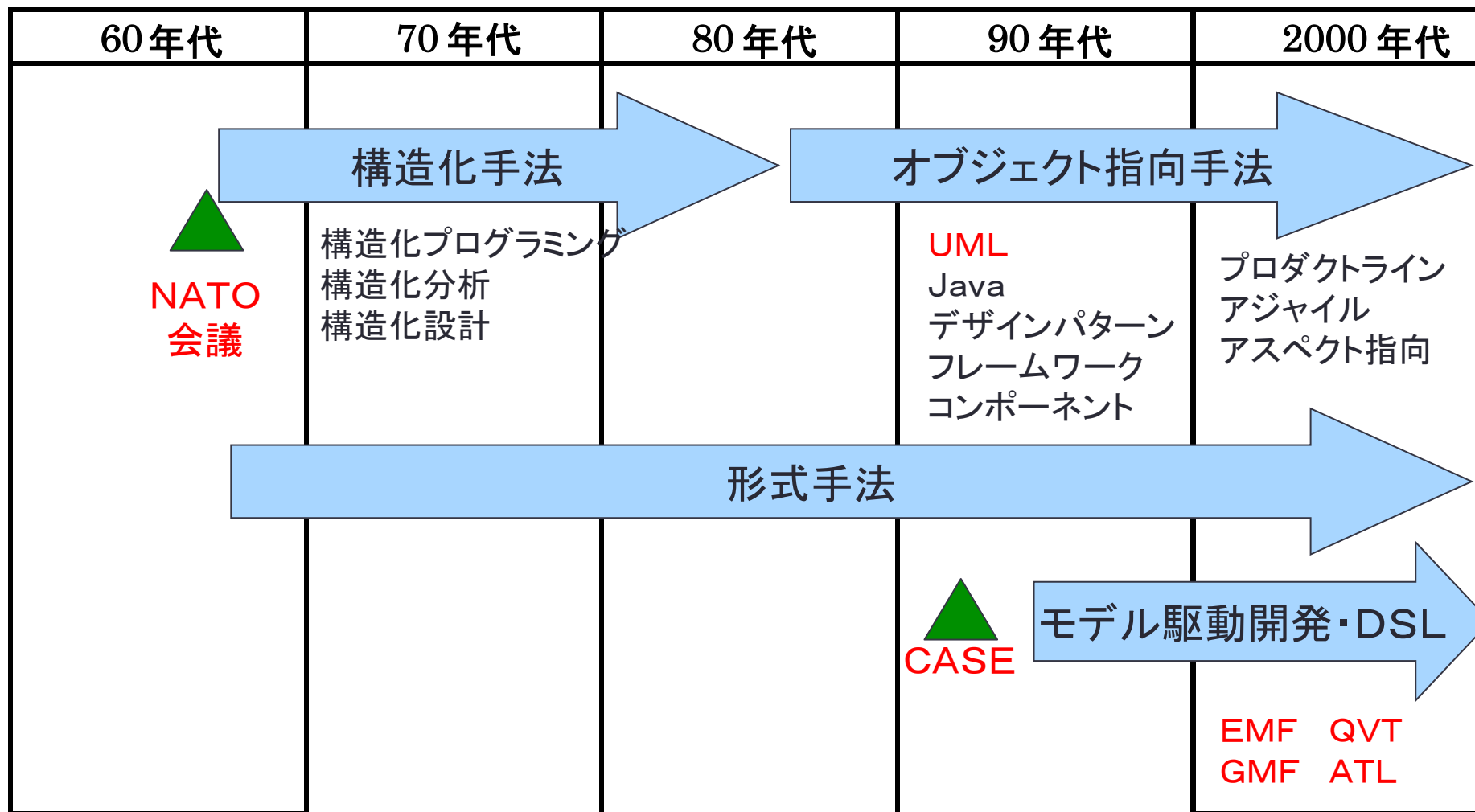
NATO SOFTWARE ENGINEERING CONFERENCE 1968

5

6.2.2. Distribution	109
6.2.3. Maintenance	110
6.3. SYSTEM EVALUATION	112
6.3.1. Acceptance testing	113
6.3.2. Performance monitoring	114
6.4. FEEDBACK TO MANUFACTURERS FROM USERS	115
6.5. DOCUMENTATION	116
6.6. REPROGRAMMING	117
7. SPECIAL TOPICS	119
7.1. SOFTWARE: THE STATE OF THE ART	119
7.1.1. Introduction	119
7.1.2. Problem areas	120
7.1.3. The underlying causes	122
7.1.4. Possible solutions	124
7.1.5. Summary	125
7.2. EDUCATION	125
7.3. SOFTWARE PRICING	129
7.3.1. Introduction	129
7.3.2. Arguments in favour of separate pricing	150
7.3.3. Arguments against separate pricing	131
7.3.4. Splitting software and hardware production	132
7.3.4.1. The argument in favour	132
7.3.4.2. The argument against	133
8. INVITED ADDRESSES	135
8.1. KEYNOTE SPEECH by A.J. Perlis	135
8.2. 'MASS PRODUCED' SOFTWARE COMPONENTS, by M.D. McIlroy	138
8.2.1. Discussion	151
9. WORKING PAPERS	157
R.W. Bemer, A.G. Fraser, A.E. Glennie, A. Opler, and H. Wiehle: <i>Classification of subject matter</i>	160
R.W. Bemer: <i>Checklist for planning software system production</i>	165
E.W. Dijkstra: <i>Complexity controlled by hierarchical ordering of function and variability</i>	181
S. Gill: <i>Thoughts on the sequence of writing software</i>	186
A.I. Llewelyn and R.F. Wickens: <i>The testing of computer software</i>	189
T.B. Pinkerton: <i>Performance monitoring and systems evaluation</i>	200
B. Randell: <i>Towards a methodology of computing system design</i>	204
F. Selig: <i>Documentation standards</i>	209
APPENDICES	213
A 1 Participants	213
A 2 Conference Schedule	218
A 3 Original list of topics that were to be discussed	219
A 4 Addresses of welcome	224
CONTRIBUTOR INDEX	226
SUBJECT INDEX	228

NATO SOFTWARE ENGINEERING CONFERENCE 1968

ソフトウェア開発技術の変遷



モデリングに関する主な会議

モデリング専門

- **MODELS**
 - International Conference on Model Driven Engineering Languages and Systems
- **MiSE**
 - Modelling in Software Engineering @ ICSE
- **ECMFA**
 - European Conference on Modeling Foundations and Applications
- **DSM**
 - Workshop on Domain-Specific Modeling @ SPLASH
- **MODELSWARD**
 - International Conference on Model-Driven Engineering and Software Development
- **UML&FM**
 - International workshop UML and Formal Methods

ソフトウェア工学・言語

- **ICSE**
 - International Conference on Software Engineering
- **FSE**
 - ACM SIGSOFT Symposium on the Foundations of Software Engineering
- **ASE**
 - International Conference on Automated Software Engineering
- **GPCE**
 - International Conference on Generative Programming: Concepts & Experiences
- **SLE**
 - International Conference on Software Language Engineering
- **FOSD**
 - International Workshop on Feature-Oriented Software Development

現在の研究テーマは

研究

- Model-driven development and its proven effectiveness
- Development and use of domain-specific languages
- Evolution of general-purpose modeling languages and related standards
- Definition of the **syntax and semantics of modeling languages**
- Tools and **meta**-tools for modeling languages and model-driven development
- **Model composition and metamodel composition**
- Definition and usage of **model transformations** and generative approaches
- Support for legacy issues and **evolution of models**
- Proposals for new **model quality assurance** techniques: analysis, testing, model checking, **validation and verification**
- Model-driven development and **cyber-physical** systems
- New methodologies/frameworks/processes for model-driven development
- Development of systems engineering and modeling-in-the-large concepts
- Integration of modeling languages and tools: hybrid **multi-modeling approaches**
- New modeling paradigms and formalisms
- Modeling in, and for, the **Cloud**

応用

- Introducing model-driven approaches into organizations
- Application, technology and methodology case studies (successful and unsuccessful)
- **Managing evolution of models**
- Industrial scale **model management** (model size, user group size, viewpoint size etc.)
- Limitations, gaps and mismatches in current modeling standards
- Economic issues of model driven development
- Achieving industrial quality benchmarks with model driven development
- Industrial requirements for domain specific modeling
- Visions for industrial strength model driven engineering
- Empirical studies about model-based development



青色：活発な研究テーマ

赤色：新しいテーマ

MDDのためのリポジトリ ReMoDD

Repository for Model Driven Development (ReMoDD) Overview | ReMoDD

www.colostate.edu/remodd/v1/

ReMoDD The Repository for Model-Driven Development

Search this site: Search

User login

Username:

Password:

Log in

○ Create new account

○ Request new password

MDD Artifacts

○ Browse

○ Browse Workshop Pages

The ReMoDD Project

○ Overview

○ Mission

○ ReMoDD Policies

○ Feature Highlights

○ Advisory Board Members

○ Senior Investigators

○ Presentations and Publications

○ Statistics

○ Video Tour

Popular content

Today's:

○ Repository for Model Driven Development (ReMoDD) Overview

○ Umple submission for Comparing Modeling Artifacts workshop at Models 2012

○ Towards the Model Driven Organization (AMINO 2013) @ MODELS 2013

All time:

Repository for Model Driven Development (ReMoDD) Overview

The Repository for Model-Driven Development (ReMoDD) is a resource that aims to support the work of researchers and educators in the Model-Driven Development (MDD) community. Researchers and practitioners can use the repository as a vehicle for sharing exemplar models, illustrative descriptions of modeling methodologies and techniques, detailed modeling case studies, modeling success stories and other forms of modeling experience and knowledge. Resources shared within ReMoDD can be used to gain significant insights into the use of models across the software lifecycle, as a source of data for MDD experiments, as a source of models for testing MDD tools, and to better understand relationships among ongoing MDD research projects. In particular, educators and trainers can use ReMoDD resources to illustrate modeling concepts and approaches in the classroom.

The development of the ReMoDD infrastructure is led by researchers in Colorado State University's Computer Science Department and Michigan State University's Computer Science Department.

Contact Info

Colorado State University Department of Computer Science
Michigan State University Department of Computer Science and Engineering
E-mail: remodd(dash)project(at)cs.colostate.edu

The development of this site is supported by the National Science Foundation under Computing Research Infrastructure Grant CNS-0854988.

Public MDD Events on ReMoDD

The MODELS Conference
The Models in Software Engineering (MISE) Workshop @ ICSE
AMINO: TowArds the Model DriveN Organization

Major MDD Event Websites

ECMFA 2013
MODELS 2013
MODELS 2013 Workshop on Models@run.time
MODELS 2013 Workshop on GeMOC: Globalization of Modeling Languages
MODELS 2013 Workshop on CMA: Fourth International Conference on Modeling Approaches

Major MDD Initiatives and Resources

The ATL Project
EMF: The Eclipse Modeling Framework
The Epsilon Project
GeMOC: Globalization of Modeling Languages
GME: The Generic Modeling Environment
Open Model Initiative
Open Model Initiative (Austria)
The Papyrus project
The SSELab

Communities

○ Forums

○ General

○ ReMoDD Feedback

○ Groups

Major MDD Initiatives and Resources

- The ATL Project
- EMF: The Eclipse Modeling Framework
- The Epsilon Project
- GeMOC: Globalization of Modeling Languages
- GME: The Generic Modeling Environment
- Open Model Initiative
- Open Model Initiative (Austria)
- The Papyrus project
- The SSELab

