ソフトウェアファクトリの定義と分類法v.7
Software Factory Taxonomy

松本　吉弘
京都高度技術研究所
工学博士、IEEE Life Fellow
http://www5d.biglobe.ne.jp/

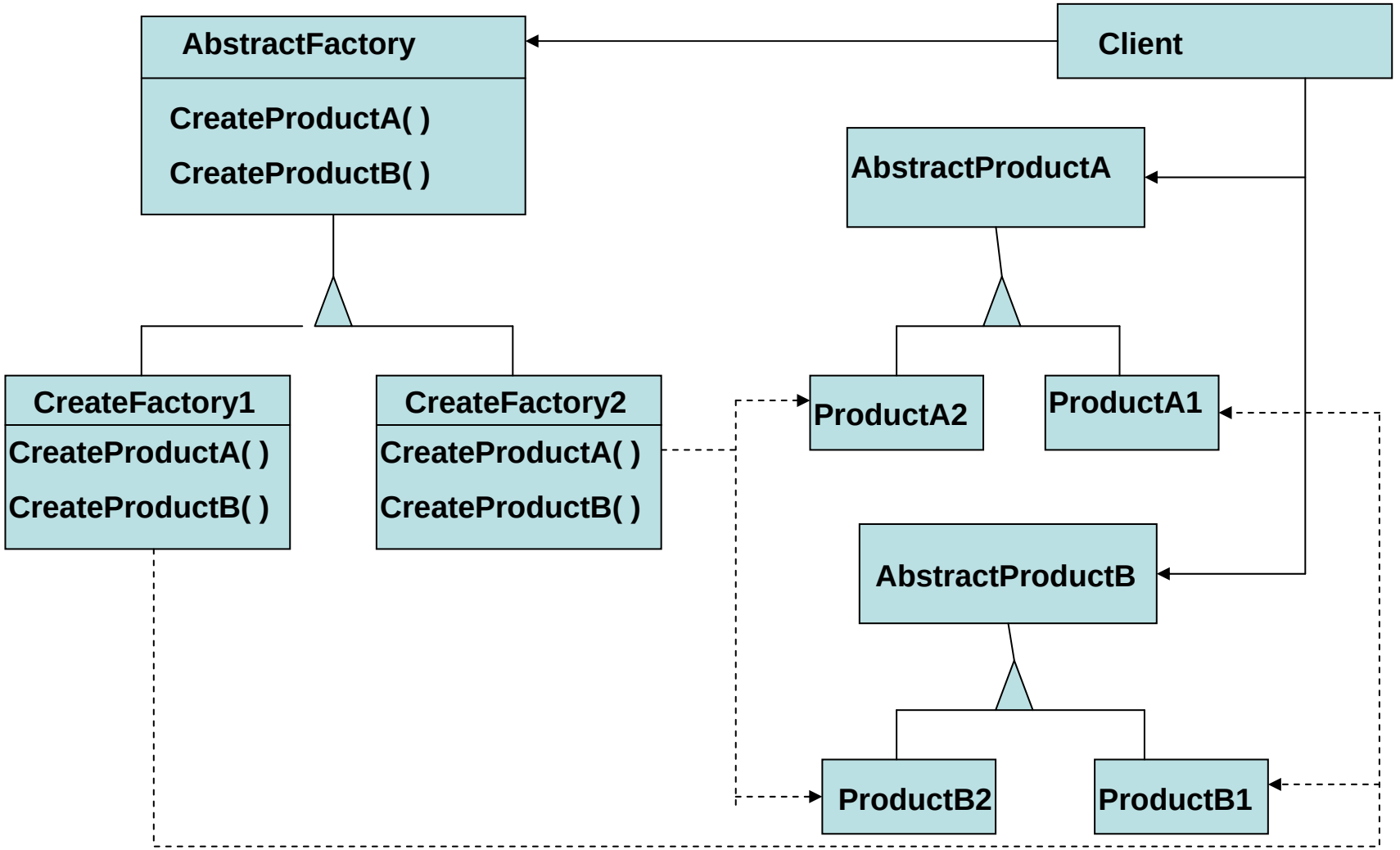
目次

- 第1章　基礎事項
- 第2章　ソフトウェアプロセスの工業化
- 第3章　垂直ソフトウェアファクトリと水平ソフトウェアファクトリ
  （Dr. Paul Clements (SEI/CMU)および筆者の合議に基づく。）
- 第4章　フレームワーク1層および固定プラットフォームの場合のソフトウェアファクトリ
  （電子ポット・ソフトウェアファクトリを例とした開発の初歩解説）
- 第5章　フレームワーク多層、固定プラットフォームの場合のソフトウェアファクトリ
  （マイクロソフト社「Software Factories」）
- 第6章　フレームワーク1層、可変プラットフォームの場合のソフトウェアファクトリ
  （東芝ソフトウェアファクトリ）

第1章　基礎事項

GOFの抽象ファクトリ・パターン

Gamma, E., et al, Design Patterns, Addison-Wesley (1995)



- **ある特定された範囲 (scope)のなかにあるアプリケーション領域 (application area: 後に定義)に関して、適用するように形成された具象ファクトリ (concrete factory) である。**
  - **抽象ファクトリ・パターンにおける concrete product のクラス集合は、プロダクトラインによって管理され、再利用される。**
  - **抽象ファクトリ・パターンにおける createProduct プロセスは、プロセスラインによって管理され、再利用される。**
    - **プロセスラインは、つぎの性質に従って分類される。**
      - **(1) quality level, (2) reusability level, (3) standardization level, (4) automation level, (5) knowledge and skill level, (6) time management, (7) cost management, (8) risk management, (9) communication management, (10) human resource management, (11) supply and acquisition management, and (12) software life cycle level**

概念とは　－　ドメインを定義するために必要

概念とは(concepts)

- 概念とは [清水83]:
  - カテゴリ内の事物の共通特性を抽象したもの
  - 事物のカテゴリ名を指すもの
- 概念をモデルの視点から表現するための理論
  - 定義的特性理論 (Propositional representation)
    - カテゴリは、カテゴリごとに個別に必要、かつ十分な定義的記述集合によって表象され、これによってその概念が示すカテゴリへの所属が、明確に決定できる。
  - 特徴的特性理論 [Smith81]
    - カテゴリは、特性の分布に由来する不均質な内的構造をもち、境界は不鮮明であり、family resemblance 、または確率的モデルによって識別する。
    - 特性の型に、Features（定性的）、およびDimensions（定量的）があるとする。

- 理論ベースの概念理論 (Exemplar modeling)

ドメインとは

ドメインとは

- ドメインとは、概念 (concepts) および知識 (knowledge) の、ある特定された領域 (area)のこと
- ドメインを定める目的:
  - ステークホルダを特定し、その満足度を最大にするため
  - 必要な資源を特定し、生産性およびプロダクト品質を最大にするため
- ドメインに含まれるもの:
  - この領域の実践分担者が理解できる概念
  - この領域に存在する問題を解決するために必要な知識

モデルとは

- **論理システム**は、論理式、論理構成要素、それらの論理式への写像、意味要素とそのベキ集合によって定義される。
- **モデル**は、論理式を満たす（論理式を真にする）ように写像することが可能な意味システム（意味要素、またはそのベキ集合が関係付けられた体系）によって定義される。

仕様（論理システム）代数、モデル代数

- **Finite Automata**
- **Universal Turing machine**
- **Nerve net**
- **Petri Net**
- **Data flow machine**
- **CCS (Calculus of Communication System) by Prof. R. Milner**
- **CSP (Communicating Sequential Processes)**
- **Z**

第2章　ソフトウェア工業化プロセス

設計パラダイムの変遷

	これまでの設計パラダイム	これからの設計パラダイム
設計プロセス	計画駆動、線形、連接	繰り返し、発見的、やってみて拙ければやり直す
目標	最適化	適応性、融通性、即応性、可変性
問題解決手段	形式性、証明可能、	レビュー、検査、運用経験による学習、リファクタリング、リエンジニアリング、進化型改善
周囲の状況	安定、予測可能	変動、不可解、予測不能
適応方法	一発解決を目標	繰り返し適応を前提
基本特性	すべてコントロールできるはず、実装以前に設計がなければならないという思想	協業、コミュニケーション重視、設計と実装を区別しない、ファシリテーション重視
根底思想	技術優先、普遍化可能	不確実的、主観的
頼れるもの	論理的、科学的アプローチ	実践経験による学習、および適応のためのガイドライン