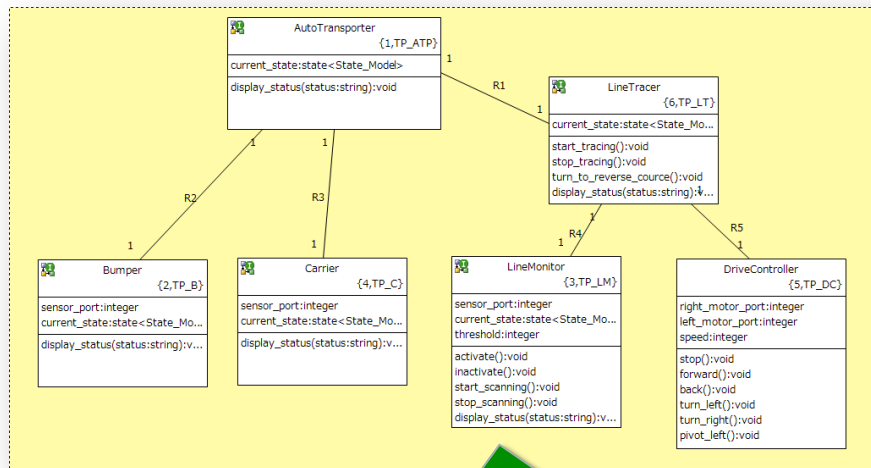
第二部 MDD概論



MDDとは

モデル駆動開発 MDD: Model Driven Development

設計モデル



モデルを動作させてテストできないの？



(半分でもいいから)自動化できないの？

ソースコード

```

/*
 * Structural representation of application analysis class:
 *   AutoTransporter (TP_ATP)
 */
struct Transporter_TP_ATP {
    Escher_StateNumber_t current_state;
    /* application analysis class attributes */

    /* relationship storage */
    Transporter_TP_LT * TP_LT_R1;
    /* Note: No storage needed for TP_ATP->TP_B[R2] */
    /* Note: No storage needed for TP_ATP->TP_C[R3] */
};

#define Transporter_TP_ATP_MAX_EXTENT_SIZE 10
extern Escher_Extent_t pG_Transporter_TP_ATP_extent;
extern void Transporter_TP_ATP_op_display_status( Transporter_TP_ATP * );

extern void Transporter_TP_ATP_R1_Link( Transporter_TP_LT *,
/* Note: TP_LT<-R1->TP_ATP unrelate accessor not needed */
extern void Transporter_TP_ATP_R2_Link( Transporter_TP_B *,
/* Note: TP_B<-R2->TP_ATP unrelate accessor not needed */
extern void Transporter_TP_ATP_R3_Link( Transporter_TP_C *,
/* Note: TP_C<-R3->TP_ATP unrelate accessor not needed */

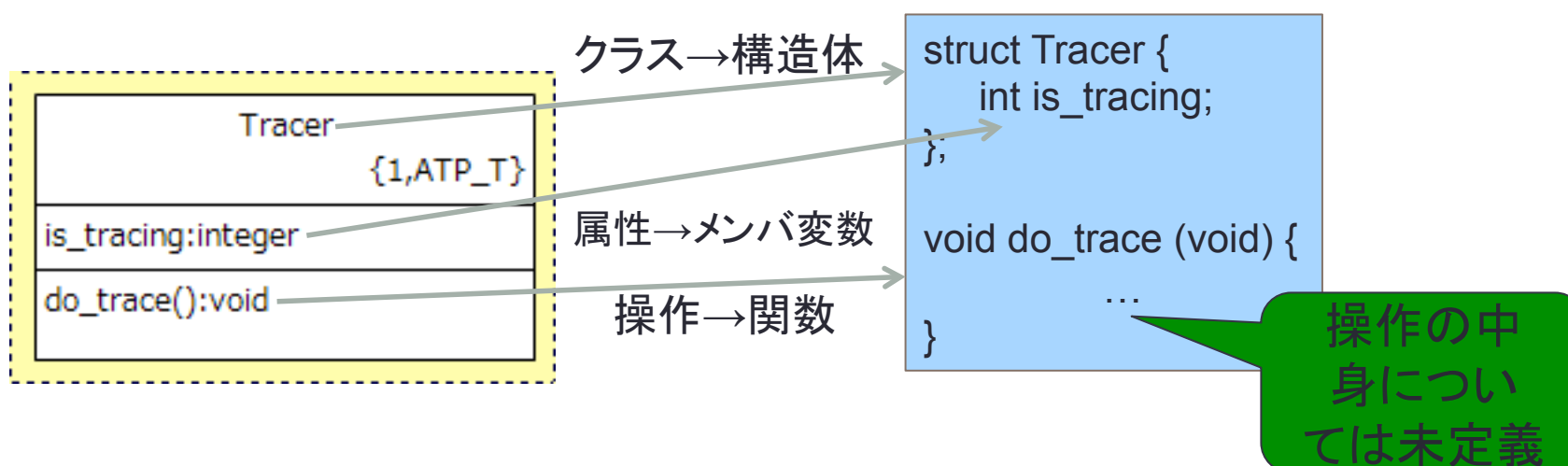
```

モデルを使って開発を楽にする方法

モデルからコードへの変換～人間が実行

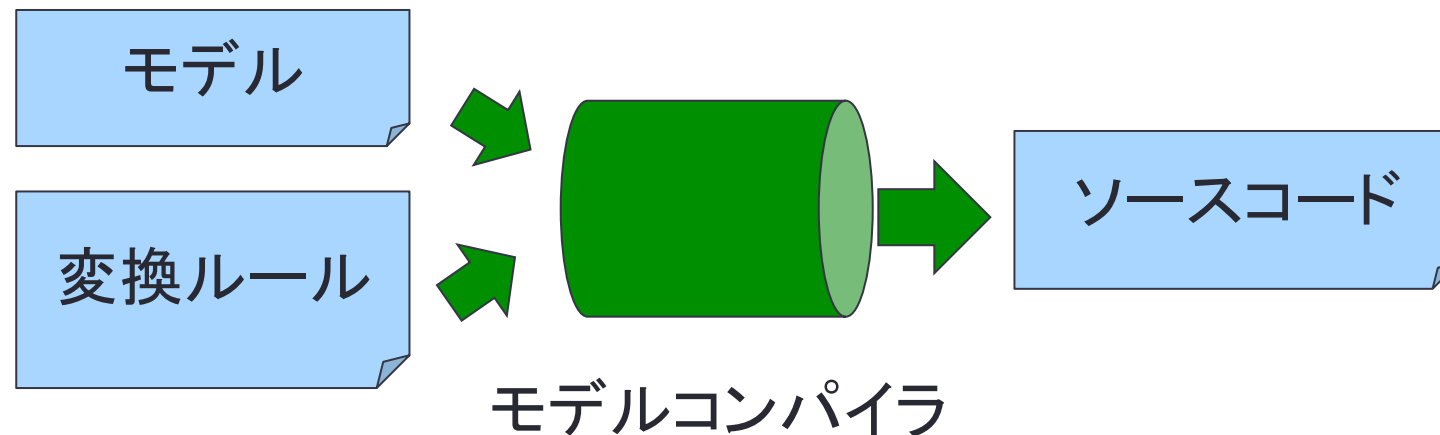
- クラスのソースコードへの変換を考えると・・・
- 決まりきったソースコードのパターンがある

モデル要素	ソースコード
クラス	構造体
属性	構造体のメンバ変数
操作	関数

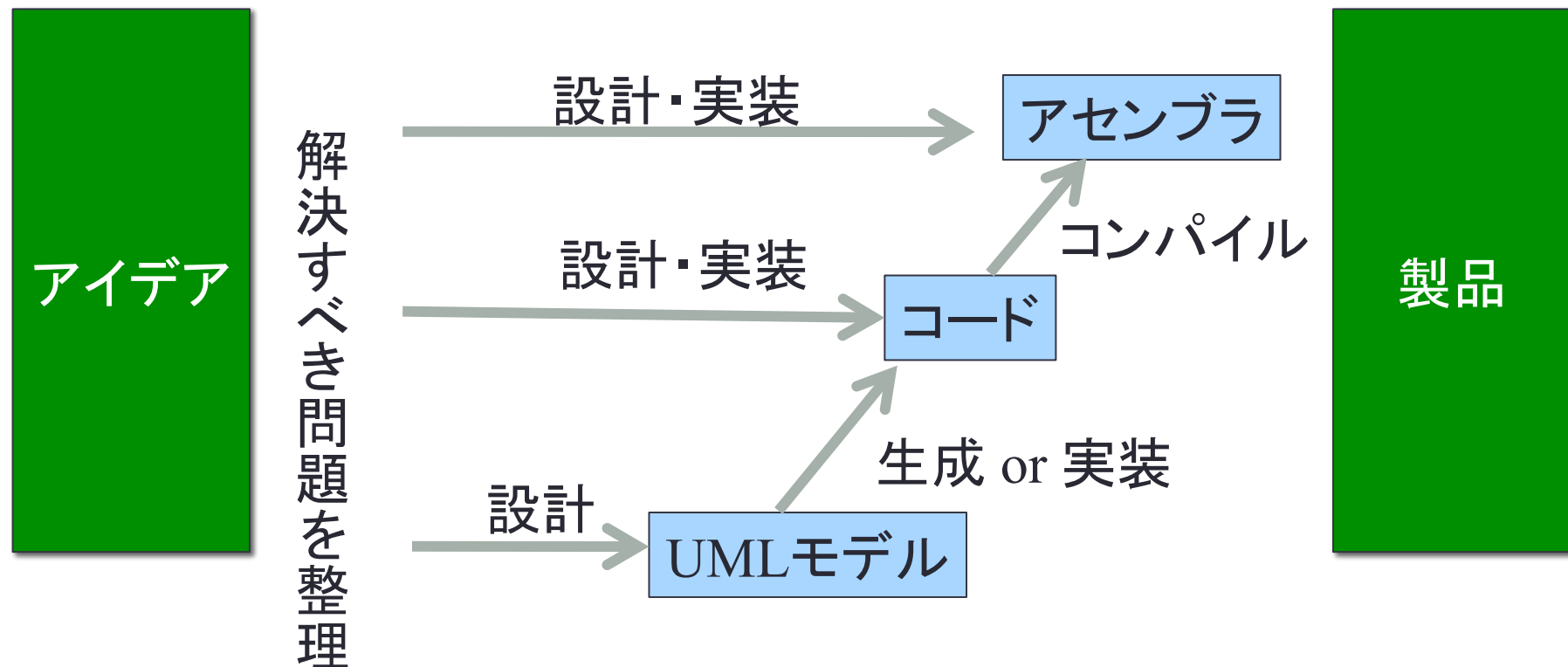


モデルからコードへの変換 ～コンピュータが実行

- 決まり切ったパターンがあるならば、コンピュータに実行させることができるのでは？
- 人間が行う「実装のパターン」を「変換ルール」としてとらえる

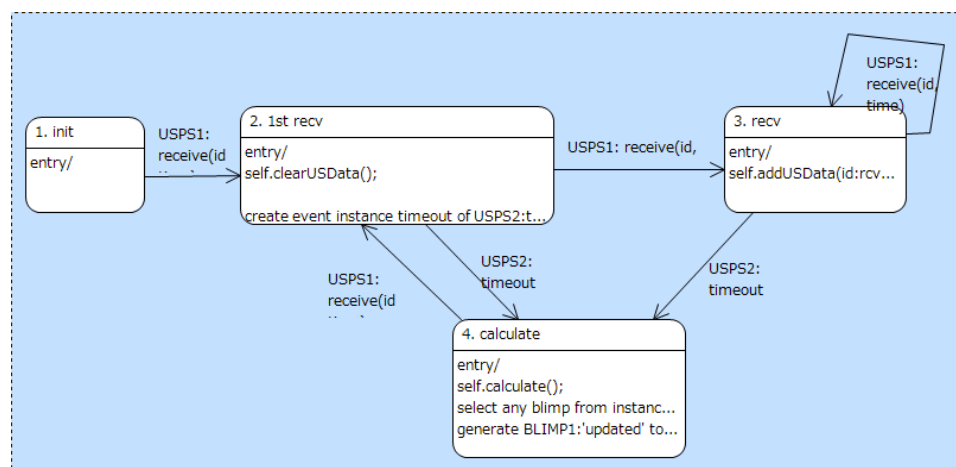


モデルの抽象度と生産性



モデルを動作させてテスト

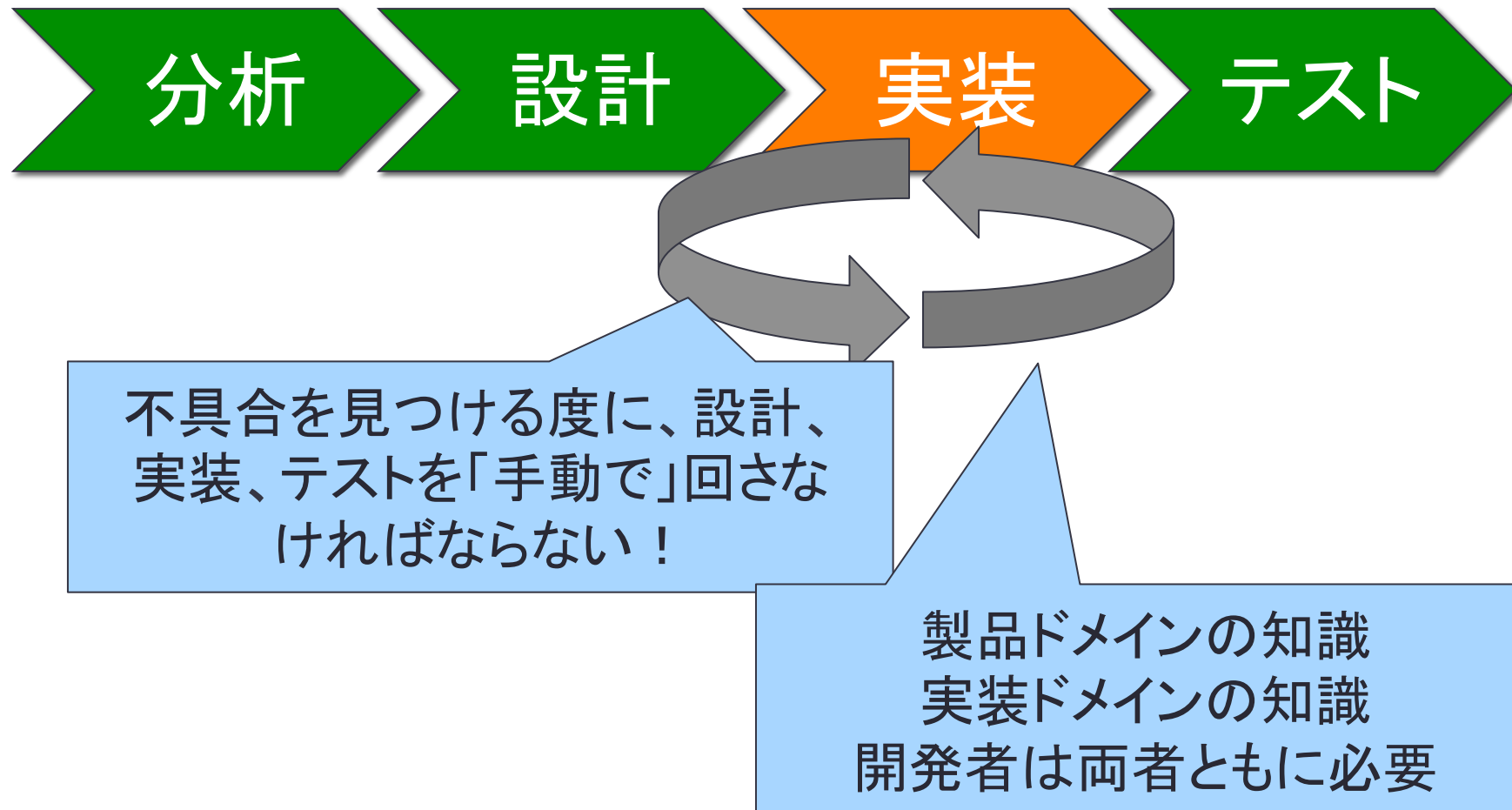
- 設計段階で動かすことはできないか？
 - 実装する前にモデル上で振る舞いをシミュレーションしてテストできれば、問題の早期発見につながる
 - 実装を待たなくてもテストが可能
- 形式的な(= コンピュータが解釈可能な)モデルを利用