レールを敷く

計画駆動型開発

Plan-driven type development

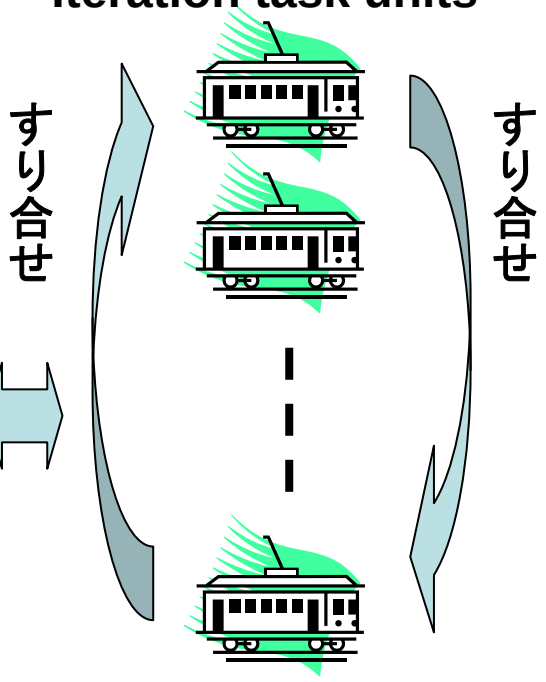
線形、または連接型レールを敷く。

漸次拡張型、進化型、アジャイル型開発

Incremental, evolutionary, agile type development

イテレーション単位

Iteration task units

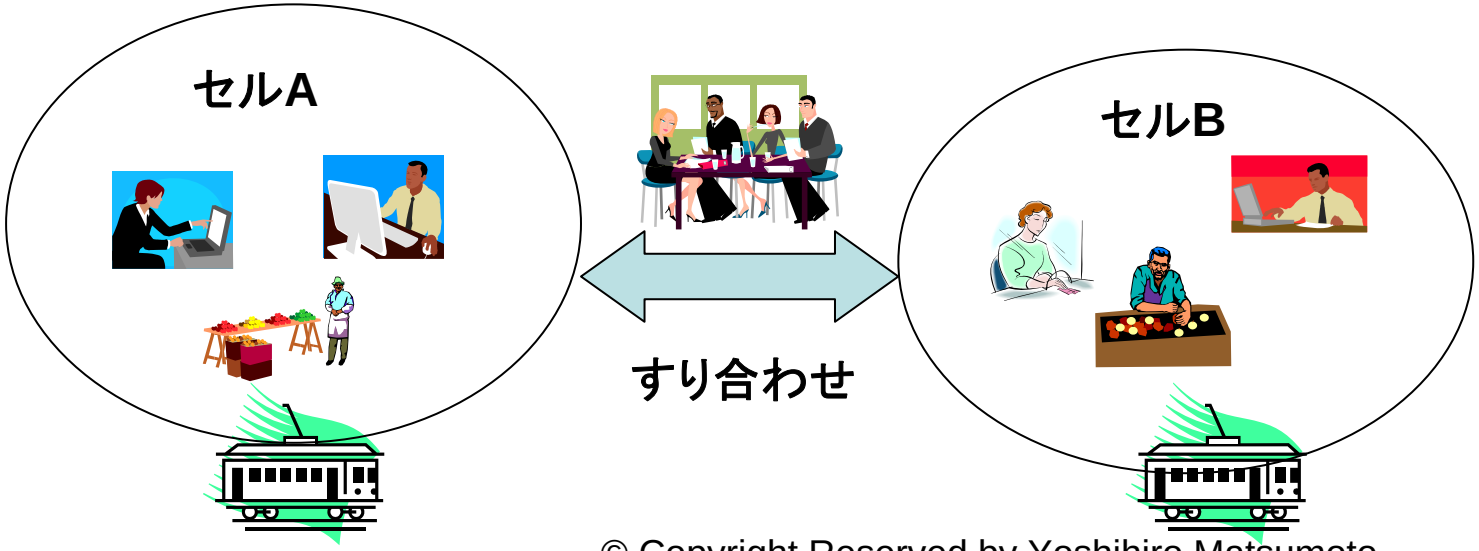


顧客 ⇔ ストーリ Stories ⇔ プロセス、アクティビティ、タスク ⇔ モジュラ化 Modularization 線形列と非線形列を識別 ⇔

イテレーション単位のなかにレールを敷く。

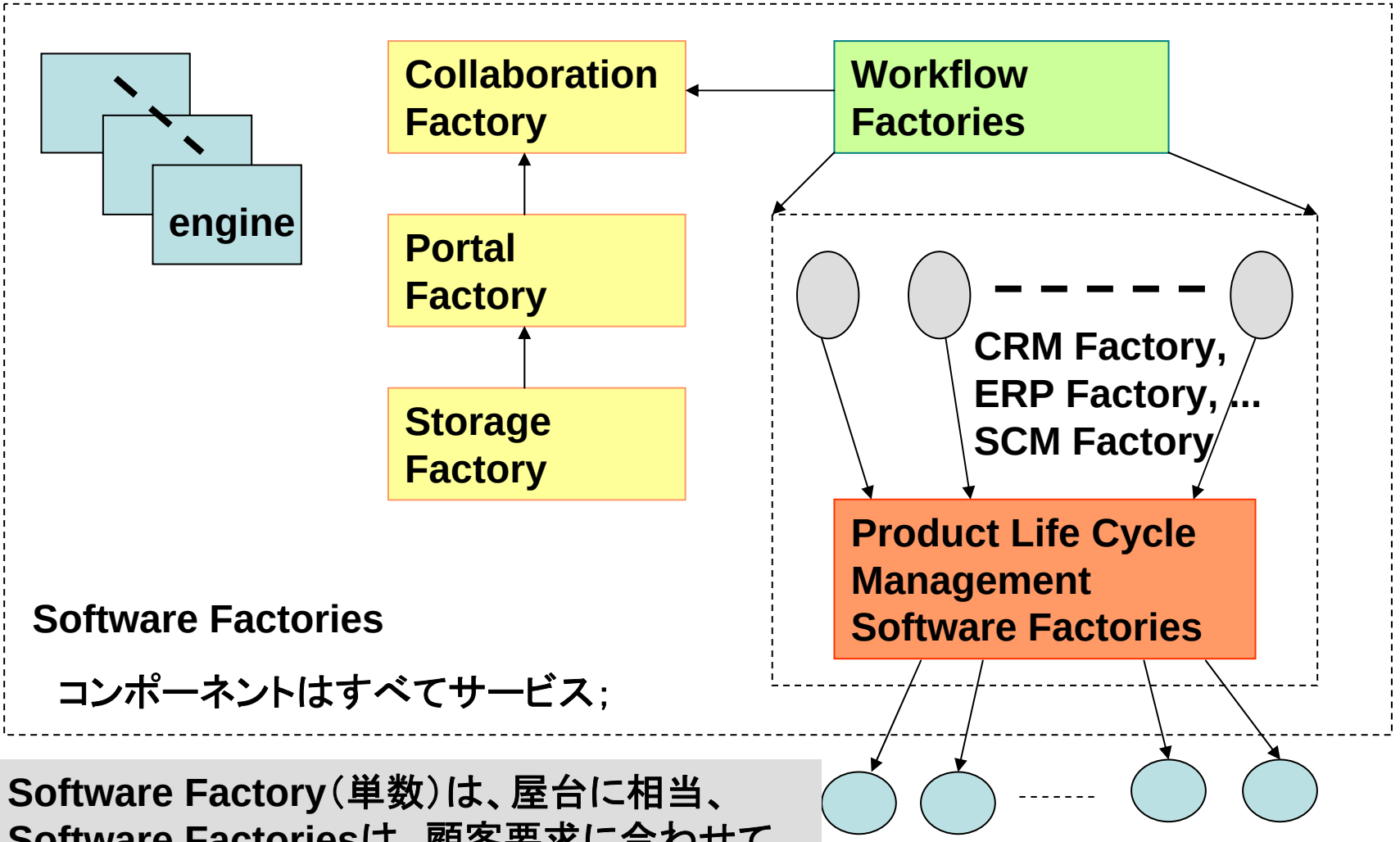
家電機器セル生産方式からのヒント

- 生産企画部は、フィーチャが類似したイテレーション単位ごとに、レール（部品、型、支援ツールと作業順序指導書）を設定した屋台を用意する。
- 具体的な製品注文が到着すると、生産企画部は、屋台のなかのレールをテーラリングし、屋台の配置を決定する。
- プロジェクトマネージャは、イテレーション単位を決定し、それぞれに技術者を貼り付ける。これが「セル」である。



補足： Jack Greenfieldと面談 (2008-1/22)

Software Factories Platformのアーキテクチャスタイルと実装トポロジ



Software Factory（単数）は、屋台に相当、
Software Factoriesは、顧客要求に合わせて
屋台を配置し、組み合わせたセル生産システム。

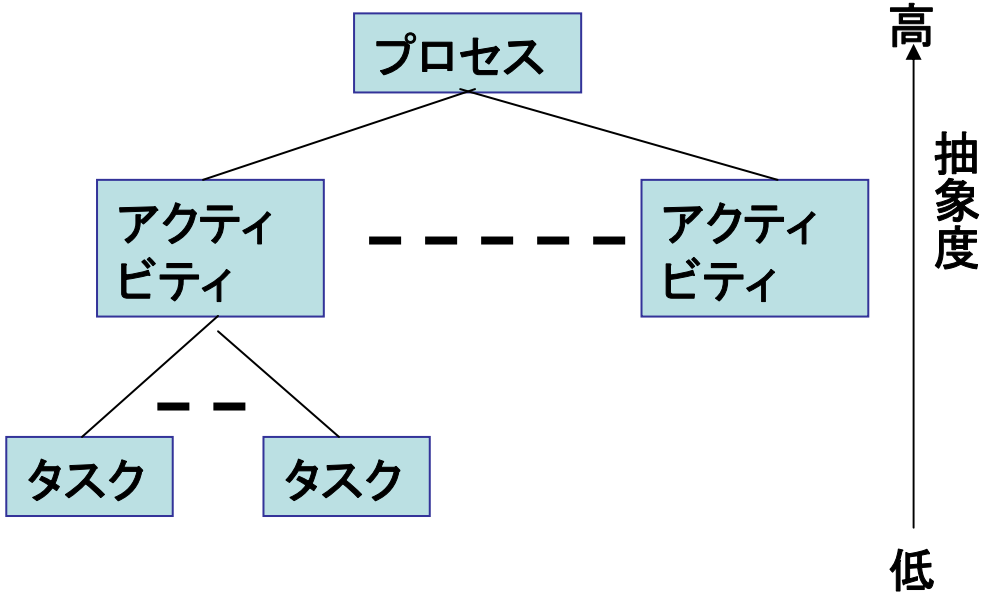
イテレーション単位の階層

イテレーション単位　＝

（同じ抽象度のプロセス、アクティビティ、またはタスクをグループにした作業単位）

イテレーション単位の内部では、作業が繰り返される。

抽象度の高いイテレーション単位の繰り返しのなかで、それより抽象度の低いイテレーション単位のすり合わせが行われる。



イテレーション単位とその階層を同定する： DSM (Design Structure Matrix) を利用する。
N-chartsと呼ばれることもある

			1	2	3	4	5	6	7	8	9	10
A	A0	1	*									
	A1	2	×	*	×							
	A2	3	×	×	*							
B	B1	4		×	×	*						
	B2	5		×	×		*	×				
C	C1	6		×	×		×	*				
	C2	7		×	×				*			
D	D1	8					×	×		*		
	D2	9				×				×	*	
	D3	10							×		×	*

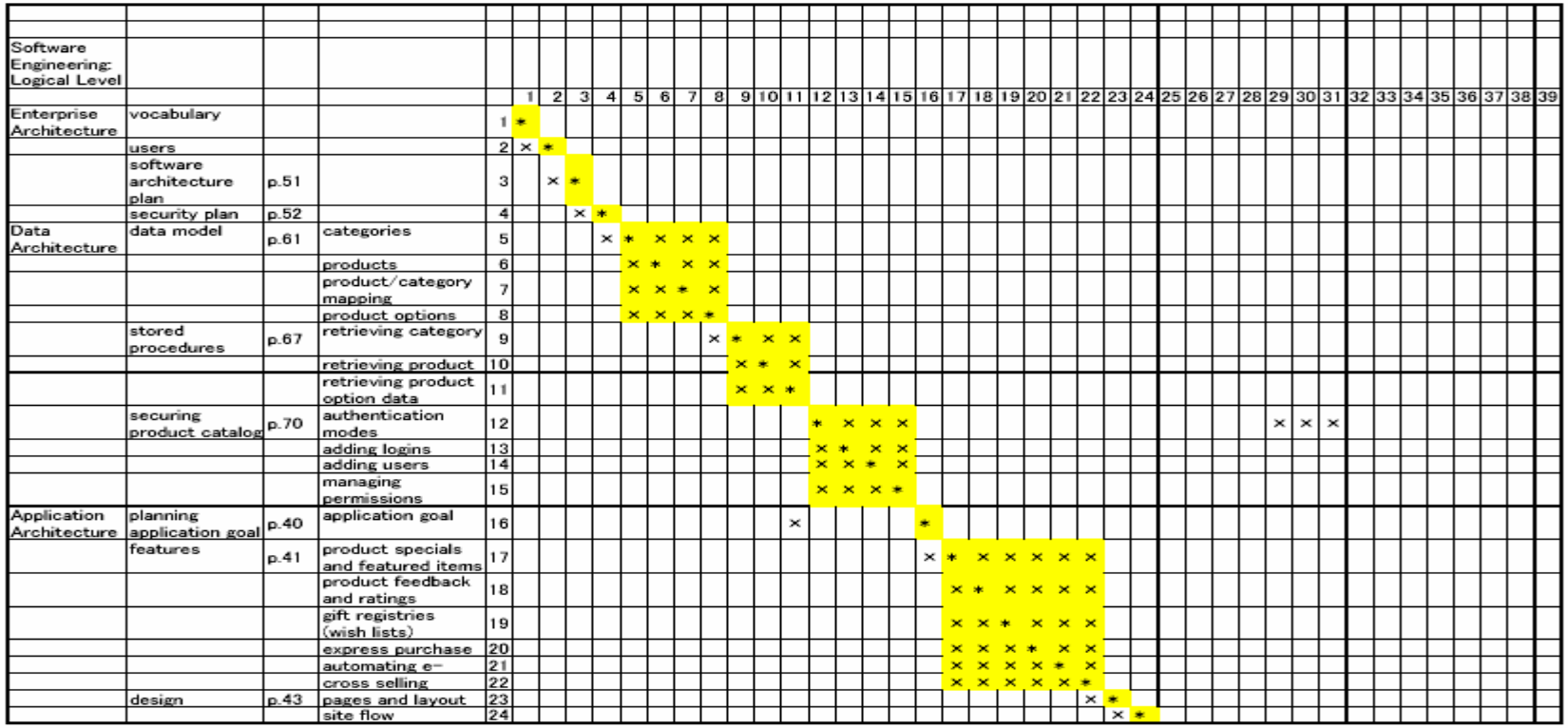
レベル2（レベル1よりも抽象度が低い）へ向けて、プロダクトを分割・詳細化し、詳細化された要素の間のディペンデンシをDSMによって記述する。

e-Commerceプロセスの例（概念レベルのDSM）

Software Engineering: Conceptual Level DSM

Software Engineering: Conceptual Level				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Enterprise Architecture	Software Requirements Specification	context and integration requirements (integration with exisiting data and services)	1	*	×																						
		purposes	2	×	*																						
		stakeholders and organizations	3		×	*	×																				
		business architecture	4			×	*																				
		required software functions	5				×	*																			
		required performance	6					×	*	×	×																
		required quality	7						×	*	×																
		required security	8						×	×	*																
Data Architecture	information entities/accesses	data model	9	×	×	×	×					*	×	×													
		information accesses (stored procedures)	10					×				×	*	×													
		securing informations	11								×	×	×	*													
Application Architecture	application goal	planning application goal and specific features	12	×	×	×	×								*												
	business/software functional	design	13												×	*	×	×	×								
	user–process/activity model	design	14													×	*	×	×								
	event/activity model	design	15													×	×	*	×								
	intrinsic/aspectual functional seperations	design	16													×	×	×	*								
	integration	integration with existing data and services	17	×															×	*							
Technical Architecture	activity/role model		18													×	×	×			*						
	role/organization /location model		19																		×	*					
	enabling and environments	security implementation	20								×			×					×			×	*				
		quality implementation	21						×	×									×					*			
		user information management	22			×						×							×						*		
		deployment	23									×			×							×				*	
		development tools, language, database servers and application framework	24																							×	*

e-Commerceプロセスの例（論理レベルのDSM）



					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Software Engineering: Logical Level																																											
Enterprise Architecture	vocabulary			1	*																																					
	users			2	×	*																																					
	software architecture plan	p.51		3		×	*																																			
	security plan	p.52		4			×	*																																		
Data Architecture	data model	p.61	categories	5				×	*	×	×	×																														
			products	6					×	*	×	×																														
			product/category mapping	7					×	×	*	×																														
			product options	8					×	×	×	*																														
	stored procedures	p.67	retrieving category	9								×	*	×	×																											
			retrieving product	10									×	*	×																											
			retrieving product option data	11									×	×	*																											
	securing product catalog	p.70	authentication modes	12												*	×	×	×														×	×	×						
			adding logins	13												×	*	×	×																						
			adding users	14												×	×	*	×																						
			managing permissions	15												×	×	×	*																						
Application Architecture	planning application goal	p.40	application goal	16											×					*																					
	features	p.41	product specials and featured items	17																×	*	×	×	×	×	×															
			product feedback and ratings	18																	×	*	×	×	×	×															
			gift registries (wish lists)	19																	×	×	*	×	×	×															
			express purchase	20																	×	×	×	*	×	×															
			automating e-	21																	×	×	×	×	*	×															
			cross selling	22																	×	×	×	×	×	*															
	design	p.43	pages and layout	23																						×	*															
			site flow	24																							×	*															
			use cases	25																									×	*	×	×											
			sequence	26																										×	*	×											
			class diagrams	27																										×	×	*											
			database design	28																												×	*										
	integration	p.50	existing data	29																													×	*	×	×							
			existing transaction services	30																														×	*	×							
			data conversion requirements	31																														×	×	*							
Technical Architecture	user information management	p.339		32		×		×																											×	*							
	security implementation	p.399		33												×	×	×	×																		×	*						
	quality implementation	p.375		34																																		×	*					
	deployment	p.421		35																																			×	*				
	application framework	p.18		36			×																																	*	×	×	×	
	development tools	p.53		37																																					×	*	×	×	
	language	p.53		38																																					×	×	*	×	
	database	p.54		39																																					×	×	×	*	

DSM (Design Structure Model)

- 別ファイルにある構造化されたExcelシートによって、物理レベルのe-Commerceプロセスの DSM (Design Structure Matrix)を説明します。

可変体と不変体

- 情報社会は、情報利用者と情報提供者が共生する複雑系
- 前者は、多様化、発散、正エントロピーを求める。
- 後者は、集約化、秩序、負エントロピーを求める。

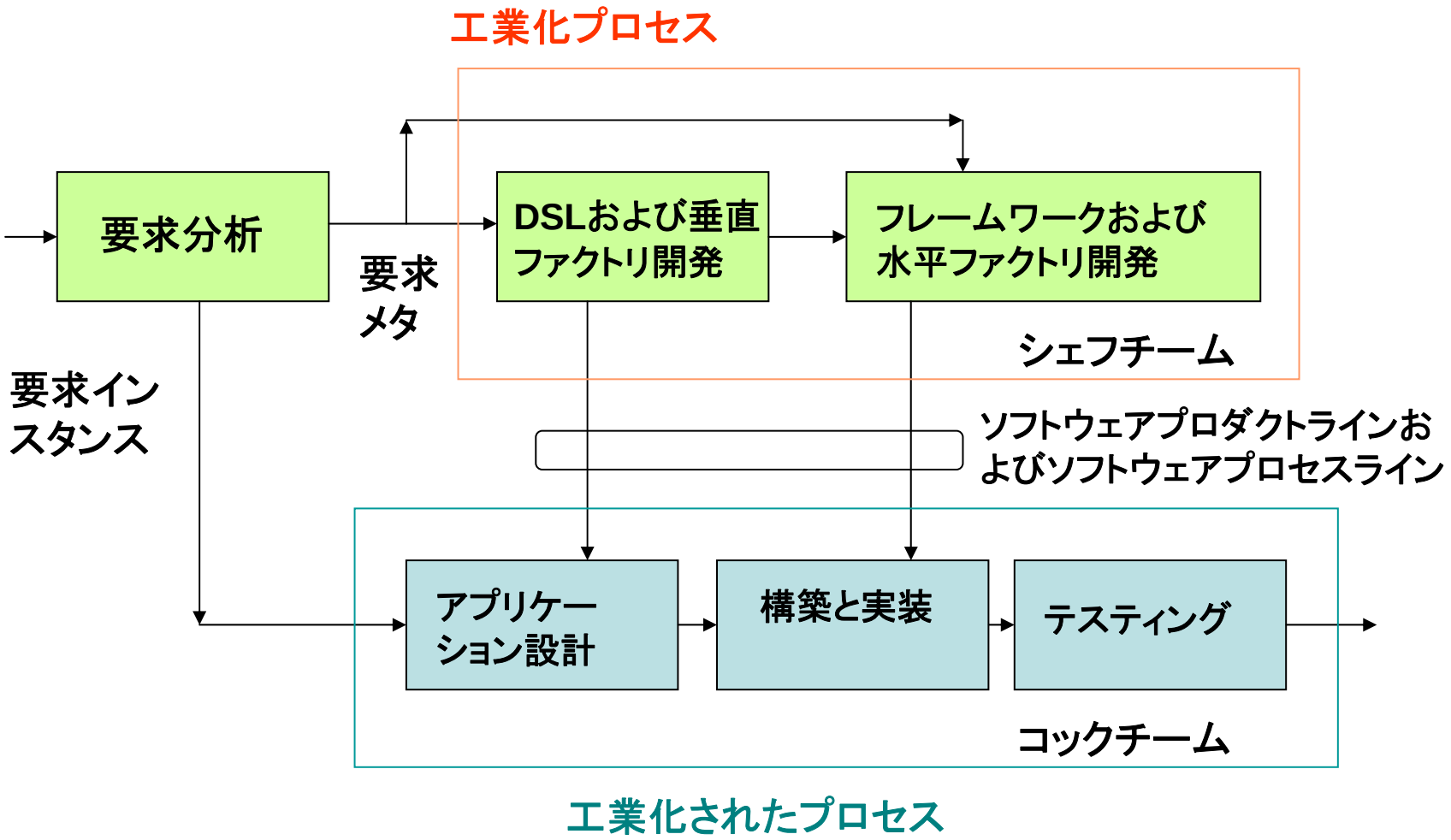
- コンピューティングおよび通信システム (computing and communication systems)は、情報社会複雑系の自己I組織化を支援するものでなければならない。

- コンピューティングおよび通信システムの開発・実装において、適用ドメインの範囲を適正に絞り込むと、可変体と不変体が識別できる。
  - 可変体 (variants): HCI (human-computer interaction) たとえばユーザ・エクスペリエンス、プログラミング言語、オペレーティングシステム、ハードウェア
  - 不変体 (invariants): ドメインに特化したメタ概念体系、モデル代数、仕様代数

工業化とは

- 適用ドメインとその範囲を特定する。
- プロダクトに関して
  - 可変体と不変体を識別する。
  - 不変体をテンプレート化する。
  - 可変体をマークアップして、テンプレートのなかで表現する。
  - プロダクトラインを編成する。
- プロセスに関して
  - 可変体と不変体を識別する。
  - 不変体をテンプレート化する。
  - 可変体をマークアップして、テンプレートのなかで表現する。
  - プロセスラインを編成する。
- 特定された適用ドメインに属する注文を受ける。
  - プロセスラインを利用して、プロセスを計画する。
  - プロダクトラインを利用して、プロダクトを形成する。

ソフトウェアプロセスの工業化例 （特定されたドメインに対して）



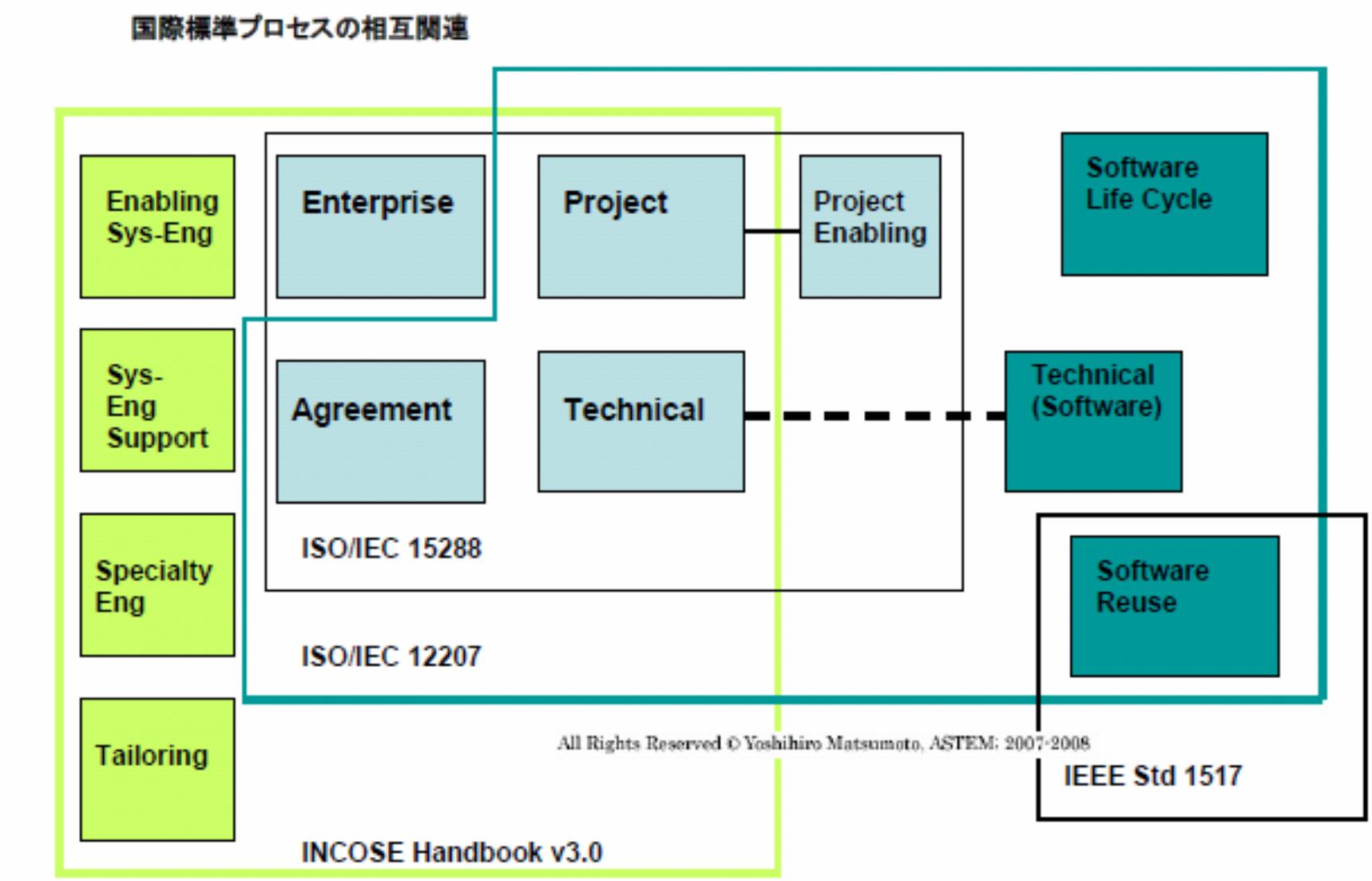
ソフトウェア工業化プロセス例

ソフトウェア工業化プロセス					
プロセスモード	プロセス	アクティビティ			
イノベーションモード (シェフモード)	新規顧客要求分析				
	プロダクトライン定義	問題およびビジネスプロセスの分析から、メタ要求を抽出し、解および実現プラットフォームの新規開発が必要か、再利用可能かを判定。プロダクトラインを新設、または既成プロダクトラインを増補。			
	問題ドメインの範囲定義、解ドメインの範囲定義、ビジネスケースの分析	問題のメタフィーチャモデル作成、ビジネスプロセスメタモデル作成、解メタフィーチャモデル作成、ビジネスプロセスおよびビジネスエンティティメタモデル(ビジネスフローと管理情報との関係)の作成			
	DSLを支援する各種サブシステムの新規開発または増補、フレームワークおよびプラットフォームの新規開発、または増補	アプリケーション・アーキテクチャパターンの新設または増補、利用可能なサービスの取得または]改訂、ユーザプロセス記述のためのドメイン特化言語 (DSL) およびフレームワークの新設または増補、			
	コモディティモードに対する支援				
	プロダクトライン保守・改善・構成管理				
コモディティモード (コックモード)	新規顧客要求分析				
	既成プロダクト系列資産を利用したプロダクト構築				
	プロダクト実現と実装				

ソフトウェア開発工業化プロセスを国際化（たとえば、オフショア化）するためには

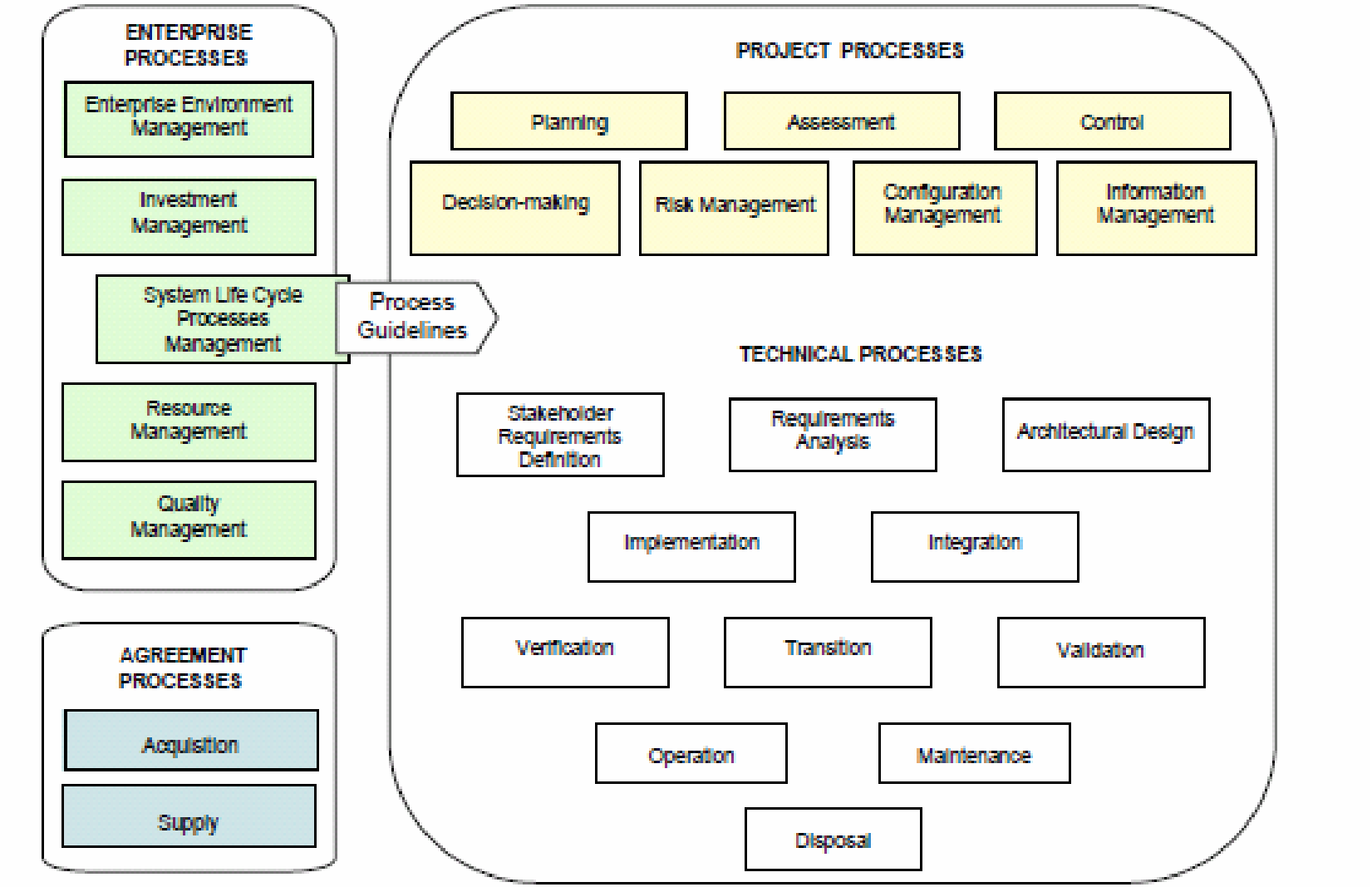
- 日本が70年代から行ってきたソフトウェア開発工業化プロセスを国内で標準化して、ISO国際標準に提案できるようにすることが必要である。
- ISO/IEC FCD1 12207のなかの、Article7.3 Software Reuseをベースにして、’Software Industrialization’というArticleを追加提案する作業を行ってはどうか。