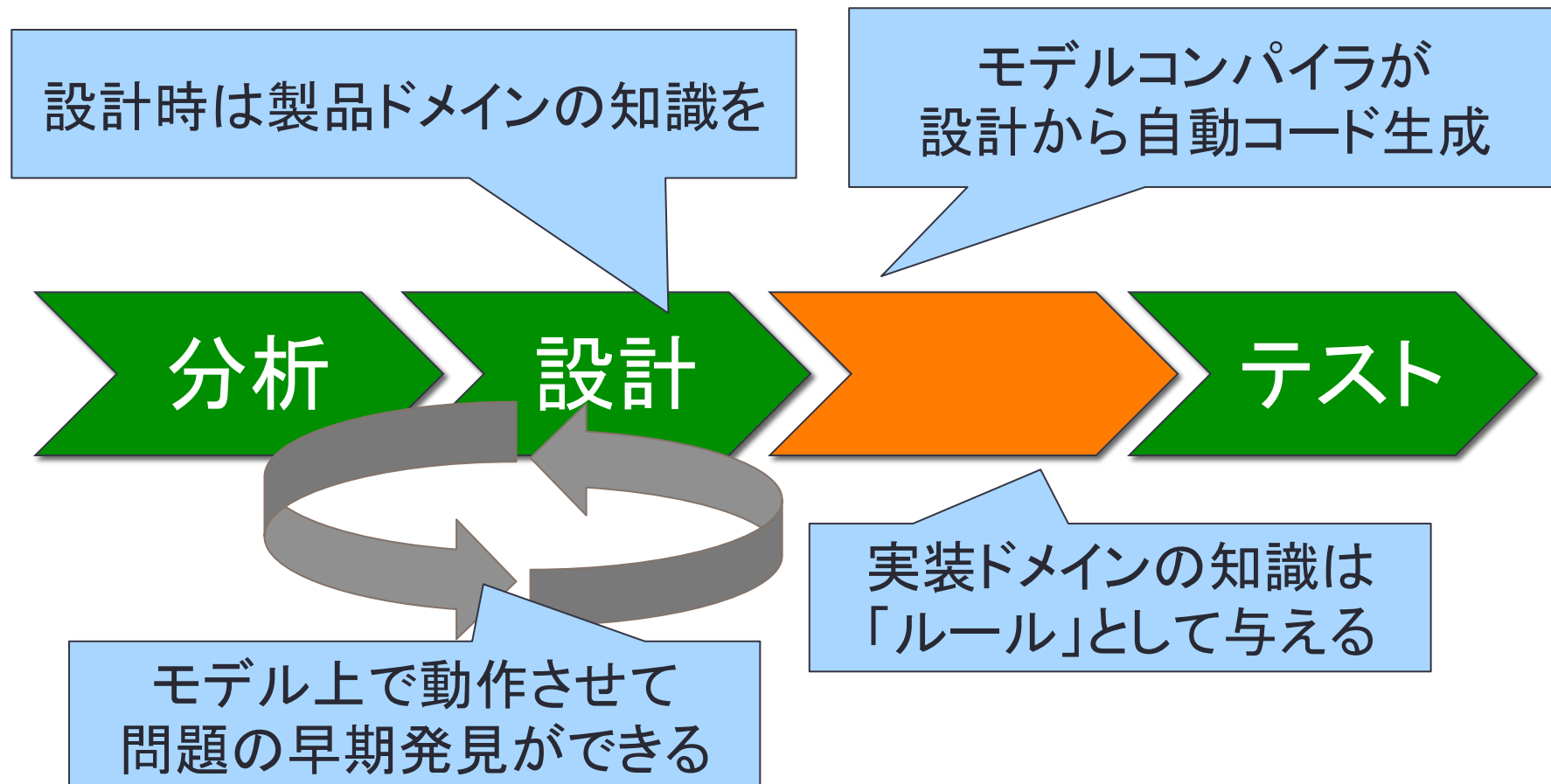
従来型開発とモデル駆動開発の比較

- 従来型開発 (aka ソースコード中心の開発)
 - 要求仕様書、設計文書をもとにソースコードを開発
 - 上流での要求仕様書や設計文書の正しさを担保するのはレビュー
 - 実際の製品の品質保証をするのは「テスト」
 - 開発の半分以上がテスト...
 - 要求仕様書・設計文書とソースコードの一貫性を保つのは困難
- モデル駆動開発
 - 要求レベル、設計レベルのモデルを作成
 - モデルから(ある程度は)自動的にソースコードを生成
 - 上流での検証が比較的容易
 - 機械可読なモデルがあるのでシミュレーションが可能
 - 自動的にソースコードを生成するので、モデルとコードの一貫性保持が容易

開発スタイルの変化: モデル駆動開発以前



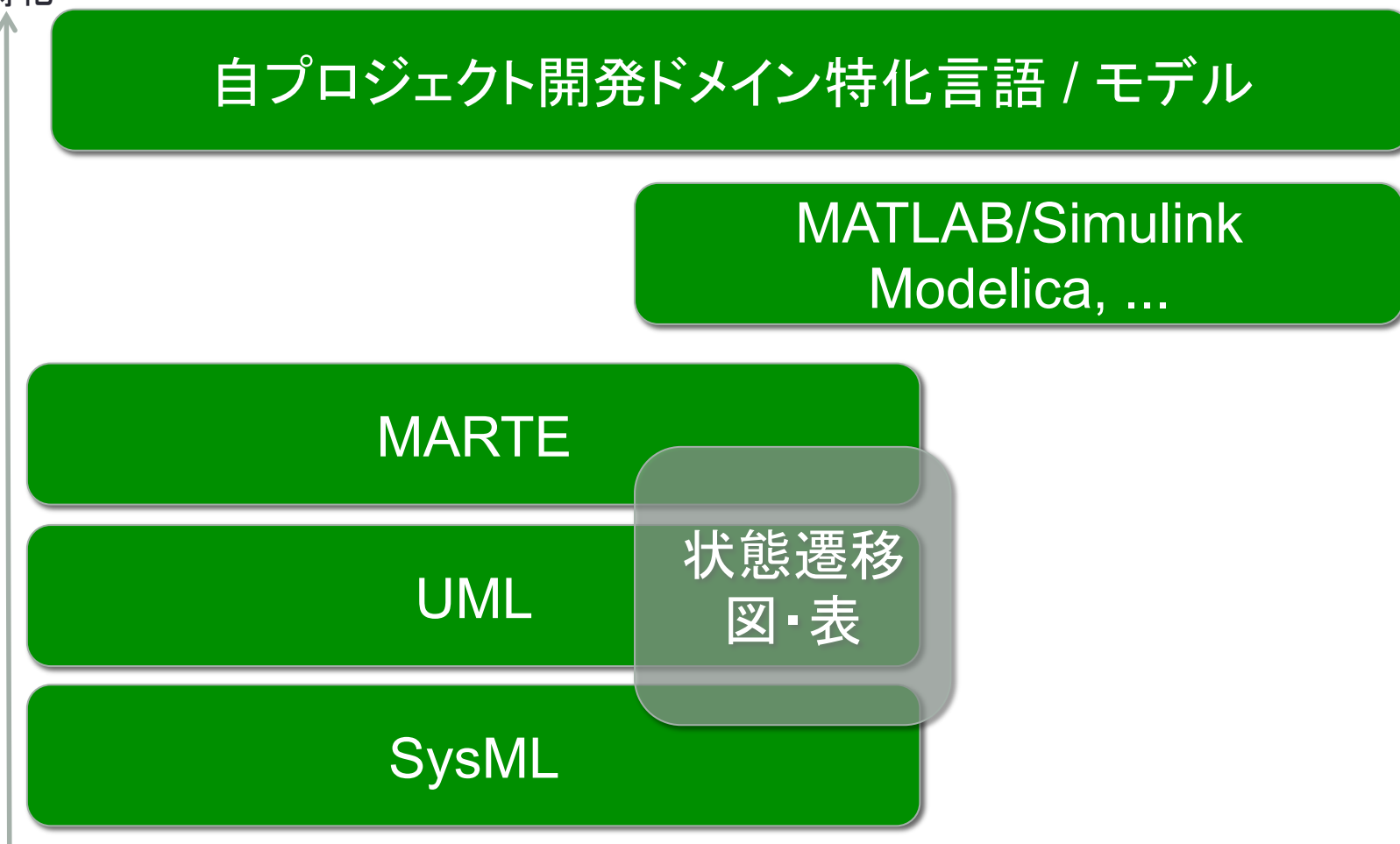
開発スタイルの変化: モデル駆動開発以降



関心の分離ができる！

モデル地図

ドメイン特化
度合い ↑



アーキテクチャ記述

(モデル駆動)

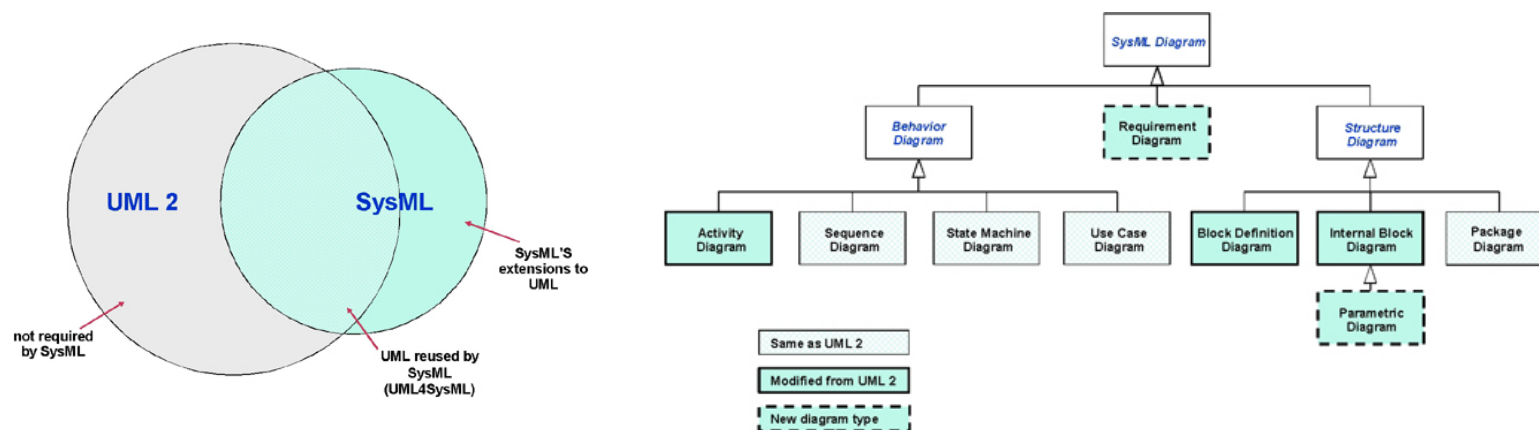
言語理論

制御工学(モデルベース)



システムモデリング言語SysML

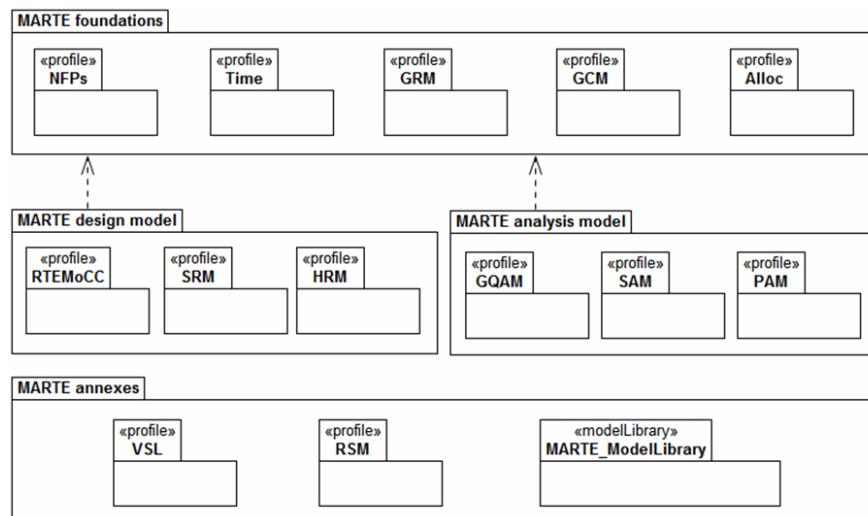
- SysML (Systems Modeling Language) は、システムをモデリングするための言語である。UMLはソフトウェアの分析、設計に用いられるが、組込みシステムのようにハードウェアとソフトウェアから構成される場合には若干力不足の側面がある。たとえば、ハードウェア制約などを記述するのが容易ではない。
- SysMLの言語仕様は、UMLの仕様を再利用した部分と新たに拡張した部分から構成される。