ISO/IEC 15288 vs. ISO/IEC 12207 vs. IEEE Std. 1517 vs. INCOSE Handbook v3.0



INCOSE Systems Engineering Handbook v.3.1・ライフサイクルプロセス

INCOSE-TP-2003-002-03.1
INCOSE SYSTEMS ENGINEERING HANDBOOK, version 3.1
August 2007



Systems Engineering Process Activities (supporting activities)

INCOSE-TP-2003-002-03.1
INCOSE SYSTEMS ENGINEERING HANDBOOK, version 3.1
August 2007

Table 1-1 Systems Engineering Process Activities Overview

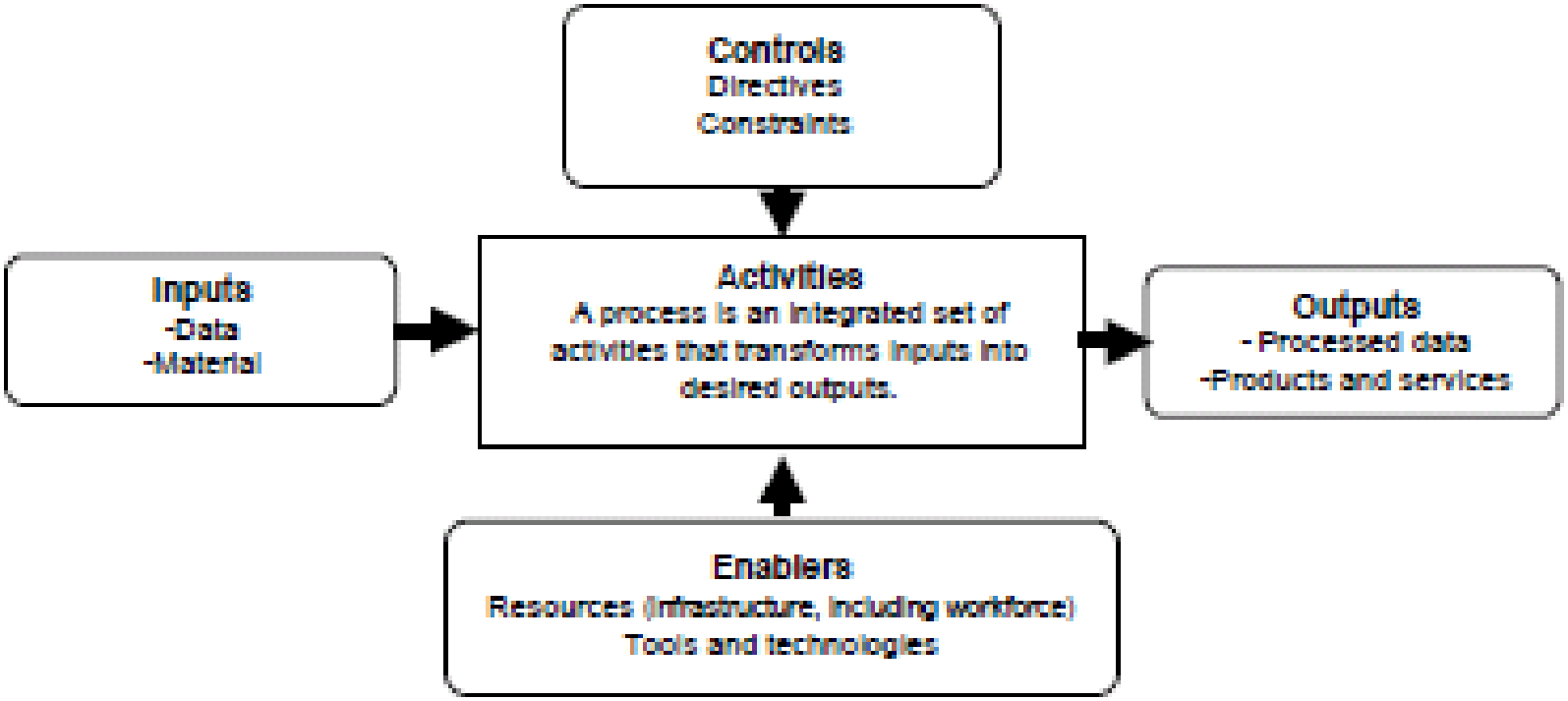
	Systems Engineering Process Activity	Focus	When it is most Useful
7.0	*Enabling System Engineering*	-	-
7.1	Decision Management	Trade studies and project reviews	Through Life
7.2	Requirements Management	System requirements	Through Life
7.3	Risk and Opportunity Management	Recognizing opportunities and risks	Through Life
8.0	*Systems Engineering Support*	-	-
8.1	Acquisition and Supply	Procurement business relationships	Through Life
8.2	Architectural Design	Technical analysis	Development Stage
8.3	Configuration Management	Control of changes through life	Through Life
8.4	Information Management	Project archives and info exchange	Through Life
8.5	Investment Management	Estimation and analysis of costs	Through Life
8.6	Project Planning	Managing technical activities	Through Life
8.7	Quality Management	Product and process assessment	Through Life
8.8	Resource Management	Skills and resource availability	Development Stage
8.9	Validation	User concurrence – correct system	Through Life
8.10	Verification	Requirements met – system correct	Through Life
9.0	*Specialty Engineering Activities*	-	-
9.1	Design for Acquisition Logistics	Integrated logistics support solutions	Development Stage
9.2	Electromagnetic Compatibility	Electro-magnetic protections	Development Stage
9.3	Environmental Impacts	Care for the biosphere and humans	Development Stage
9.4	Human Factors	Human capabilities and well-being	Development Stage
9.5	Mass Properties	Physical characteristics of the system	Development Stage
9.6	Modeling, Simulation, & Prototype	Early validation and testing	Development Stage
9.7	Safety/Health Hazards	Minimum risk to users	Through Life
9.8	Sustainment Engineering	Continued use of system	Through Life
9.9	Training Need Analysis	Basis for training requirements	Development Stage

Systems Life Cycle Process N-squared chart

INCOSE-TP-2003-002-03.1
INCOSE SYSTEMS ENGINEERING HANDBOOK, version 3.1
August 2007

n-squared chart illustrating input-output dependencies between the System Life Cycle Processes																						
1	x	x	x		x		x	x	x	x				x	x	x	x					x
	2	x					x							x	x	x	x					x
		3	x	x	x		x							x	x	x	x					x
			4	x			x							x	x	x	x					x
				5	x		x							x	x	x	x					x
					6	x	x	x						x	x	x	x					x
						7	x	x						x	x	x	x					x
							8	x	x	x				x	x	x	x					x
								9	x	x				x	x	x	x					x
									10	x				x	x	x	x					x
										11				x	x	x	x					x
x	x	x		x	x	x	x				12			x	x	x	x	x	x	x	x	x
												13		x	x	x	x	x	x	x	x	x
													14	x	x	x	x	x	x	x	x	x
														15	x	x	x	x	x	x	x	x
														x	16	x	x	x	x	x	x	x
														x	x	17	x					x
														x	x	x	18					x
x											x	x	x	x	x	x	x	19	x			x
x											x	x	x	x	x	x	x	x	20			x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			21		x
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	22	x
											x	x	x	x	x	x	x	x				23

プロセス・コンテキスト図テンプレート



INCOSE-TP-2003-002-03.1
INCOSE SYSTEMS ENGINEERING HANDBOOK, version 3.1
August 2007

標準プロセスとソフトウェアファクトリ組織 (1)

	15288 Process Group	15288 Process	Corresponding INCOSE Process: Activities	Corresponding 12207 Process: Activities	ソフトウェアファクトリ組織
	Agreement	Acquisition	Acquisition	Acquisition	外注／購買部
		Supply	Supply	Supply	システム営業部
	Enterprise	Enterprise Environment Management	Enterprise Environment Management	Project-Enabling Processes: 1. Life Cycle Model Management; 2. Infrastructure Management; 3. Project Portfolio Management; 4. Human Resources Management; 5. Quality management	システム技術部
		Investment Management	Investment Management		システム事業部スタッフ
		System Life Cycle Processes Management	System Life Cycle Management		システム技術部
		Resource Management	Resource Managemnt		システムエンジアリング教育・訓練センター
			Quality Management		システム技術部
	Project	Project Planning	Project Planning	Project Planning	製品部長、製品課長、プロジェクトマネージャ
		Project Assessment	Project Assessment	Project Assessment and Control	
		Project Control	Project Control		
		Decision-Making	Decision Making	Decision Management	
		Risk Management	Risk and Opportunity Management	Risk Management	
		Configuration Management	Configuration Management	Configuration Management	
		Information Management	Information Management	Information Management	
				Measurement	
			Enabling Systems Engineering Process Activities: 1. Decision Management; 2. Requirements Management; Risk and Opportunity Management;		
			Systems Enginnering Support Activities: 1. Acquisition and Supply; 2. Architecture Desisgn; 3. Configuration Management; 4. Information Management; 5. Investment Management; 6. Project Planning; 7. Quality Management; 8. Resource Management; 9. Validation; 10. Verification		
			Speciality Engineering Activities: 1. Design for Acquisition Logistics; Electromagnetic Capability Analysis; 3. Environmental Impact Analysis; 4. Human Factors; 5. Mass Properties Engineering Analysis; Modeling, Simulation, and Prototyping; 7. Safety & Health Hazard Analysis; 8. Sustainment Engineering Analysis; Training Needs Analysis		

標準プロセスとソフトウェアファクトリ組織 (2)

	15288 Process Group	15288 Process	Corresponding INCOSE Process: Activities	Corresponding 12207 Process: Activities	ソフトウェアファクトリ組織
	Technical	Stakeholder Requirements Definition	Stakeholder Requirements Definition	Stakeholder Requirements Definition	システム技術部
		Requirements Analysis	Requirements Analysis	System Requirements Analysis	
		Architectural Design	Architectural Design	System Arcitectural Design	
		Implementation	Implementation	Implementation	
		Integration	Integration	System Integration	
		Verification	Verification	System Qualification Testing	
		Transition	Transition	Software Installation	システム建設・サービス部
		Validation	Validation	Software Acceptance Support	
		Operation	Operation	Software Operation	
		Maintenance	Maintenance	Software maintenance	
		Disposal	Disposal	Software Disposal	

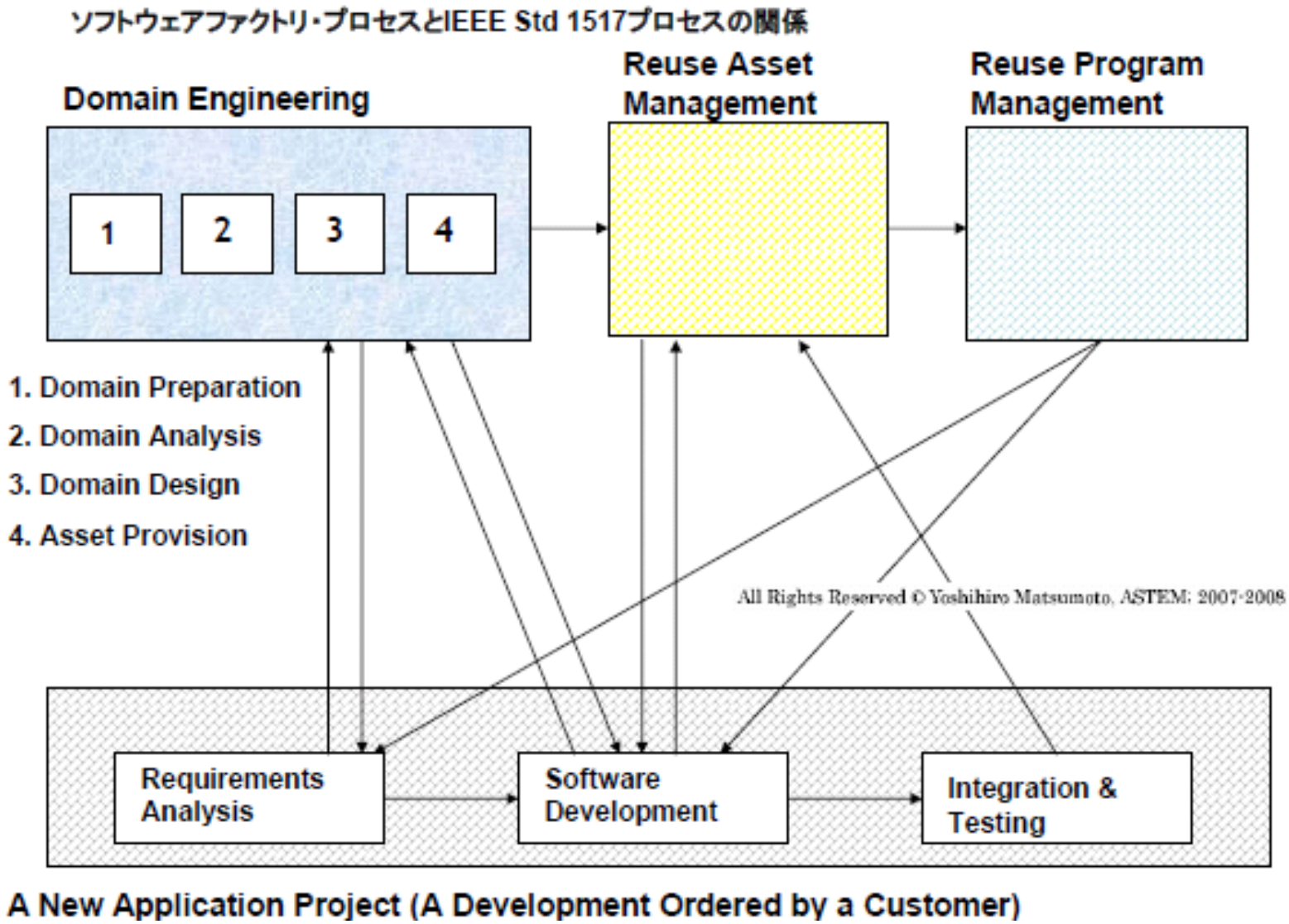
標準プロセスとソフトウェアファクトリ組織 (3)

	12207 process Group	12207 Process	IEEE Std 1517	ソフトウェアファクトリ組織
	SW Implementation	Software Implementation		ソフトウェアプロジェクト
		Software Requirements Analysis		
		Softwre Architectural Design		
		Software Detailed Design		
		Software Construction		
		Software Integration		
		Software Quaslification Testing		
	SW Support	Software Documentation Management		技術管理部
		Software Configuration Management		
		Software Quality Assurance		試験・検査部および品質管理・保証部
		Software Verification		
		Software Validation		
		Software Review		
		Software Audit		
		Software Problem Resolution		
	Software Reuse	Domain Engineering	Integration of Reuse, Reuse Support Process, Reuse Organizational Support Process (for more detail, see slide ″Content Summaru of IEEE Std 1517″)	再利用資産開発・管理部
		Reuse Asset Management		
		Reuse Program Management		

IEEE Std 1517-1999の
プロセス項目

	Integration of reuse	Acquisition process	Initiation
			RFP preparation
			Contract preparation and update
			Supplier monitoring
			Acceptance and completion
		Supply process	Prepare a proposal to respond to an RFP from an acquirer
			Prepare a contract to provide a system, software product, or asset to an acquirer
			Determaine the procedures and the resources needed to manage a project to develop and deliver a system, software product or asset to an acquirer
		Development process	Process implementation
			System requirements analysis
			Systgem architectural design
			Software requirements analysis
			Software architectural design
			Software detailed design
			Software coding and testing
			Software integration
			Software qualification testing
			System integration
			System qualification testing
			Software installation
			Software acceptance support
		Operation process	Operation of the system
			Providing operation support to the users of the system
		Maintenance process	Modify an exisiting software product
			Migrate an exisiting software product
			Retire an existing software product
	Reuse support process	Asset management process	Process implementation
			Asset storage and retrieval process
			Asset management and control
	Reuse organizational life cycle process	Reuse program administration process	Initiation
			Domain identification
			Reuse assessment
			Planning
			Execution and control
			Review and evaluation
	Reuse cross-project life cycle process	Domain engineering process	Process implementation
			Domain analysis
			Domain design
			Asset provision
			Asset maintenance

ソフトウェアプロセスとソフトウェア再利用プロセス



ISO FCD1-12207-Article7.3 Software Reuse

ISO FCD1-12207 Article7.3	
Domain Engineering Processes	
	Process Implementation
	Domain Analysis
	Domain Design
	Asset Provision
	Asset Maintenance
Reuse Asset Management	
	Process Implementation
	Asset Storage and Retrieval Definition
	Asset Management and Control
Reuse Program Management	
	Initiation
	Domain Identification
	Reuse Assessment
	Planning
	Execution and Control
	Review and Evaluation

ドメインエンジニアリング・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.1 Domain Engineering Process (1)

- *目的*
  - ドメインモデルの開発・保守
  - ドメインアーキテクチャの開発・保守
  - 資産 (assets) の開発・保守

- *アクティビティおよびタスク*
  - プロセスの準備
    - ドメイン技術者によるドメインエンジアリング計画の作成と実施
    - ドメイン技術者によるドメインモデル、ドメインアーキテクチャを表現する形式の選択
    - ドメイン技術者による、資産に対する問題、または変更に関する要請受領、解決、および資産管理者へのフィードバック手順の決定
  - ドメイン分析
    - ドメイン技術者によるドメインの境界と他のドメインとの関係の定義
    - ドメイン技術者によるこのドメインにおけるソフトウェアプロダクト開発者が現在もつ、または将来もつであろうニーズの把握
    - ドメイン技術者によるドメインモデルの作成

- *成果（アウトカム）*
  - ドメインモデル、ドメインアーキテクチャ の表現形式
  - ドメインの境界と他のドメインとの関係
  - ドメインモデル、すなわち本質的な共通した、および異なった特徴 (features)、能力 (capabilities)、概念 (concepts)、および機能 (functions)
  - ドメインに帰属するシステム群 (family of systems)を、それらの共通性および多様性とともに記述したドメインアーキテクチャ

ドメイン・エンジニアリング・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.1 Domain Engineering Process (2)

- *アクティビティおよびタスク(つづき)*
  - ドメイン分析(つづき)
    - ドメイン技術者による語彙(vocabulary)構築
    - ドメイン技術者によるドメインモデルの分類と文書化
    - ドメイン技術者による組織で決めた資産受理 (acceptance)および認定(certification)手順に従ったドメインモデルおよびドメイン語彙の評価
    - ドメイン技術者が主宰するドメイン分析のレビュー
    - ドメイン技術者によるドメインモデルの資産管理者への引渡し
  - ドメイン設計
    - ドメイン技術者によるドメインモデルと整合し、組織で決めた標準に基づいたドメインアーキテクチャの作成と文書化
    - アーキテクチャ設計技法および組織で決めた資産受理および認定手順に従ったドメインアーキテクチャの評価
    - ドメイン技術者による資産仕様の開発と文書化

- *成果(つづき)*
  - ドメインに帰属する資産
    - ドメインに帰属する資産は、調達、または開発され、それらのライフサイクルを通して保守される。
    - ドメインモデルおよびドメインアーキテクチャは、それらのライフサイクルを通して保守される。

ドメイン・エンジニアリング・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.1 Domain Engineering Process (2)

- *アクティビティおよびタスク（つづき）*
  - ドメイン設計（つづき）
    - 資産それぞれに関して、組織としての資産受理および認定手順に従った資産仕様の評価
    - ドメイン技術者が主宰するドメイン設計のレビュー
    - ドメイン技術者によるドメインアーキテクチャの資産管理者への引渡し
  - 資産提供
    - ドメイン技術者による、調達、または開発による資産の取得
    - 資産の文書化と分類
    - ドメイン技術者による、組織としての資産受理および認定手順に従った資産の評価
    - ドメイン技術者が主宰する資産のレビュー
    - ドメイン技術者による資産の資産管理者への引渡し

ドメイン・エンジニアリング・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.1 Domain Engineering Process (3)

- *アクティビティおよびタスク(つづき)*
  - 資産保守
    - ドメインモデルおよびドメインアーキテクチャに対する適合性、資産を利用しているシステムまたはソフトウェアプロダクトに対する影響、将来想定される資産利用者に対する影響、資産の再利用性を考慮した上での、ドメイン技術者による、資産に関する改変依頼および選択肢変更要請の分析

再利用資産マネジメント・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.2 Reuse Asset Management Process (1)

- *目的*
  - 廃棄計画を勘案した再利用資産の生涯管理

- *アクティビティおよびタスク*
  - プロセスの実現
    - 資産管理者による資産管理に関わる資源と手順を定義する資産管理計画作成
    - 資産管理者による資産管理計画の実施
    - 資産管理者が主宰する資産管理計画のレビュー
  - 資産収蔵および検索の定義
    - 資産管理者による資産収蔵および検索機構の実現と保守
    - 資産管理者による資産分類に使用する分類計画の開発、文書化および保守
    - 資産管理者が主宰する資産収蔵および検索機構のレビュー
  - 資産管理およびコントロール
    - 資産管理者に提出された資産に対する、資産受理および認定判定基準に従った評価
    - 受理された資産の資産収蔵および検索機構による再利用可能化

- *成果（アウトカム）*
  - 資産管理戦略
  - 資産分類構想
  - 資産受理、認定、および廃棄判定基準
  - 資産収蔵および検索機構
  - 資産利用記録
  - 資産改変コントロール
  - 収蔵および検索機構にある資産に関する問題検出、修正および改変実施を利用者に通知する仕組み

再利用資産マネジメント・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.2 Reuse Asset Management Process (2)

- *アクティビティおよびタスク（つづき）*
  - 資産管理およびコントロール（つづき）
    - 再利用分類判定基準に従った資産の分類
    - 資産管理者による構成管理の実施
    - 資産管理者による資産が再利用された経路の追跡、および資産再利用実績のドメイン技術者に対する報告
    - 資産管理者からドメイン技術者に対する、資産再利用者から提出された資産改変要請および問題報告の取次ぎ
    - 資産管理者による上記改変要請および問題報告、およびそれに対する対処行動の監視と記録
    - 資産管理者による、資産再利用者およびドメイン技術者に対する、資産のなかに存在する問題発見通知、資産に対する改変発生通知、資産収蔵および検索機構からの資産の廃棄通知

再利用プログラム・マネジメント・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.3 Reuse Program Management Process (1)

- *目的*
  - 組織としての再利用プログラムを計画、確立、管理、コントロールおよび監視し、体系的に再利用機会を引き出すこと
- *アクティビティおよびタスク*
  - プロセスの始動
    - 再利用到達点 (goals)、目的 (purposes)、目標 (objectives)および範囲 (scope)の確立による、組織としての再利用プログラムの始動
    - 再利用スポンサの任命
    - 参加者の識別と役割の申し渡し
    - 組織としての権威と責任を明示するために必要な運営機能 (steering function)の確立
    - 支援機能の確立
  - ドメインの識別
    - 再利用プログラム実行管理責任者 (administrator)による、ドメインの識別と文書化
    - 再利用プログラム実行管理責任者による、ドメインの評価
    - 再利用プログラム実行管理責任者が主宰する、ドメインのレビュー
    - 再利用プログラム実行管理責任者による将来のために必要なドメインの詳細化および範囲の見直し
- *成果（アウトカム）*
  - 目的、範囲、および目標が明示された組織としての再利用戦略定義
  - 強力な再利用機会をもつドメインの同定
  - 組織として体系化された再利用能力の査定 (assessment)
  - ドメインそれぞれに潜在する再利用性の査定

再利用プログラム・マネジメント・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.3 Reuse Program Management Process (2)

- *アクティビティおよびタスク（つづき）*
  - 再利用査定　(assessment)
    - 再利用プログラム実行管理責任者による、組織の体系的再利用能力(capability)の査定
    - 再利用プログラム実行管理責任者による、再利用成功度の査定
    - 再利用プログラム実行管理責任者による、上記査定結果を元にした組織に対する再利用プログラム戦略およびその実現計画の詳細化勧告
    - 再利用プログラム実行管理責任者による、再利用基盤（スキル、技法、再利用プロセス、組織構造、および計量法）の漸進的改善
  - 計画作成
    - 資源と手順を定義した再利用プログラム実現計画の作成、文書化および保守
    - 計画のレビュー
    - 計画の再利用運営機能および関連管理者による承認および支持とりつけ
    - 再利用プログラム実行管理責任者によるレビュー