UML profile for MARTE



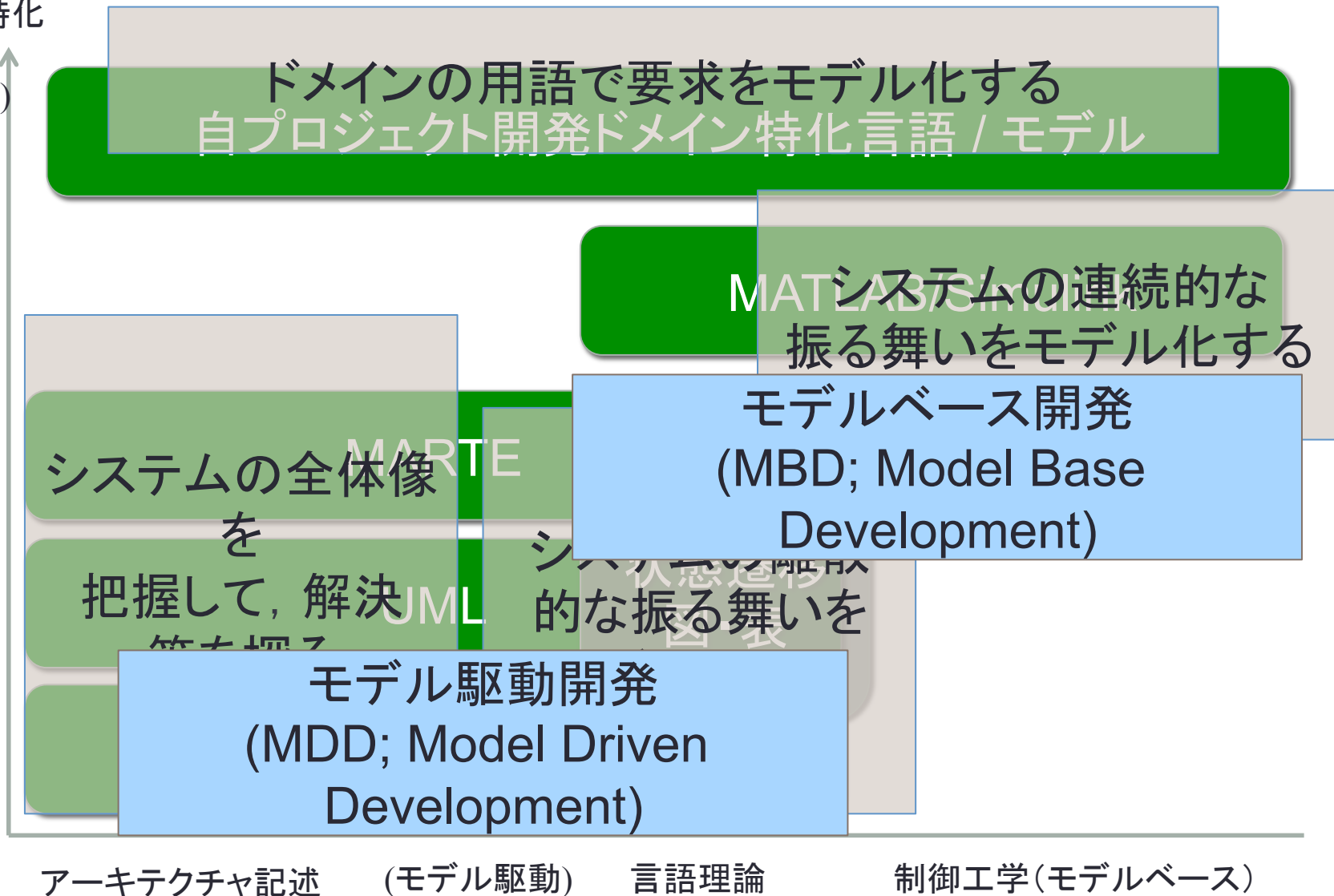
- このUMLプロファイル仕様は、モデル駆動のリアルタイム組み込みシステム (Real Time and Embedded Systems=RTES) 開発に必要な機能をUMLに追加するためのものである。
- UML profile for MARTE (略称 MARTE) と呼ばれるこの拡張は、仕様記述、設計および確認・検証の各段階をサポートする。



NFPs = Non-Functional Properties (非機能属性)
 GRM = Generic Resource Modeling (総称的リソースモデリング)
 GCM = Generic Component Model (総称的コンポーネントモデル)
 Alloc = Allocation modeling (割当てモデリング)
 RTEMoCC = RTE Model of Computation & Communication (RTE計算通信モデル)
 SRM = Software Resource Modeling (ソフトウェア・リソースモデリング)
 HRM = Hardware Resource Modeling (ハードウェア・リソースモデリング)
 GQAM = Generic Quantitative Analysis Modeling (総称的定量分析モデリング)
 SAM = Schedulability Analysis Modeling (スケジューラビリティ分析モデリング)
 PAM = Performance Analysis Modeling (パフォーマンス分析モデリング)
 VSL = Value Specification Language (値定義言語)
 RSM = Repetitive Structure Modeling (反復的構造モデリング)

モデル地図

ドメイン特化
度合い
(抽象度?)

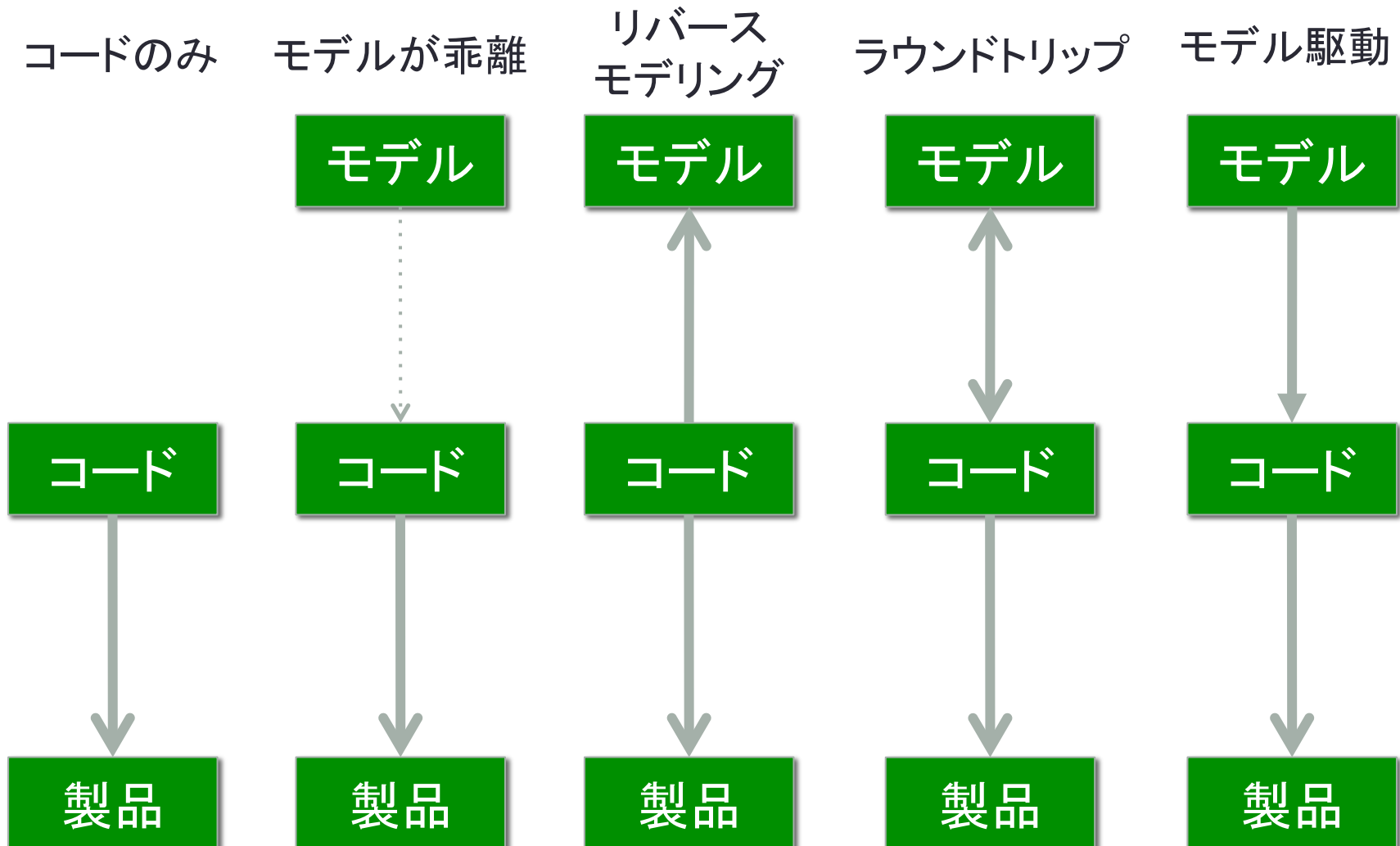


モデルの様々な利用方法

Martin Fowlerは以下の3つに分類

- **スケッチとしてのUML** (UML as sketch)
 - コミュニケーションの道具
- **設計図としてのUML** (UML as blueprint)
 - 設計 → スケルトン生成
 - コードの可視化
- **プログラミング言語としてのUML** (UML as programming language)
 - コード生成
 - シミュレーションによる検証

様々なモデルの利用



FAQ

- 性能が大幅に低下するんじゃないの？
 - ほとんど性能低下はありません。
 - Cでの記述とほぼ同等の性能が得られます。
 - パレートの法則。
- メモリを無駄に消費するんじゃないの？
 - 確かに増えますが大規模ではほとんど問題になりません。
 - たいてい同様のコードを手でコーディングしてます。
 - 大規模開発では手でのコーディングよりもサイズダウンという事例もあるようです。
 - 小規模開発にはちょっときついかもしれません。
- デバッグはできるの？コンパイラのバグに対応出来るの？
 - 手でのコード開発とほぼ同等の可読性が得られます。
 - MDDしている8割の会社はモデルコンパイラに手を入れてません。

いつか来た道...

NATO会議での高級言語に関する議論

4.3.4. HIGH-LEVEL LANGUAGES

The use of high-level languages in writing software systems was the subject of a debate.

*d'Agapeyeff: (from *Reducing the cost of software*)*

»In aiming at too many objectives the higher-level languages have, perhaps, proved to be useless to the layman, too complex for the novice and too restricted for the expert.« I maintain that high-level programming languages have, to this extent, failed.

Fraser: Software is generally written in a low-level language. Has anyone written low-level software in a high-level language? Would you do it again?

*David: (from *Some thoughts about the production of large software systems (2)*)*

»Few large systems have been written in high-level languages. This is not surprising since there are inevitable penalties in size and performance of compiled code, and these factors have been paramount in people's minds. In my opinion, this view is no longer appropriate in many instances since techniques are available to overcome these penalties. Secondary memories can take the squeeze out of the size issue, and performance can be brought to a high level with the aid of a traffic analysis of flow of control in the system. Thus, those modules crucial to performance can become **56** the subject of special attention. Indeed, the vast range of programmer performance indicated earlier may mean that it is difficult to obtain better size-performance software using machine code written by an army of programmers of lesser average calibre.

The advantages of coding in a high-level language lie in increased programmer productivity, fewer bugs in the code, fewer programmers required, increased flexibility in the product, 'readability' of the source code ('self-documentation') and some degree (unspecified) of machine independence (better called 'portability'). Many of these advantages have a face validity, which is fortunate since they are difficult to support with hard evidence. There is evidence, however, on the flexibility issue drawn from the Multics experience. Multics is coded almost entirely in a subset of PL/I known as EPL. The early versions of Multics were large and slow. Major improve-

MDDに利用できるツール

- Clooca
 - モデリング言語のオンラインプラットフォーム
- BridgePoint
 - Executable UML、組込み向けの高品質コード生成
 - クラス図、状態チャート図、アクション言語による完全なるMDD
- Enterprise Architect
 - Professional版:コード埋込みクラス図
 - Ultimate版: コード埋込みクラス図/状態チャート図
 - 体験版あり
- Rhapsody
 - コード埋込みクラス図/状態チャート図

少しだけ Clooca の宣伝

clooca(クルーカ) -Have your language-

www.clooca.com/?lang=ja

cloocaは、実際に製品を開発しているソフトウェア開発者向けのサービスです。特にモデルベース開発に興味がある方は、使っていただいても構いません。あなたはDomain Specific Modeling Languageという言葉をご存知ですか？知らない方は英語ですがスライドも見てください。→What is clooca? DSMLはソフトウェア開発を加速させます。それは完全なソースコードを生成可能な専用モデリング言語です。

cloocaを用いると、ソースコード生成可能なモデリング言語をクラウド上に作ることが出来ます。

まだピンと来ない方もいらっしゃるでしょう、下の3つの具体例を見てください。

3つの事例

for Android

画面遷移図を描いてAndroid用ソースコードを生成する。

はじめる

画面遷移図から、menu、layout用のXMLとJavaのコードを生成します。

デモ

for ETロボコン

ETロボコン2013

ETロボコンにDomain Specific Modelingで挑戦！！

はじめる

for Model Driven Development

Simple Class Diagram

Web上でクラス図を書いてソースコード生成

はじめる

デモ

九州大学大学院システム
情報科学府情報知能工
学専攻社会情報システム
工学コース(QITO)の学
生が起業し開発

<http://www.clooca.com/>



コード生成の仕組み

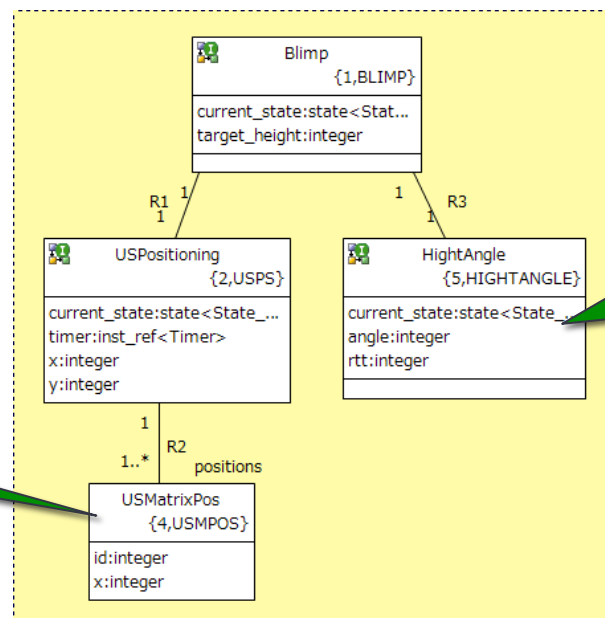
モデルからコードへの変換～様々なレベル

- コード生成(というかモデリング)には構造と振る舞いの両面が必要
- クラス図などからスケルトンコードの生成
 - 詳細な振る舞いはソースコード中に手で記述
 - →簡単に出来そう。一貫性保持が困難。
- クラス図にコード断片を埋込みコード生成
 - 振る舞いはコード断片の形でクラス図に埋込み
 - 完全なコード生成が可能
 - →抽象度が上がっていない。
- クラス図とステートマシン図からコード生成
 - 構造はクラス図、振る舞いはステートマシン図で記述
 - より詳細な振る舞いはコード断片としてステートマシン図に埋込み
 - 完全なコード生成が可能

変換ルールの与え方: マーク

- 指定したモデル要素のみに適用するルール
 - 「指定」のことを「マーク」と呼ぶ
- 特別扱いしたい要素にマークを付ける
 - 永続的/一時的オブジェクト、リモートオブジェクト
 - 起動時、モデル記述以外の世界とのインタフェース