- *成果（アウトカム）（つづき）*
  - 提起されているアプリケーションに対して、対象物が再利用が適しているかどうかの評価
  - 組織としての再利用戦略
  - 関係者の間におけるフィードバック、コミュニケーション、通知機構の確立
  - 再利用プログラムの監視および評価

再利用プログラム・マネジメント・プロセス
ISO/IEC FCD 12207
IEEE P12207/CD2/FCD 7.3.3 Reuse Program Management Process (3)

- *アクティビティおよびタスク（つづき）*
  - 実施およびコントロール
    - 再利用プログラム実現計画のなかのアクティビティ実施
    - 再利用プログラム実行管理責任者による進捗監視および調整
    - 発生した問題および非適格の記録と解決
    - 再利用プログラム実行管理責任者による組織的な後方支持、支援および約定の確認
  - レビューと評価
    - 再利用プログラム実行管理責任者による、定期的な達成度、継続的適性度、および有効度の定期的査定
    - 再利用プログラム実行管理責任者による、査定結果および学習内容の運営機能および関連管理者に対する開示
    - 再利用プログラム実行管理責任者による、再利用プログラムの変更、拡大、および改善勧告と実施

第3章　垂直ソフトウェアファクトリと水平ソフトウェアファクトリ

（Dr. Paul Clements (SEI/CMU)および筆者の合議に基づく。）

Terminology (1)

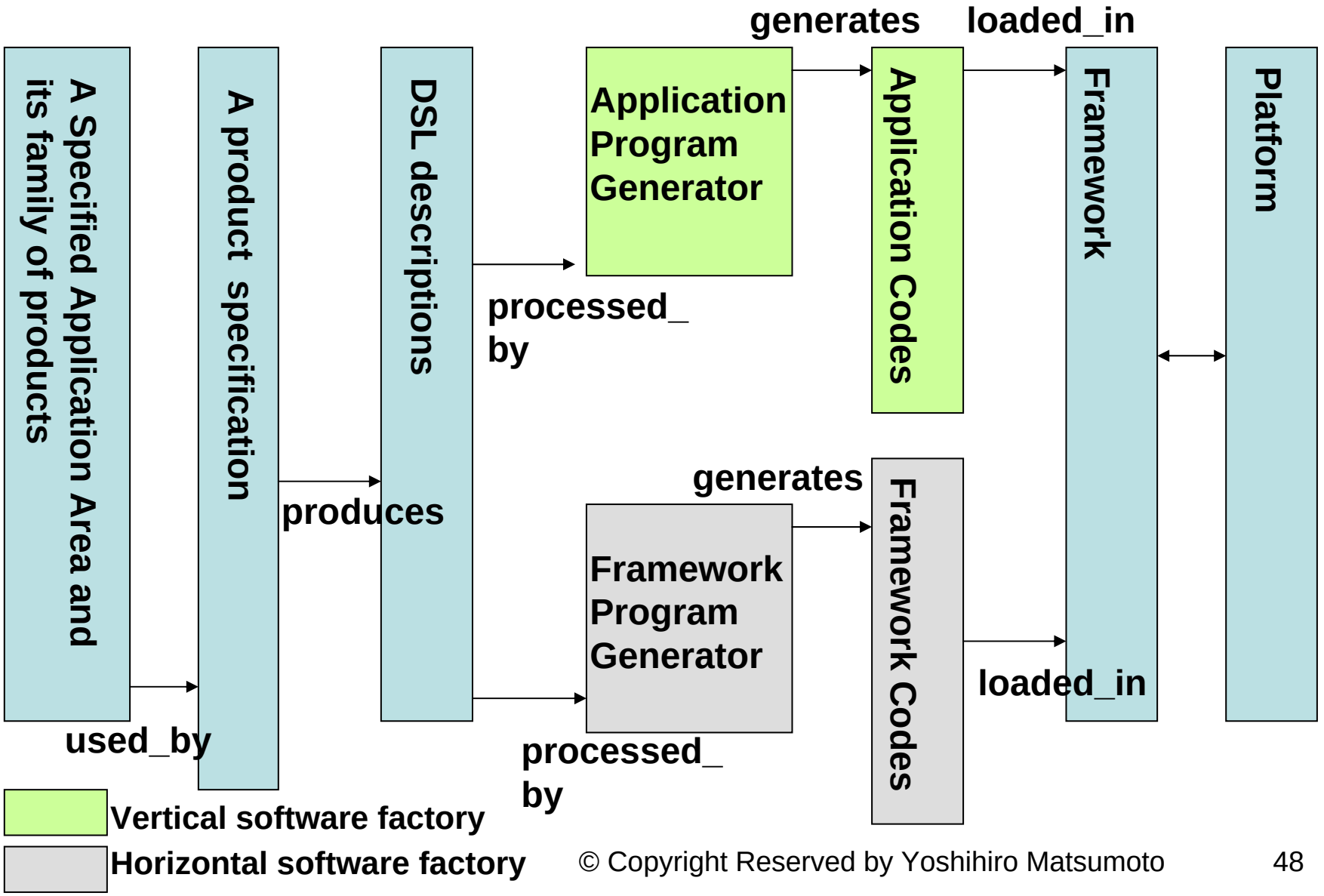
- **Application area**
  - **An application area encapsulates knowledge for building a wide variety of product families**
- **Product family or family of products**
  - **A product family is a group of products that can be built from a common set of assets.**
- **Product line**
  - **A product line is a group of products sharing a common managed set of features that satisfy the specific needs of a selected market.**

Reference: Czarnecki, K. and U.W. Eisenecker, Generative Programming, Addison-Wesley (2000)

Terminology (2)

- **Framework**
  - **A framework provides the interface between the codes generated by the application program generator and the underlying platform. The framework codes are generated by the framework program generator.**
- **Platform**
  - **A platform provides hardware, operating system, language processing systems, communication control system, human-computer interaction management system, middleware, and database management system**

Software Factory Processes for building a new product



Verticalization and Horizontalization

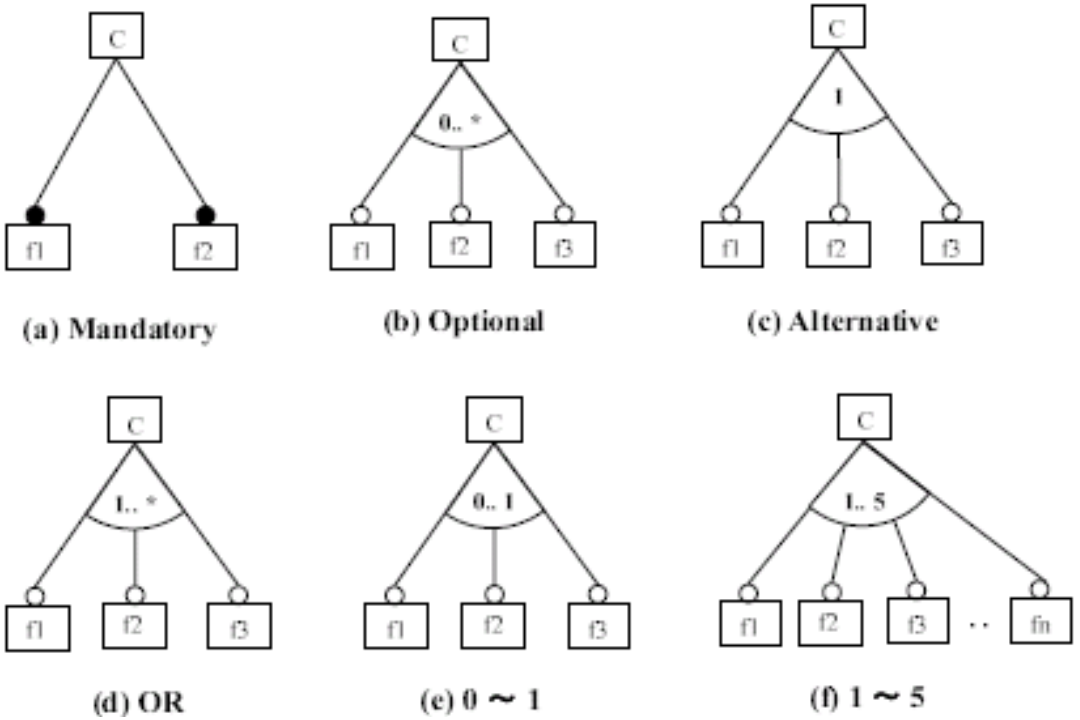
- **“Verticalization” means the ability to build a family of products within an application area [Paul Clements].**
- **“Horizontalization” means the ability to build a production capability for families across a wide variety of application area [Paul Clements].**
- **“Horizontalization” means the ability to build a production capability for families across a wide variety of application area, and the ability to build a production capability for a wide variety of products, each of which must be built on each different platform, within a product family [Yoshihiro Matsumoto].**

第4章　ソフトウェアフレームワークが1層で、かつプラットフォームが固定されている場合のソフトウェアファクトリ（電子ポット・ソフトウェアファクトリを例とした開発の初歩解説）

- ここで説明する電子ポット・ソフトウェアファクトリは、2002年から2004年に
- かけて、旧武蔵工業大学・ソフトウェア工学研究室（松本吉弘教授）で開発
- されたものです。詳細は、下記を参照ください。
- http://blues.se.uec.ac.jp/mi-tech_sftlab/contents/2003/pdf/s0263065.pdf
- http://blues.se.uec.ac.jp/mi-tech_sftlab/contents/2004/pdf/0363011.pdf

フィーチャダイアグラム

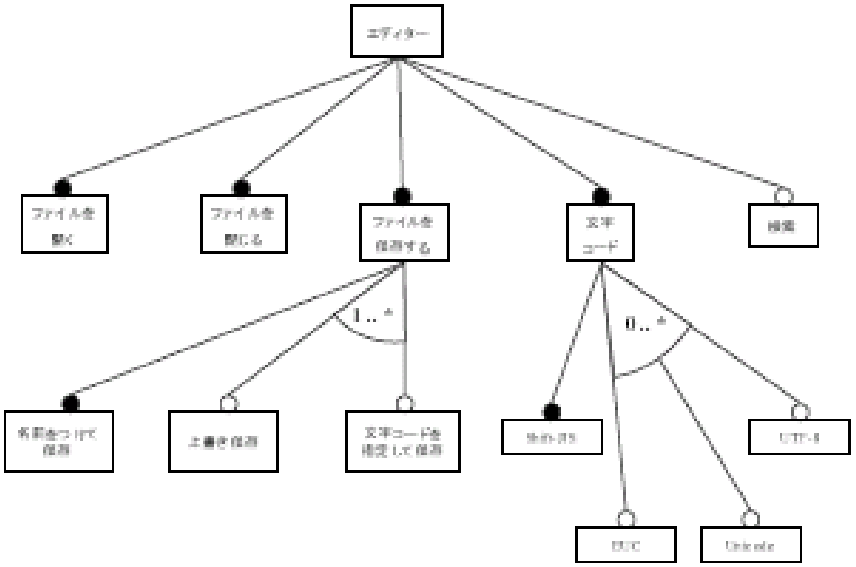
多重度表現	表現される意味
1	正確に 1 個だけを選ばなければならない
0.. 1	多くても 1 個選ばなければならない（0〜1 個選ばなければならない）
0.. n	多くても n 個選ばなければならない
m.. n	少なくても m 個，多くても n 個選ばなければならない
0..*	任意の個数（0 個以上）選ばなければならない
1..*	少なくても 1 個以上選ばなければならない
m..*	少なくても m 個以上選ばなければならない

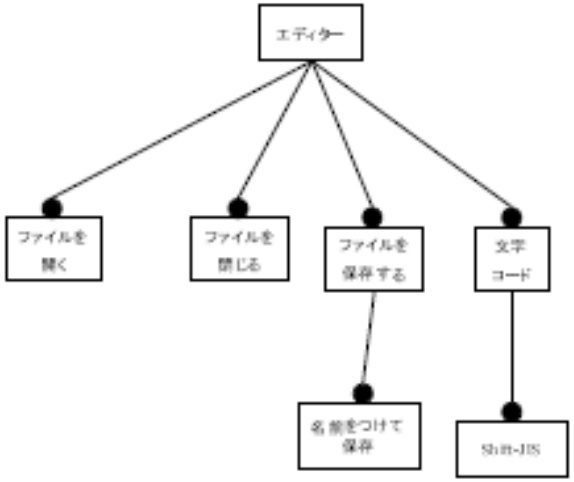


共通フィーチャと可変フィーチャ

フィーチャダイアグラム中の共通フィーチャは以下のように定義される．

- 親が概念（概念ノード）である全ての Mandatory features
- 共通フィーチャの親を持つ全ての Mandatory features





問題フィーチャダイアグラムの作成

1. 対象ドメインのフィーチャとなりえるものを抽出.
2. 得られたフィーチャを基にフィーチャダイアグラムを構築．このステップで，フィーチャの分類も行う.
3. フィーチャの最適化のための分析.

機能的なもの：**SESSAME** の仕様

電源の ON ／ OFF，給湯する，沸騰する，保温する，給湯口をロックする，給湯口のロックを解除する，保温モードを高温（98℃）に変更する，保温モードを節約（90℃）に変更する，保温モードをミルク（60℃）に変更する，蓋を開閉する，タイマを使う，水位を表示する，水量オーバを知らせる，高温エラーを知らせる，温度上がらずエラーを知らせる，アラームを鳴らす

ハードウェア構成：**SESSAME** の仕様

満水センサ，第 1 水位センサ，第 2 水位センサ，第 3 水位センサ，第 4 水位センサ，ヒータ，サーミスタ，給湯ポンプ，給湯口ロック，蓋センサ，タイマ，アラーム，ブザー

その他：**SESSAME** の仕様

PID 制御方式，温度制御テーブル方式，目標温度 ON ／ OFF 方式，ヒステリシス付き目標温度 ON ／ OFF 方式，

さらに，独自に付加した機能については以下のものが抽出された.

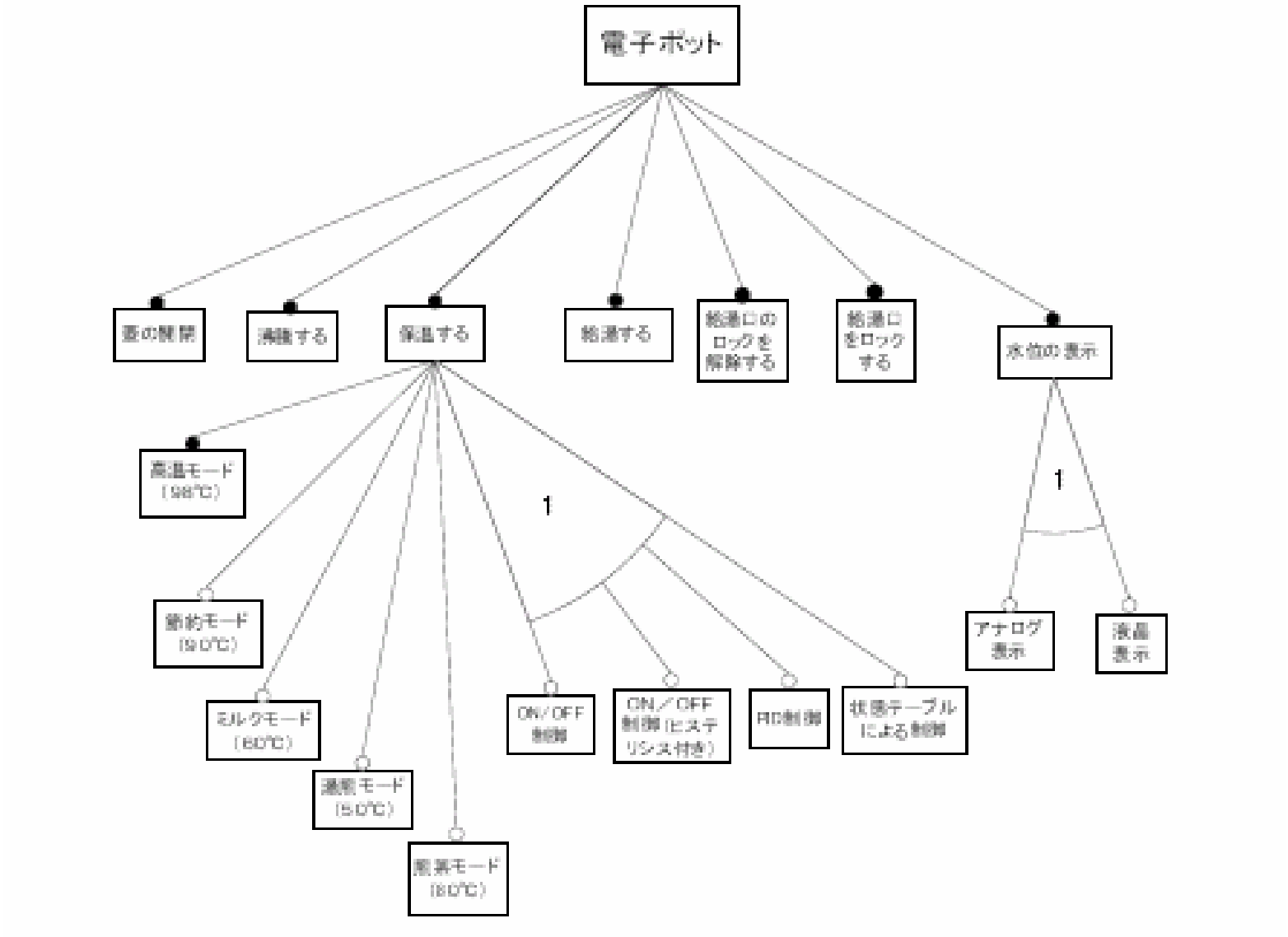
機能的なもの：追加の機能

自動的に給湯口をロックする，保温モードを湯煎（50℃）に変更する，保温モードを煎茶（80℃）に変更する，水量不足を知らせる，アナログメータで水位を表示する

ハードウェア構成：追加の機能

水量不足センサ，アナログ水位メータ

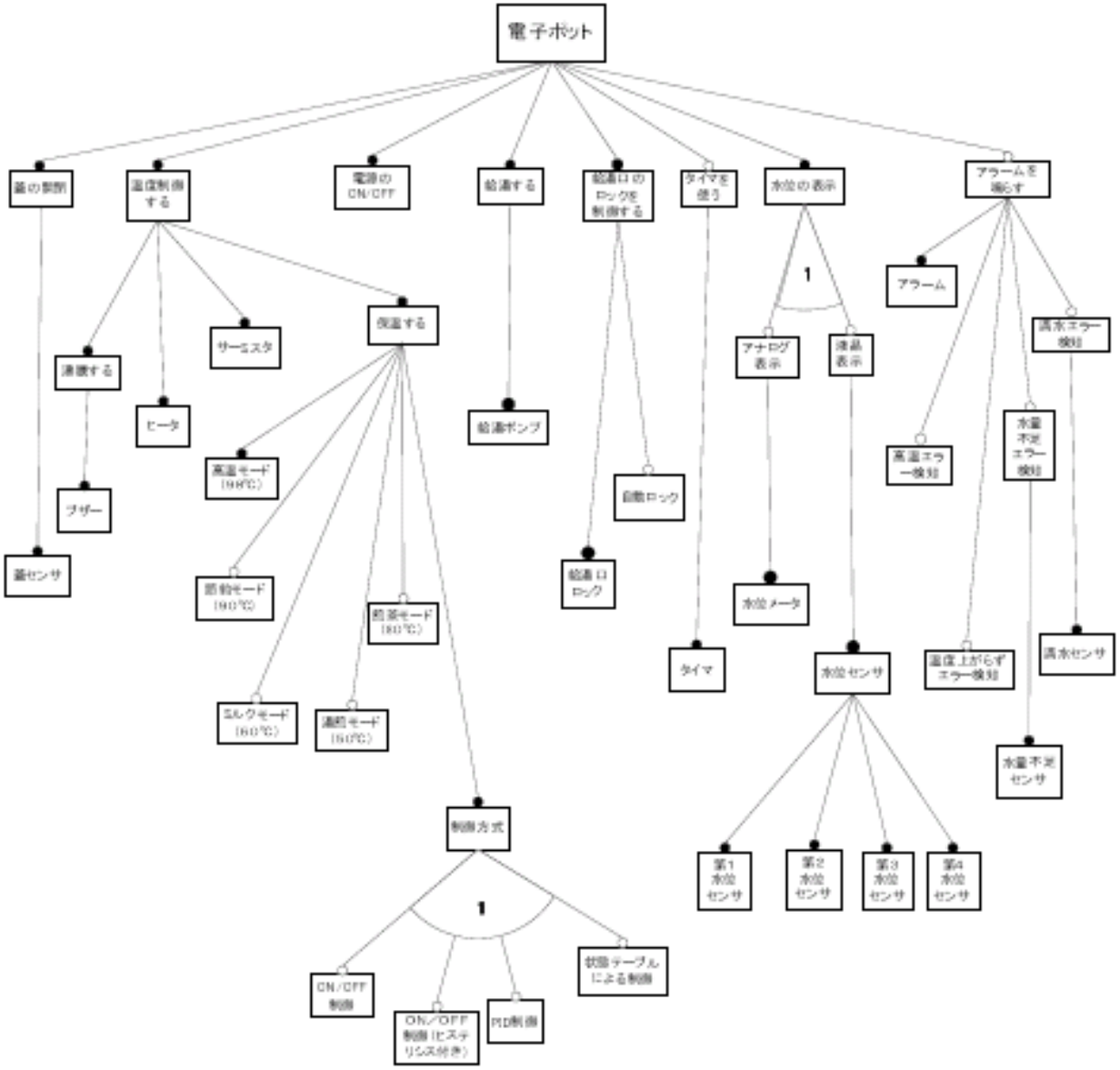
電子ポットの問題フィーチャダイアグラム



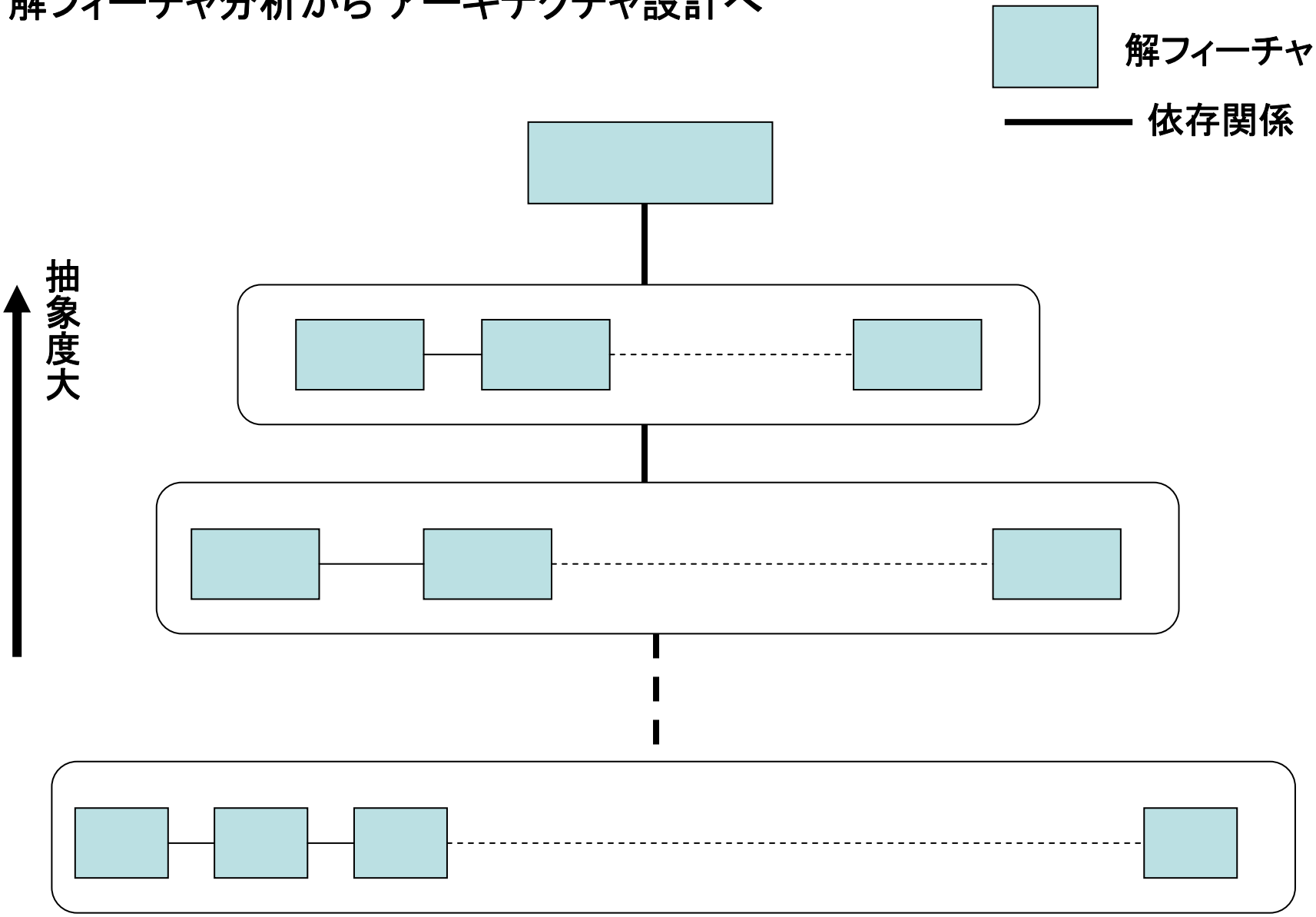
機能的フィーチャからハードウェアフィーチャへのマッピング

1 対 1 対応	機能的なフィーチャ	ハードウェア構成のフィーチャ
○	給湯する	給湯ポンプ
	沸騰する	サーミスタ，ヒータ，ブザー
	保温する	サーミスタ，ヒータ
	給湯口をロックする	給湯口ロック
	給湯口のロックを解除する	給湯口ロック
○	蓋を開閉する	蓋センサ
	タイマを使う	タイマ，ブザー
○	液晶表示	第 1 水位センサ，第 2 水位センサ，第 3 水位センサ，第 4 水位センサ
○	水量オーバを知らせる	満水センサ
	高温エラーを知らせる	サーミスタ
	温度上がらずエラーを知らせる	サーミスタ
○	アラームを鳴らす	アラーム
	自動的に給湯口をロックする	タイマ，給湯口ロック
○	水量不足を知らせる	水量不足センサ
○	アナログ表示	アナログ水位メータ