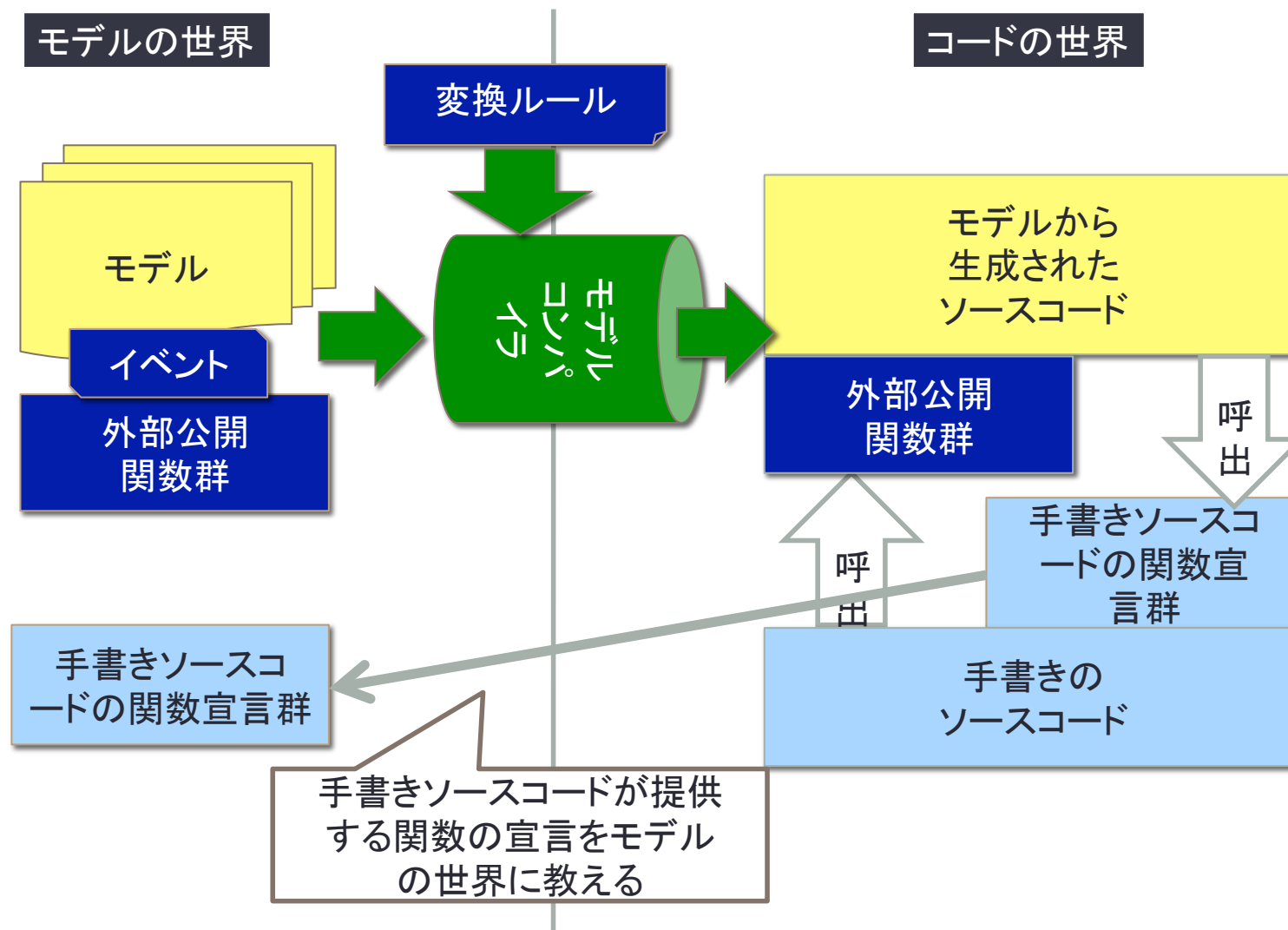
このクラスのインスタンスは電源を切っても覚えておく



起動時にこのクラスのインスタンスを作成して、このメソッドを呼び出す！

モデルの世界 vs. コードの世界

モデルとコードの世界の関係



厳密なモデル定義とモデル変換定義

- 厳密なモデル表記 (MOF、OCL)
- 厳密なモデル変換定義 (QVT)

QVTの例

```
mapping Simple_Class_To_Java_Class
refine Simple_Class_And_Java_Class {
domain {(SM.Class)[name = n, attributes = A] }
body {
  (JM.Class)[
    name = n,
    attributes = A->iterate(a as = {} |
      as + Simple_Attribute_To_Java_Attribute(a))
  ]
}
}
```

MOF: Meta Object Facility

OCL: Object Constraint Language

QVT: Queries, Views, and Transformations

ATL: ATLAS Transformation Language (QVTのRFPを受けてINRIAが開発)

Model-Oriented Programming

The screenshot shows the Umple website homepage. The left sidebar contains various navigation links such as 'Umple Home Page', 'Key Links', 'Downloads', and 'Team'. The main content area features the Umple logo and a navigation bar with buttons for 'UmpleOnline', 'Manual', 'News+Code', and 'Download'. Below this, there is a section titled 'Model-Oriented Programming - Umple.org' which describes Umple as a modeling tool and programming language family. It includes an 'Example' section with a code snippet and a corresponding class diagram.

Umple Home Page

Key Links

- Umple Online
- User Manual
- Google Code site with News
- Wiki
- Blog
- Additional Examples
- Philosophy and Vision
- Tutorials
- Best practices

Downloads

- Official Releases
- Cutting edge command line jar
- Instructions
- License

For Researchers

- Publications and presentations
- CiteULike Umple Group
- Mendeley Umple Group

Social and External Media

- Umple Facebook Page
- Umple Google+ Page
- Umple YouTube Playlist
- Umple Wikipedia Page
- Umple Dribbble Page
- Umple FreeCode Page

Mailing Lists

- Main mailing list
- Help mailing list
- Contributor mailing list

Team

- Umple developer list
- UCOSP

Development status

- Continuous building
- Test report
- Root of code tree
- Grammar
- Class diagram of Umple itself
- Architecture Overview
- Umple Issue Tracker
- Planned development
- Recent changes
- Cheat sheet of commands

Related Sites

- Cruise Home page
- Dr. Lethbridge Home page

Umple

[UmpleOnline](#) [Manual](#) [News+Code](#) [Download](#)

Model-Oriented Programming - Umple.org

Umple is a *modeling tool* and *programming language family* to enable what we call Model-Oriented Programming. It adds abstractions such as Associations, Attributes and State Machines derived from UML to object-oriented programming languages such as Java, C++, PHP and Ruby. Umple can also be used to create UML class and state diagrams textually.

Umple is an open source project, so details will evolve. However, it is ready to be used for real systems. In fact the Umple compiler itself is written in Umple. Any Java, C++ or PHP project could use Umple. We have found the resulting code to be more readable and have many fewer lines. This is because Umple means you can avoid having to code a lot of 'boilerplate' code that would be needed to implement associations and attributes, a system based on Umple should also be less bug-prone.

Umple has also been found to help students learn UML faster in the classroom. Umple works [online](#), as an [Eclipse plugin](#), and as a stand-alone command-line Jar. For further information and downloads, see the left margin.

Example

The following example shows how to declare attributes and associations in the first steps when modeling a system using Umple.

```

01. class Student {}
02.
03. class CourseSection {}
04.
05. class Registration
06. {
07.     String grade;
08.     * -- 1 Student;
09.     * -- 1 CourseSection;
10. }
11.

```

The class diagram to reflect the Umple code above is shown below.

```

classDiagram
    class Student
    class Registration {
        grade
    }
    class CourseSection
    Student "1" -- "*" Registration
    Registration "*" -- "1" CourseSection

```

Many other examples can be found in the [Umple user manual](#), in [UmpleOnline](#) and on the [Umple Examples Wiki page](#).

ArchJava を彷彿

<http://cruise.eecs.uottawa.ca/umple/>



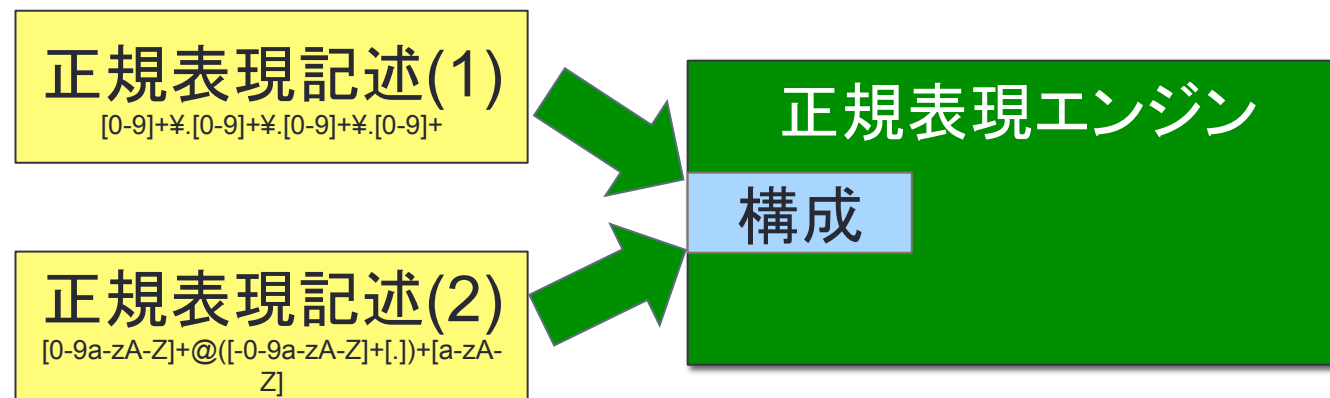
ドメイン特化型開発とDSL

ドメイン特化型開発とは(1)

特定目的向けの言語を設計することによってソフトウェア開発における多くの問題をより簡単に解決する

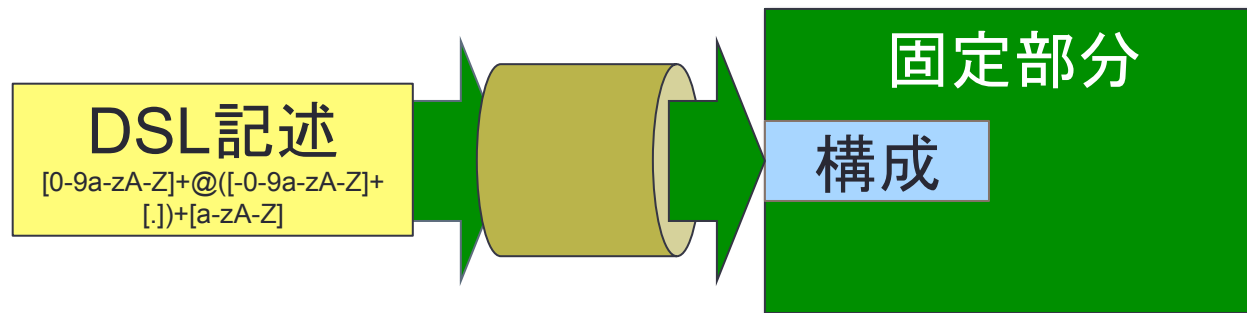
～Steve Cook他:ドメイン特化型開発[1]～

- 例: 正規表現と正規表現エンジン
 - テキストのパターンに対してある処理を行う場合
 - 正規表現がなければ毎回パーサを実装
 - 生産性低い、品質の問題も
 - 正規表現を定義、正規表現エンジンとして再利用
 - 「特定目的向けの言語」→「多くの問題を簡単に解決」



ドメイン特化型開発とは(2)

- プロダクトラインソフトウェア工学の実現手段のひとつ
 - 相違部
 - ドメイン特化型言語(Domain-Specific Language; DSL)で記述
 - DSL記述から構成情報などを生成
 - 共通部
 - 従来手法で開発
- モデル駆動開発のひとつ
 - 要求や設計をドメインに特化したモデル(DSM / DSML)で記述
 - 汎用のモデリング言語よりも記述が容易



DSLのカテゴリ

- 外部DSL
 - ホスト言語とは異なる言語で開発されたDSL
 - 例: Latex、SQL など
- 内部DSL (or 組み込みDSL)
 - ホスト言語がもつ拡張機構を用いて構築したDSL
 - DSL部分以外はホスト言語の機能がそのまま使用できる
 - 拡張機構をもつ言語としてScalaなどがある

ドメイン特化型開発の利点

生産性

- モデルと思考のミスマッチが減少
- アプリケーションの自動生成

品質

- 設計と実装の一貫性保持
- ドメイン特化最適化
- 汎用言語での記述エラーを最小化

コミュニケーション

- 様々なステークホルダがモデルを利用しやすくなる

ドメイン特化型開発の欠点

コスト

- 多品種展開できなければ無意味

教育

- 独自の教育・教育コンテンツが必要
- 汎用言語よりは導入が容易だといわれている

ドメイン特化型開発のプロセス

ドメインエンジニアリング

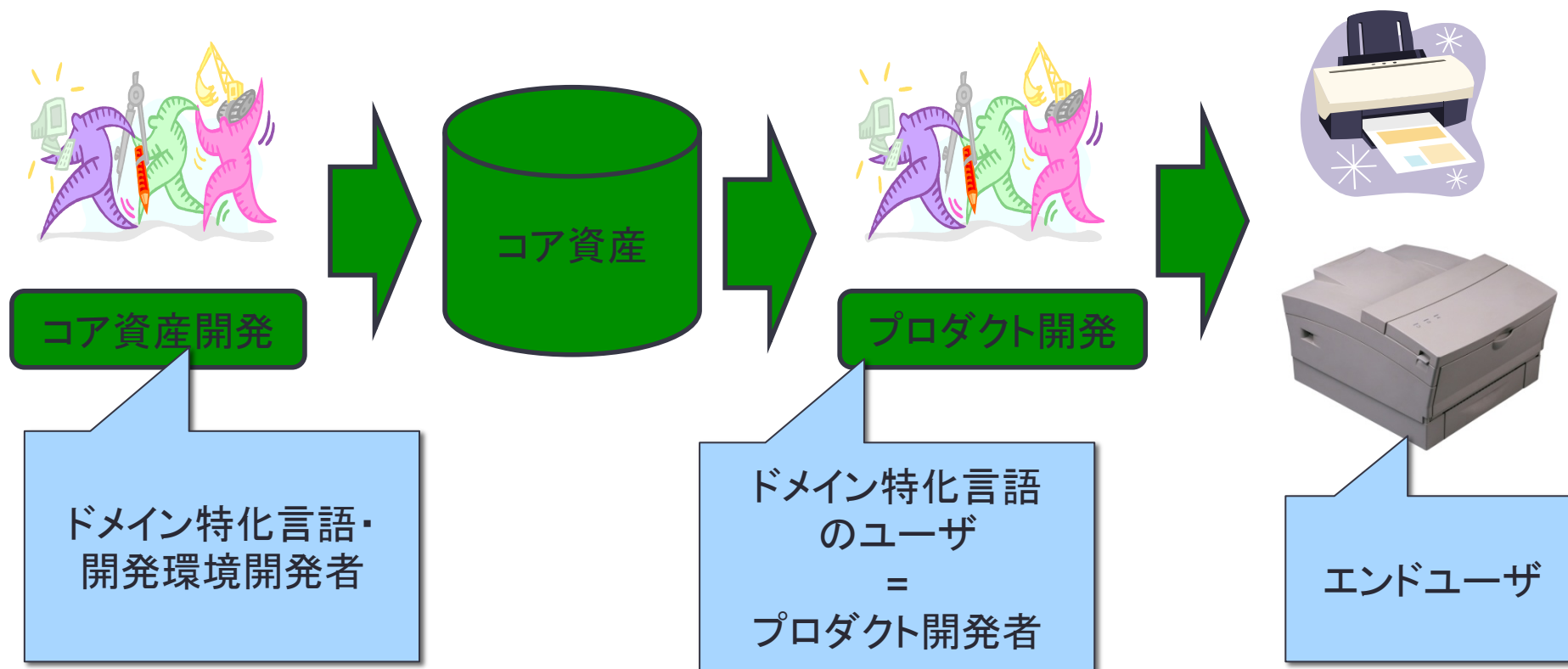
- チームビルディング
- スコーピング
- 可変性の特定
- ドメイン特化モデルの発見
- ドメイン固有記法の定義
- 共通アーキテクチャ設計と実装コンポーネントの確定
- コードジェネレータの開発
- ドメインテスト

アプリケーションエンジニアリング

- ドメイン特化言語によるアプリケーションの記述
- アプリケーションテスト

ドメイン特化型開発のロール

- エンドユーザ
- ドメイン特化言語のユーザ
- ドメイン特化言語・開発環境の開発者



DSL開発環境・ツール

- Microsoft DSL Tools
 - Visual Studio Professional Edition以上
 - VS上で開発できる独自言語を作り出すことができる。
 - モデルエディタ
 - メタモデル・図表現の開発支援
 - ジェネレータの開発支援
 - C#, VB + テンプレートによるコード生成
- EMF/GMF
 - Eclipse上で動作
 - DSL Toolsとほぼ同等
- MetaEdit+
 - モデルエディタ
 - メタモデル・図表現・表の開発支援
 - ジェネレータの開発支援
 - MEALスクリプト言語によるコード生成
- Enterprise Architect
- テキスト系
 - Xtext@Eclipse

モデル駆動開発の導入10個のtips

1. ドメインをしっかりと狭く保つ
 - モデル駆動開発を適用する
2. 十分に理解できているドメインに適用する
3. MDDをクリティカルパスに適用する
 - パイロットプロジェクトでは十分な注意と資源が割かれない・・・
4. MDDは最初から適用すると一番うまくいく
 - MDDに適したソフトウェアアーキテクチャがあり, 後から変更することは難しい
5. ほかの場所で相殺される利益に注意するべし
6. コード生成ばかりに気を取られない
 - 上流でのシミュレーションによる品質向上, 良いソフトウェアアーキテクチャの獲得など
7. 抽象的に考えられる人ばかりではない
8. 多くのプロジェクトは規模の拡大時に失敗する
9. 人やものの考え方にツールやプロセスを適合させる, 逆ではなく
10. そう, たった9つだけだ

By 久住先生(九州大学)

第三部 研究事例



事例1：組込みソフトウェア開発のための外部環境分析

Naoyasu Ubayashi, Yasutaka Kamei, Masayuki Hirayama, and Tetsuo Tamai: A Context Analysis Method for Embedded Systems ---Exploring a Requirement Boundary between a System and Its Context, 19th IEEE International Requirements Engineering Conference (RE 2011), IEEE Computer Society, pp.143-152 (2011).

自動車 CC(クルーズコントロール) 操作マニュアルからの抜粋

- 次のような状況のときはCCを使用しないでください。
使用すると思わぬ事故につながるおそれがあります。
 - 交通量の多い道や急カーブのある道
 - 凍結路や積雪路などのすべりやすい路面
 - 急な下り坂

どうして問題になるのか？

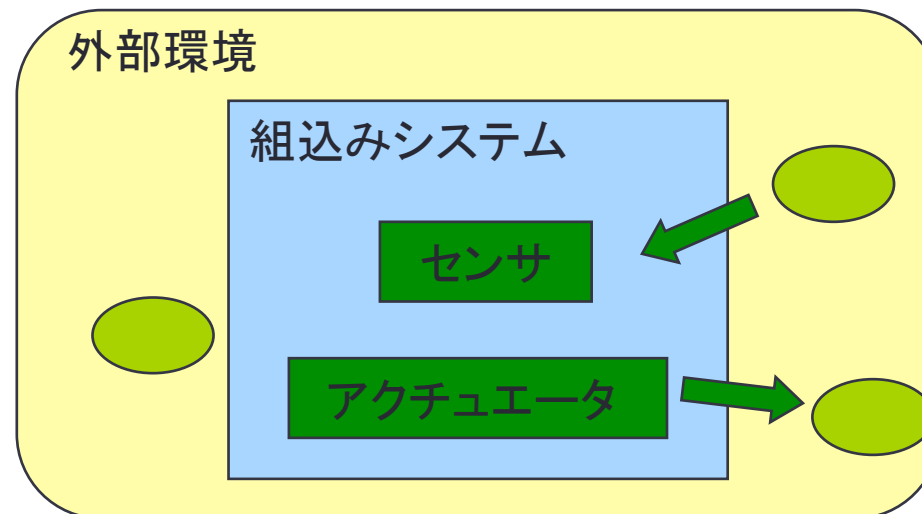
- 交通量の多い道や急カーブのある道
 - 道路の状況にあった速度で走行できないため事故につながるおそれがある
- 凍結路や積雪路などのすべりやすい路面
 - タイヤが空転し、車のコントロールを失うおそれがある
- 急な下り坂
 - エンジンブレーキが十分効かないため、セットした速度を超えてしまい、思わぬ事故につながるおそれがある

安全性と考慮すべき外部環境の境界

- 自動車開発において「安全性」は最も重要なアイテムの一つである
- しかし、すべての外部環境下において本当に安全性を確保できるのか？
 - もしも走行中に、突然、落石があったらどうする？
 - もしも走行中に、橋が崩壊したらどうする？
 - 結局、考えれば切りがない(フレーム問題)
- 考慮すべき(製品として保証すべき)外部環境の境界を決定することが必要

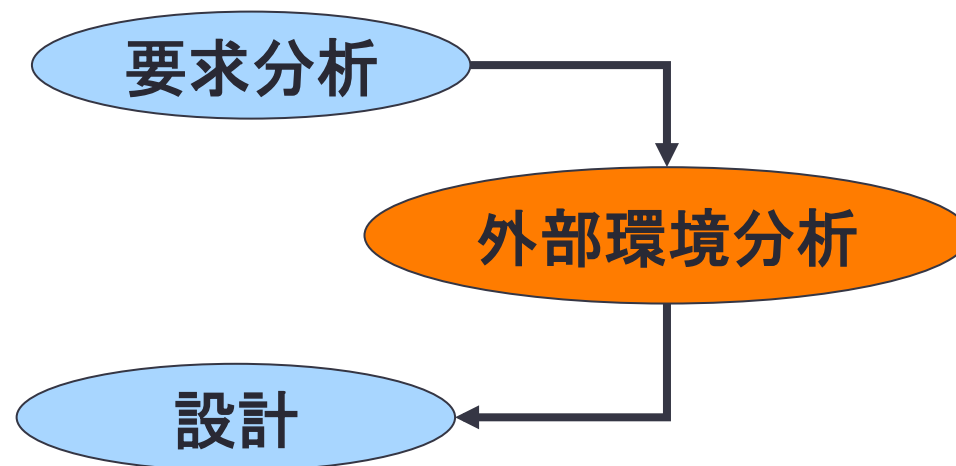
組み込みシステムとは

- 機器に組み込まれて動作するシステム
 - 車載システム、携帯電話、作業用ロボット
- センサ、アクチュエータを介して外部環境とインタラクション

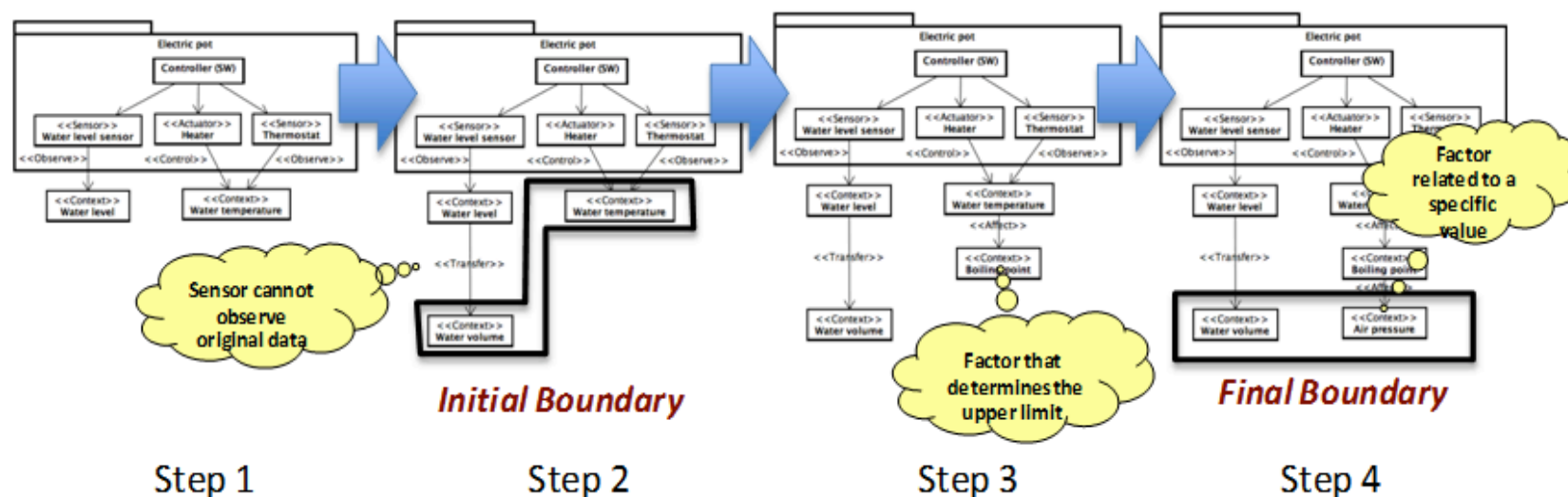


外部環境分析手法 CAMEmb

- CAMEmb (Context Analysis Method for Embedded Systems)
- 要求分析終了時に外部環境を分析する工程を置く
- 外部環境に起因する不具合を減らすことができる



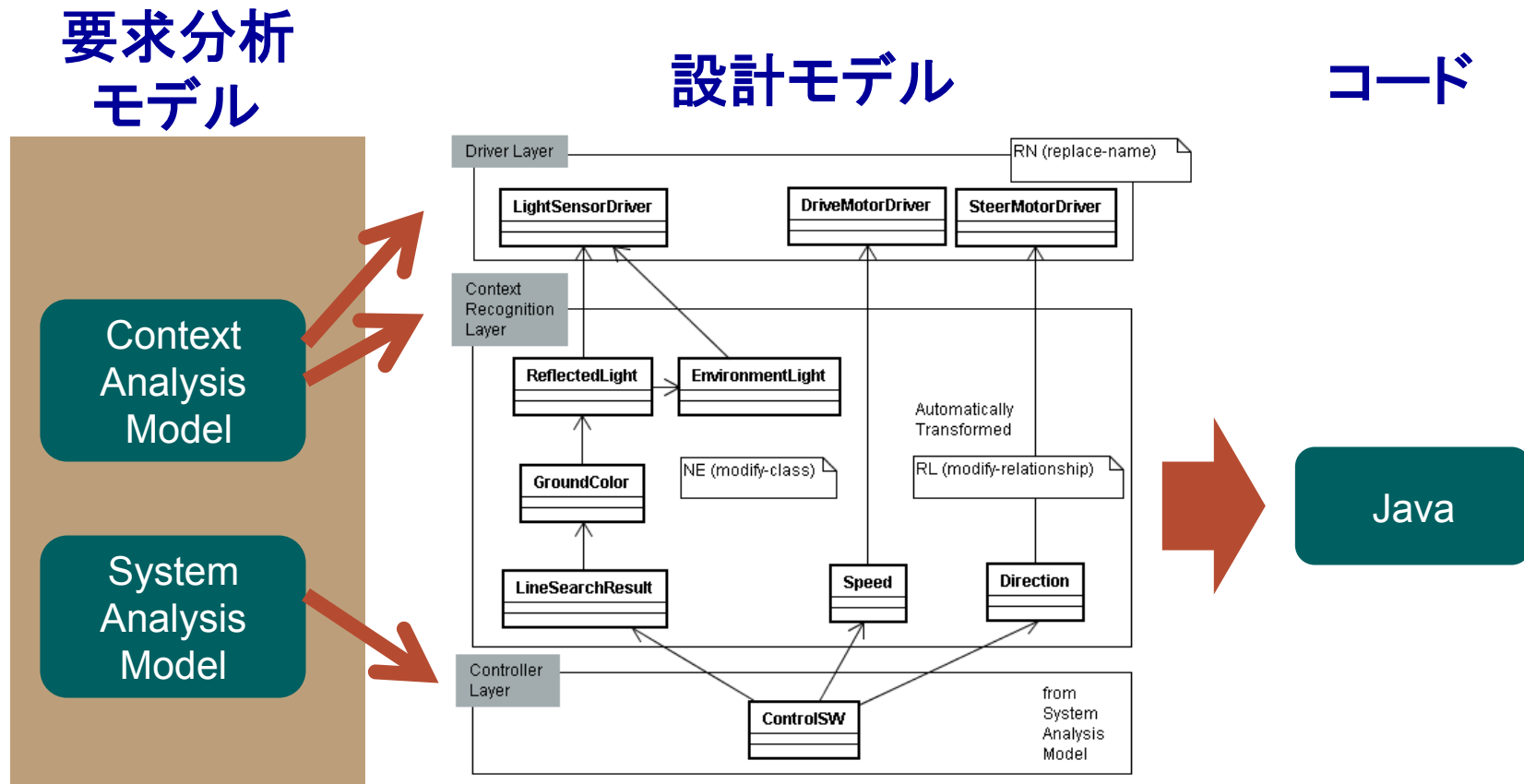
CAMEmbによる外部環境分析



外部環境分析のためのガイドワード

No.	Category of << Affect >>	Guide word
1.	physical phenomena	factor that determines the upper limit
2.	physical phenomena	factor that determines the lower limit
3.	physical phenomena	factor related to a specific value
4.	influence to sensing	factor that interferes with the observation
5.	influence to actuation	factor that interferes with the control

CAMEmbベースのモデル駆動開発



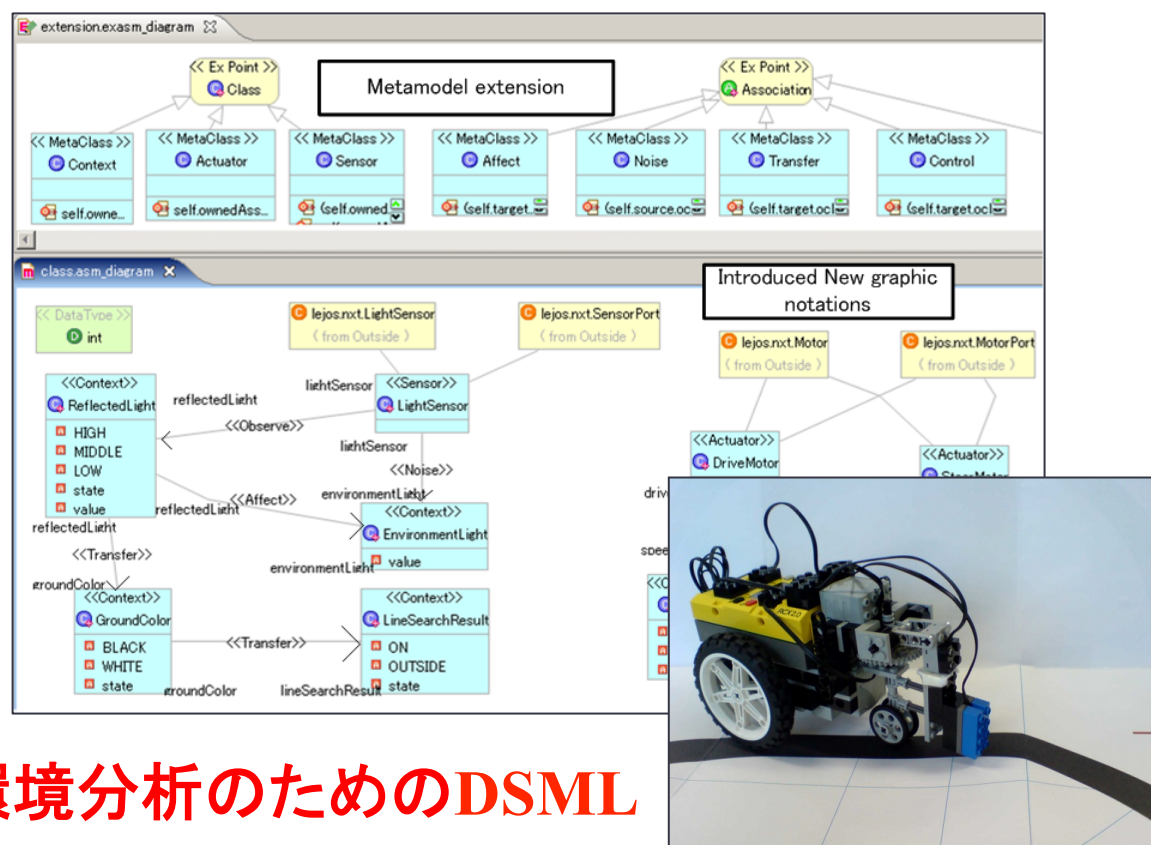
外部環境分析結果をコードに反映(自動化)

開発環境 CAMEmbModeler

ライトレーサの外部環境モデル

- モデルエディタ
 - EMF
 - GMF
- モデルコンパイラ
 - アスペクト指向 (AspectM)
 - Prolog

拡張可能なアスペクト指向
モデリング言語AspectMを使用
してDSLを構築



外部環境分析のためのDSML

Ubayashi, N., Otsubo, G., Noda, K., and Yoshida, J.:
An Extensible Aspect-oriented Modeling Environment, CAiSE 2009, pp.17-31, 2009.



事例2: YET ANOTHER MDD

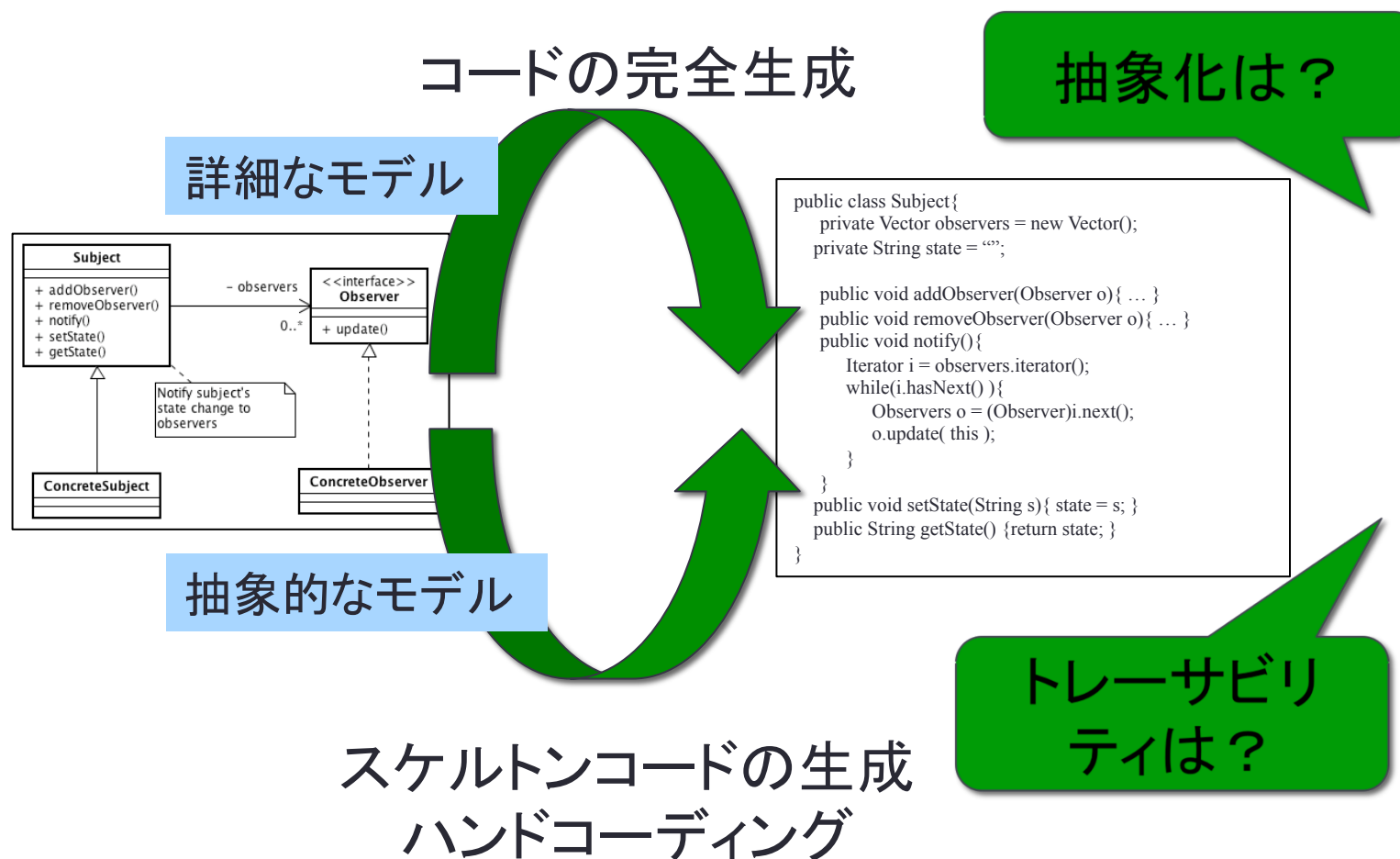
コード生成を仮定しないMDD

Naoyasu Ubayashi, Jun Nomura, and Tetsuo Tamai, Archface: A Contract Place Where Architectural Design and Code Meet Together, 32rd ACM/IEEE International Conference on Software Engineering (ICSE 2010), ACM PRESS, pp.75-84 (2010).

MDDの限界

- DSLが提供される場合はMDDは有効。
- その一方、DSLが提供できない一般的な開発では、コード生成をベースとするMDDには限界がある。たとえば、UMLでモデリングし、Javaで実装するような場合。
- しかし、このような開発形態は多々見られる。これに対する支援はできないものか？
- モデルとは本当にコードの生成元のためだけに存在するのだろうか？
- モデルにはコードとは異なった役割があるのではないか？
- 鍵となるのは抽象化。

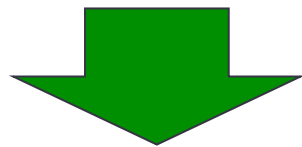
詳細化モデル vs. 抽象化モデル



抽象化, Abstraction, ...

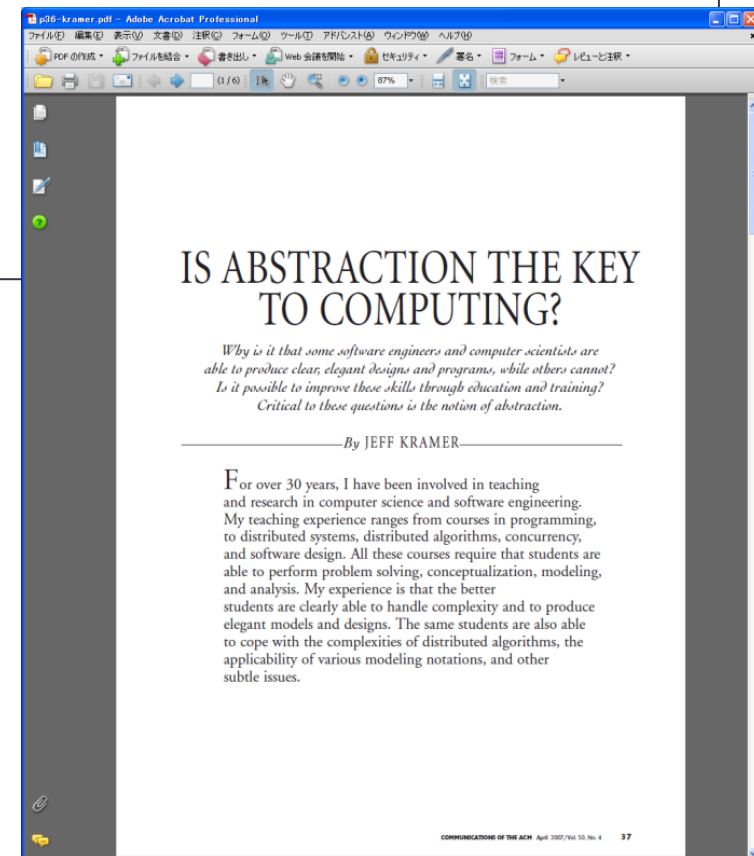
- 明快でエレガントな設計やプログラミングができるエンジニアとそうでないエンジニアがいるのは何故であろうか？
- その答えは抽象化能力にある。

Kramer, J., Is Abstraction the Key to Computing ?
Communications of the ACM,
Vol. 50, Issue 4, pp.36-42, 2007.

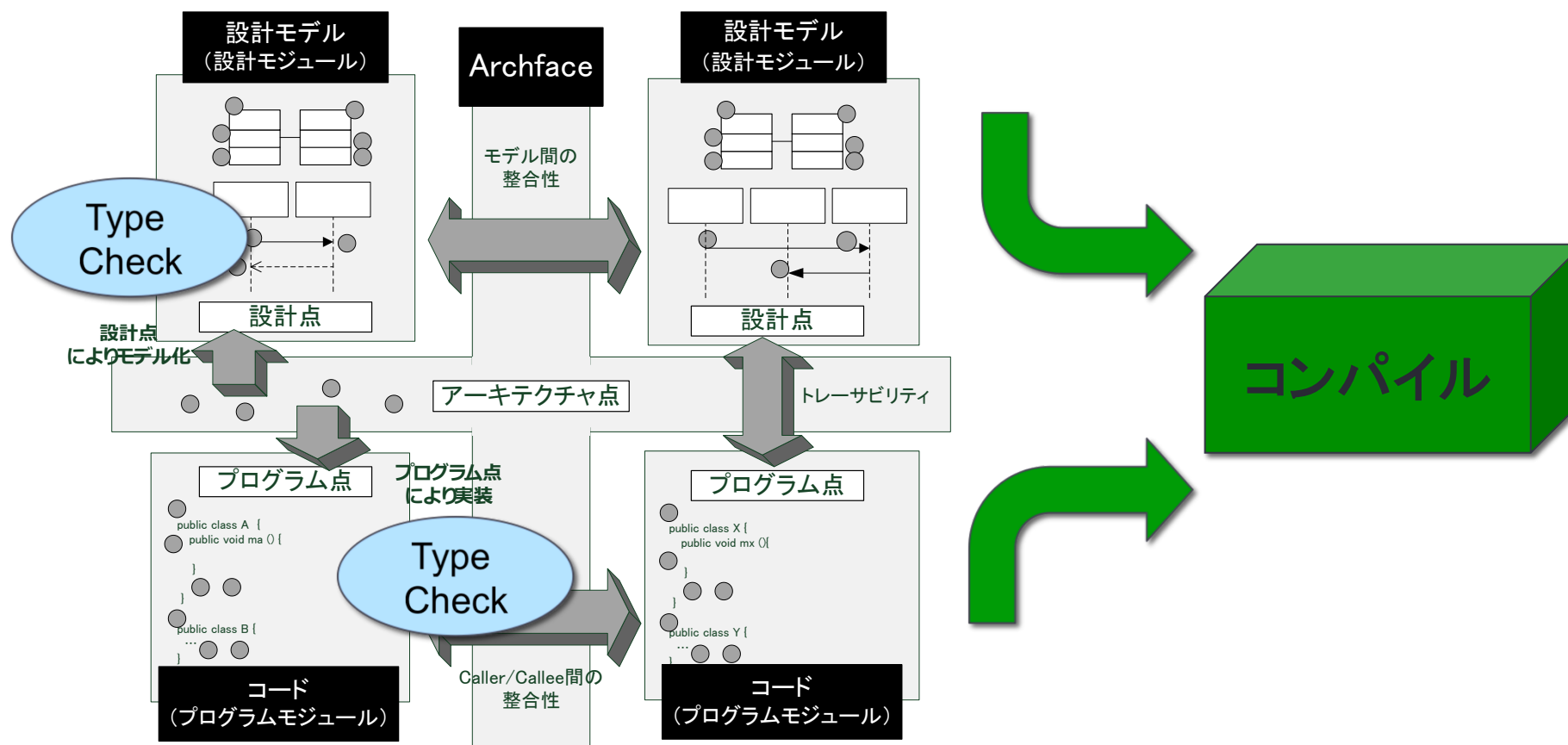


我々のアプローチ

- モデルを抽象化記述のためのSWモジュールと見なす
- モデルとコードはコンパイラの対象
- コンパイラをパスすればモデルとコードのトレーサビリティは保証

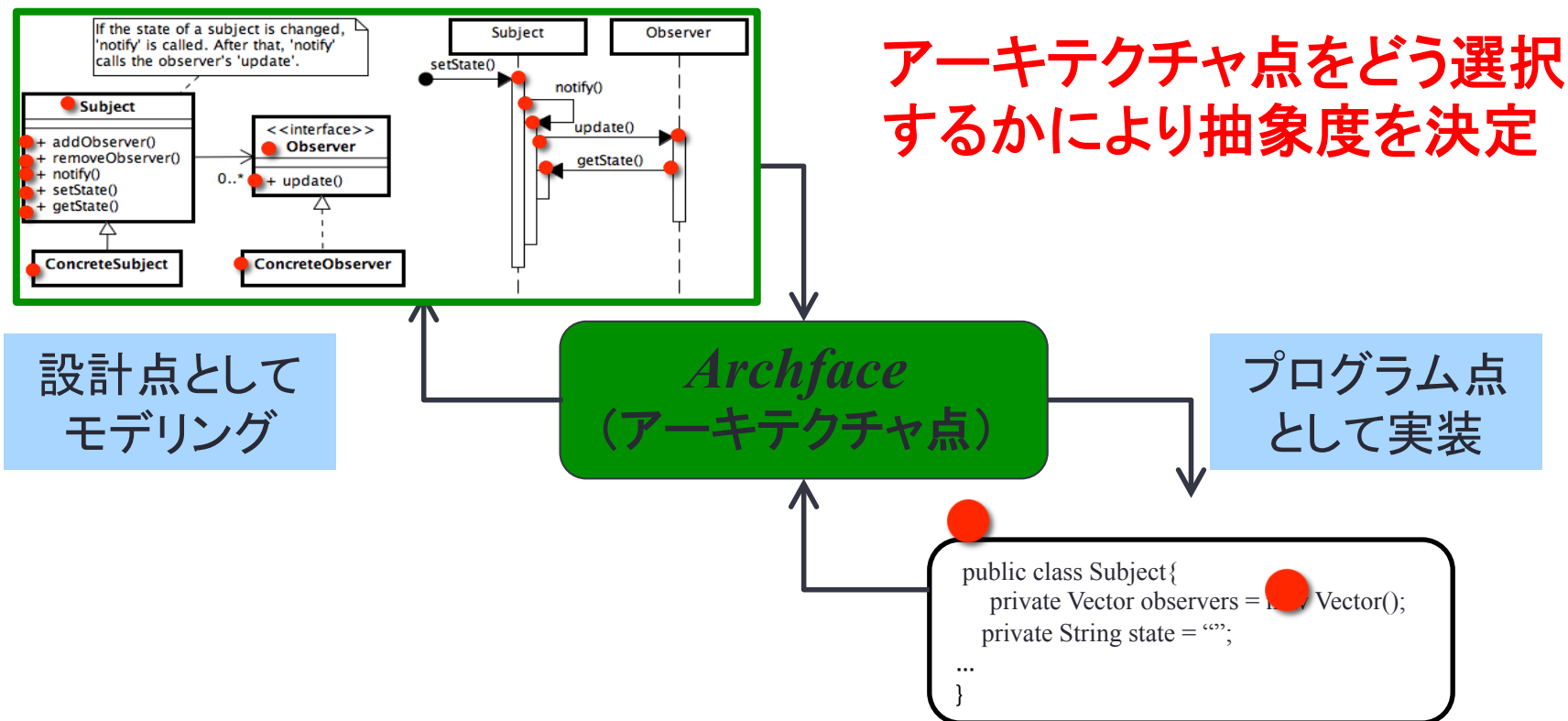


インタフェース機構 Archface



Naoyasu Ubayashi, Jun Nomura, and Tetsuo Tamai, Archface: A Contract Place Where Architectural Design and Code Meet Together, 32rd ACM/IEEE International Conference on Software Engineering (ICSE 2010), ACM PRESS, pp.75-84 (2010).

Archface = 設計 + プログラム インタフェース



トレーサビリティチェック

設計は
Archfaceを
モデル化しているか？

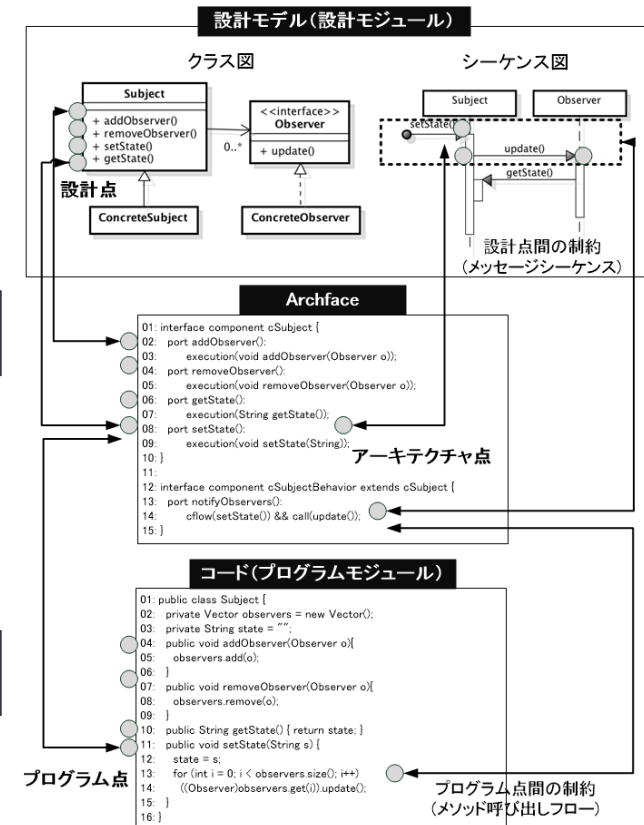
Type
Check

SMT Solver

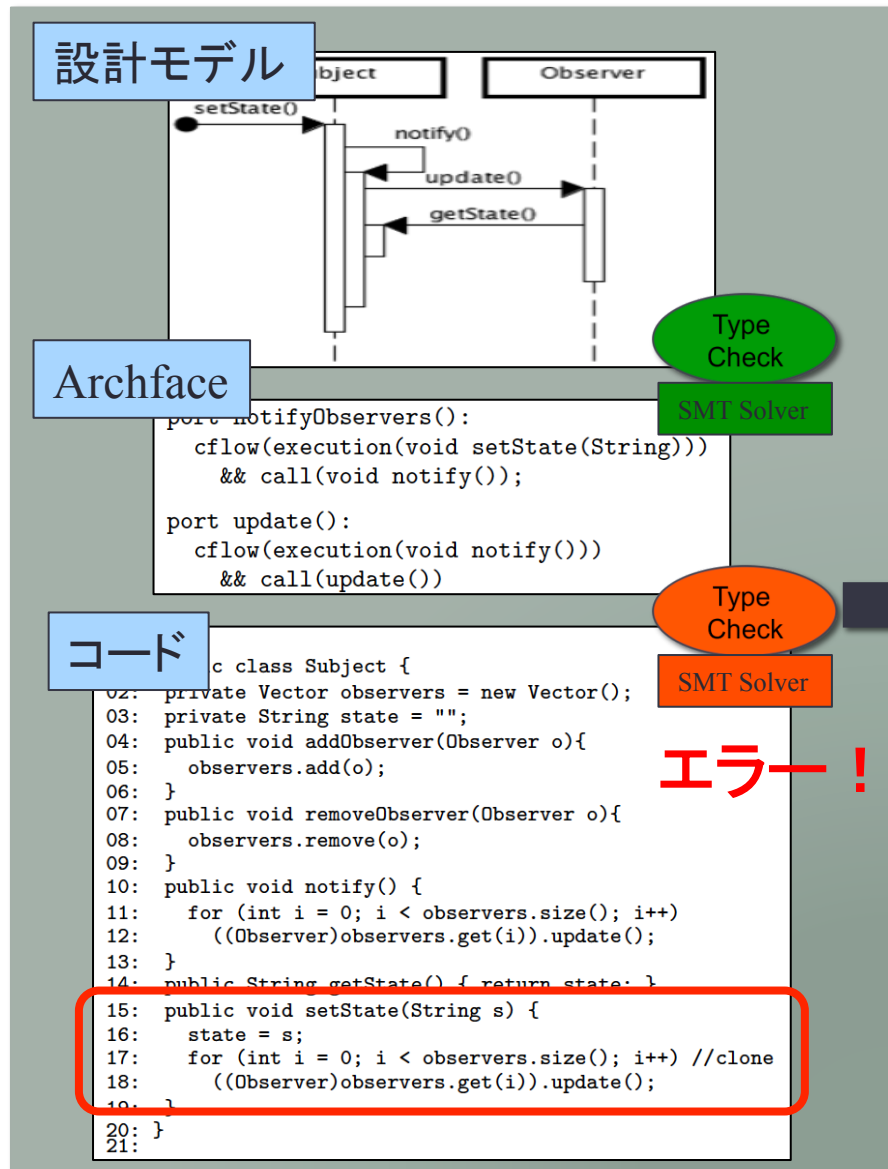
コードは
Archfaceを実装してい
るか？

Type
Check

SMT Solver



抽象化洗練の例



エラーへの対処方法

1. 設計の方が正しいと判断しコードを設計モデルに合わせる
2. コードの方が正しいと判断し設計モデルをコードに合わせる
3. **抽象レベルの設定に問題があると判断し設計モデルの抽象度を変更する**

port notifyObservers():
 cflow(execution(void setState(String)))
 && call(update());

Type
Check

SMT Solver

これに伴い設計モデルも変更

iArch: Archface中心のIDE

The screenshot displays the iArch IDE interface. The main workspace is divided into several panes:

- Project Explorer:** Shows the project structure with 'Observer Pattern' and 'test' folders.
- Diagram Editor:** Displays a sequence diagram with three lifelines: Actor, : Subject, and : Observer. The messages are: Actor to Subject (setState), Subject to Observer (update), and Observer to Subject (getState). A palette on the right provides tools for creating and editing diagram elements.
- Code Editor:** Shows the implementation of the Observer pattern in Java. The code defines a Subject interface with methods registerObserver, unregisterObserver, notify, getState, and setState. It also defines an Observer interface with an update method. The Subject class implements these methods, and the Observer class implements the update method.
- Problems View:** Located at the bottom, it shows a list of errors and warnings. One error is highlighted: 'Subject sequence order is invalid' in the 'Sequence_Invalid.diagram' file.

```
interface Subject {
    void registerObserver(Observer observer)
    void unregisterObserver(Observer observer)
    void notify()
    String getState()
    void setState(String state)
}

interface Observer {
    void update()
}

Subject = (Subject.notify->Observer.update->Subject).
Observer = (Observer.update->Observer).
```

EMF + Graphiti + Xtext で開発

第五部 今後の動向

MDDのためのリポジトリ ReMoDD

The screenshot shows the ReMoDD website interface. The main content area is titled "Repository for Model Driven Development (ReMoDD) Overview". It includes a search bar, a user login section, and several navigation menus. The central text describes the repository's purpose and lists various initiatives and resources.

Major MDD Initiatives and Resources

- The ATL Project
- EMF: The Eclipse Modeling Framework
- The Epsilon Project
- GeMOC: Globalization of Modeling Languages
- GME: The Generic Modeling Environment
- Open Model Initiative
- Open Model Initiative (Austria)
- The Papyrus project
- The SSELab

<http://www.cs.colostate.edu/remodd/v1/>

今後のモデリングコミュニティ

- オープン化
 - The Open Model Initiative (<http://www.openmodels.org>)
 - 誰もが、コピー、利用、修正、配布できるオープンな参照モデルを共同で開発するプロジェクト
 - オープンソースコミュニティのモデリング版
- DSL開発の容易化加速
 - 様々なプロジェクト
 - EMF (Eclipse Modeling Framework)
 - GMF (Graphical Modeling Framework)
 - Graphiti (Graphical Tooling Infrastructure)
 - Xtext (Framework for Development of Programming Languages and Domain Specific Languages)
 - ATL (ATL Transformation Language)
 - Papyrus

ホットな研究トピック

- スケッチとしてのモデリングを支援する環境
- 不確かさを含むモデル駆動開発
- トレーサビリティ
- 双方向変換
- Model@Runtime
- LOP (Language-oriented programming)
- Model-Oriented Programming – Umple.org

- 参考資料
 - チュートリアル「モデル駆動工学の原理と応用」(日本ソフトウェア科学会誌「コンピュータソフトウェア」)

おわり



謝辞：本資料作成にあたり、九州大学 久住憲嗣准教授に多大な協力をいただきました。