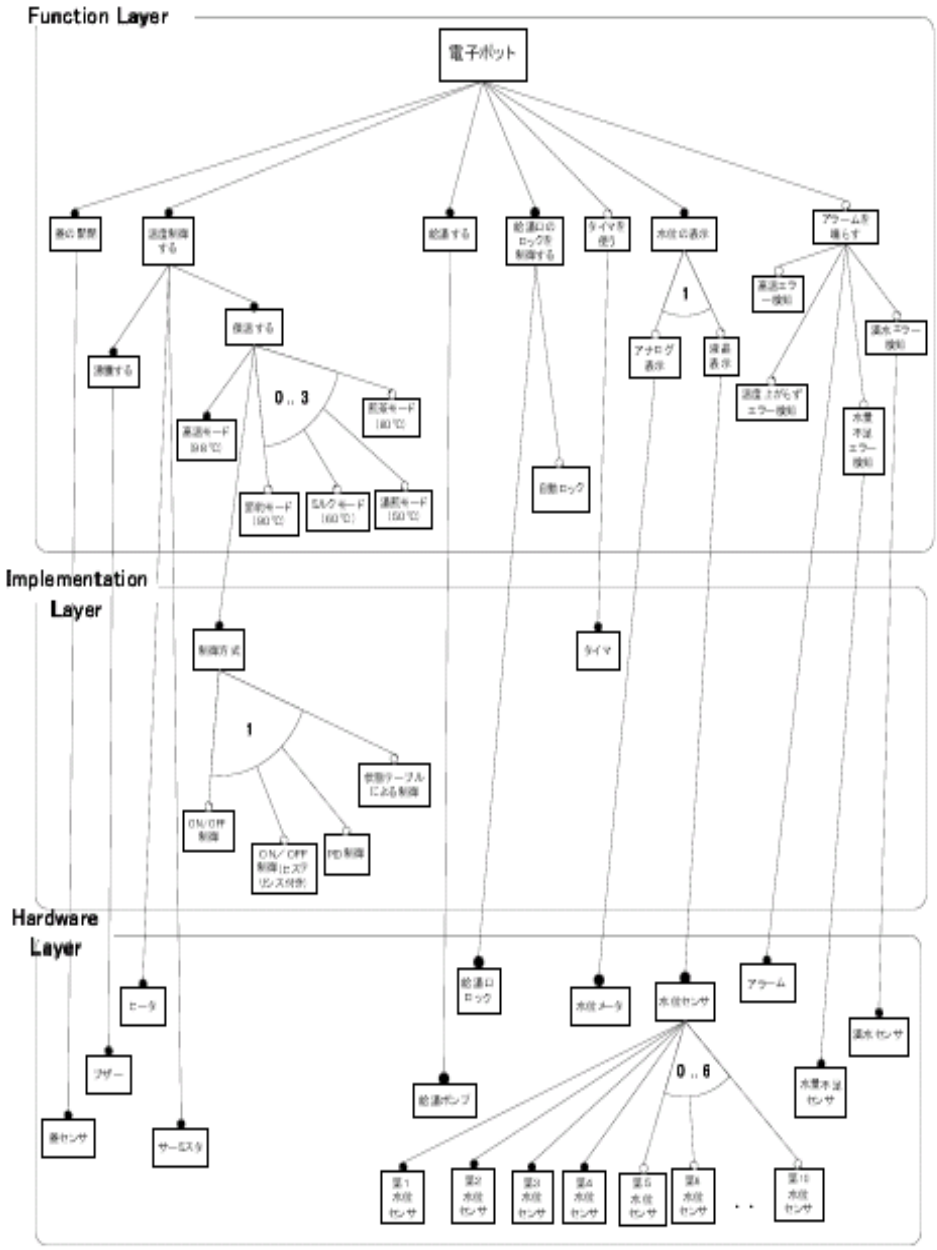
最適化を行った最終的問題フィーチャダイアグラム



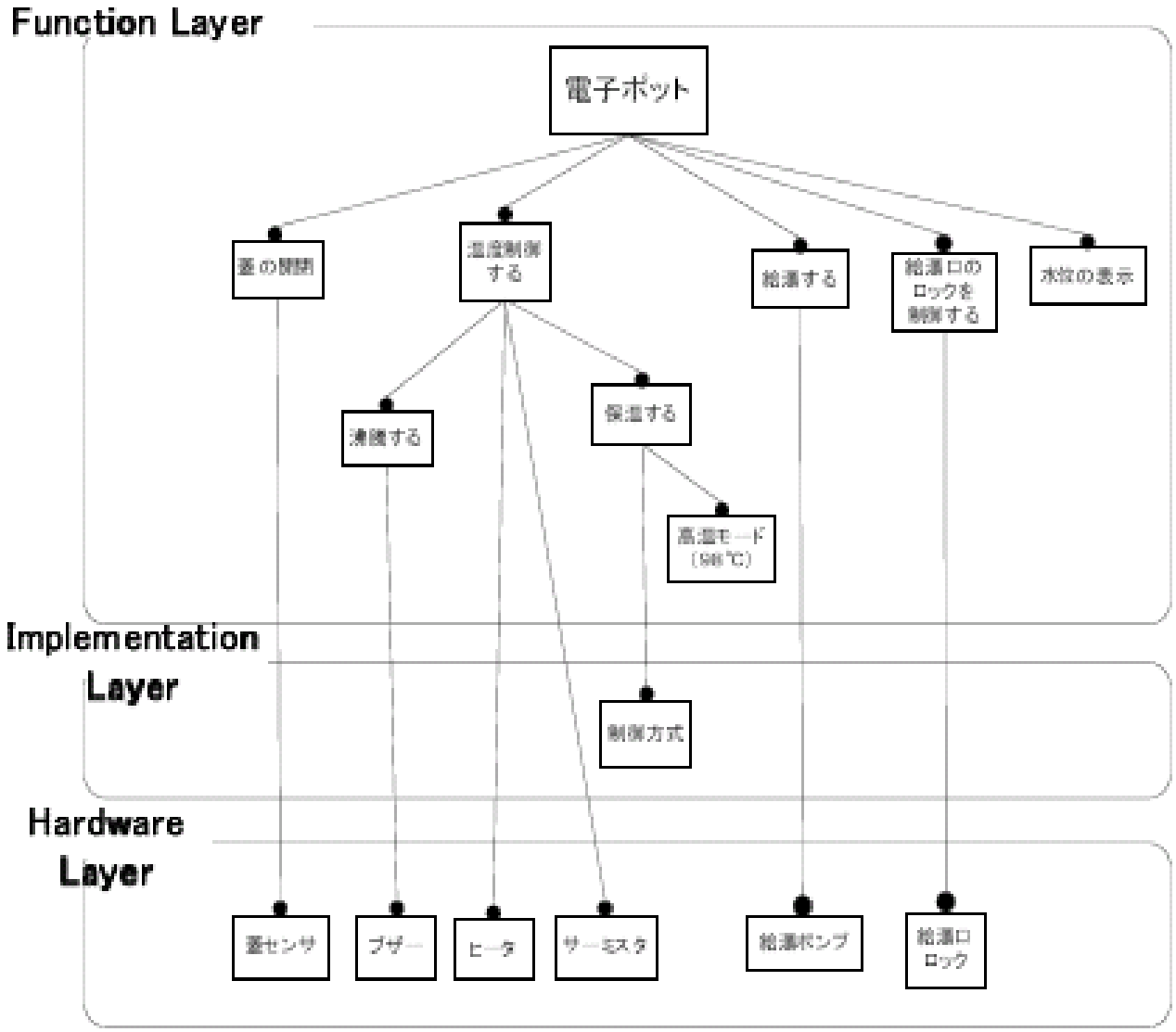
解フィーチャ分析から アーキテクチャ設計へ



解フィーチャダイアグラム



共通解フィーチャの抽出

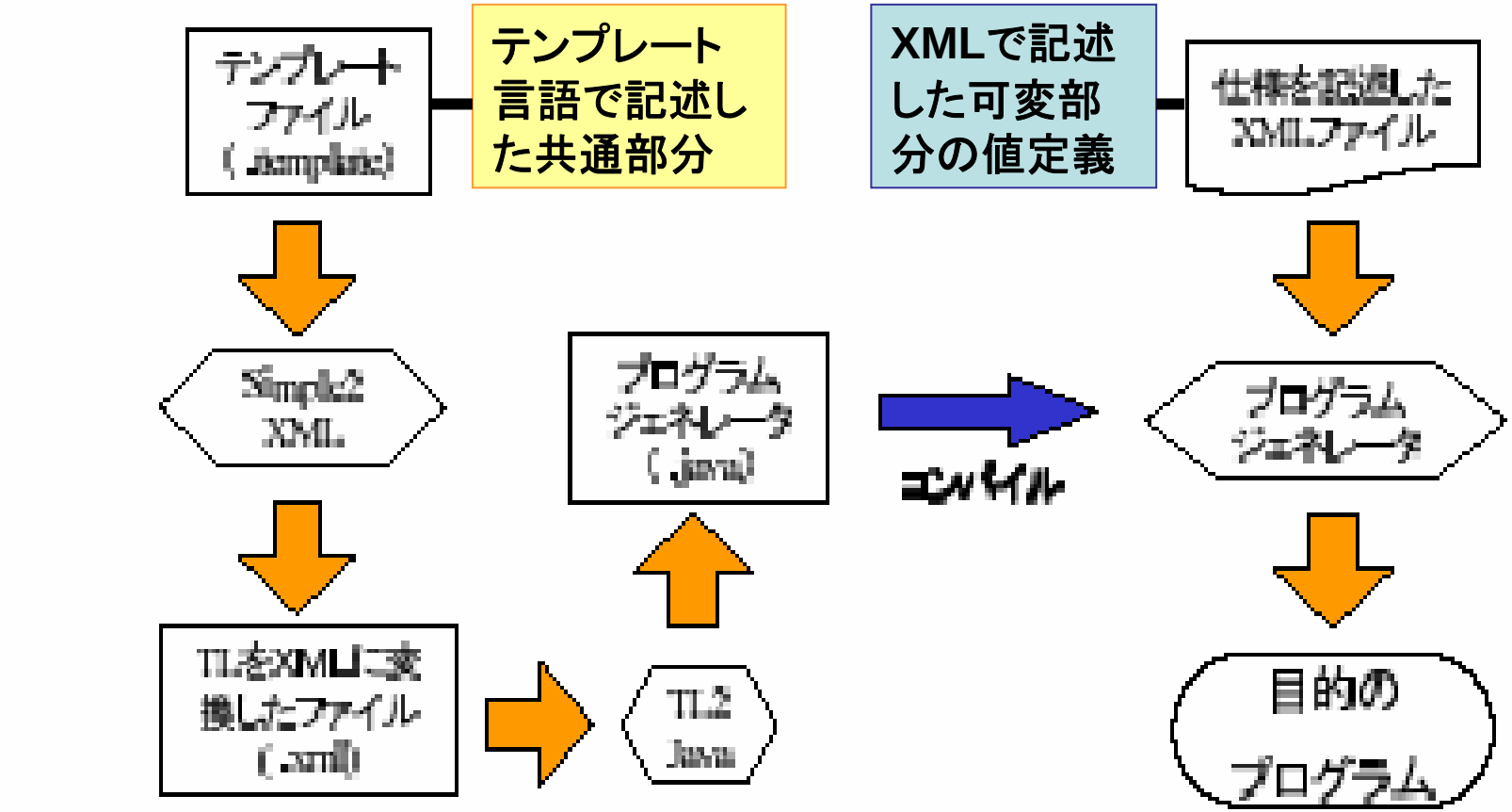


アーキテクチャ設計およびプログラミング手順

- 解コンポーネント構造図からクラス図を作成する。
- テンプレート言語を用いて、共通変数記号、可変変数記号を定義し、テンプレートプログラムを記述する。
- 可変変数記号に値を定義する。
- 両者をまとめた文書を、DSL記述と称する。
- プログラムジェネレータによって、DSL記述から目的プログラムを自動生成する。

ドメイン特化型言語によるプログラム自動生成

ドメイン特化型言語 (Domain-Specific Language)とは、共通部分を記述するテンプレート言語およびそのなかで使われる可変部に対する値定義を記述するための言語をまとめた総称です。



商用プログラムジェネレータの例： CodeSmith Studio

テンプレート言語で記述した共通部分の例

```
1 public class Sample {
2     String[] bookname = {
3     #for "//book"#
4     "#"."#",
5     #end#
6     };
7
8     public static void main(String[] args) {
9         Sample d = new Sample();
10        for(int i=0; i<args.length; i++){
11            for(int j=0; j<d.bookname.length; ++j) {
12                if(args[i].equals(d.bookname[j])){
13                    System.out.println(args[i]+" is in the booklist");
14                    break;
15                }else if(j == d.bookname.length - 1){
16                    System.out.println(args[i]+" is not in the booklist");
17                }
18            }
19        }
20        #if "count(//book)>5"#
21        System.out.println("本棚に本が置け去せん");
22        #fi#
23    }
24}
```

可変部分に対する値定義記述の例

```
<?xml version="1.0" encoding="shift_jis"?>
<booklist>
  <book category="computer">XMLって何?</book>
  <book category="food">ラーメン大好き</book>
  <book category="computer">初歩のUML</book>
  <book category="computer">Java</book>
</booklist>
```

第5章　フレームワークが多層で、かつプラットフォームが固定されている場合のソフトウェアファクトリ（例：　マイクロソフト社「Software Factories」）

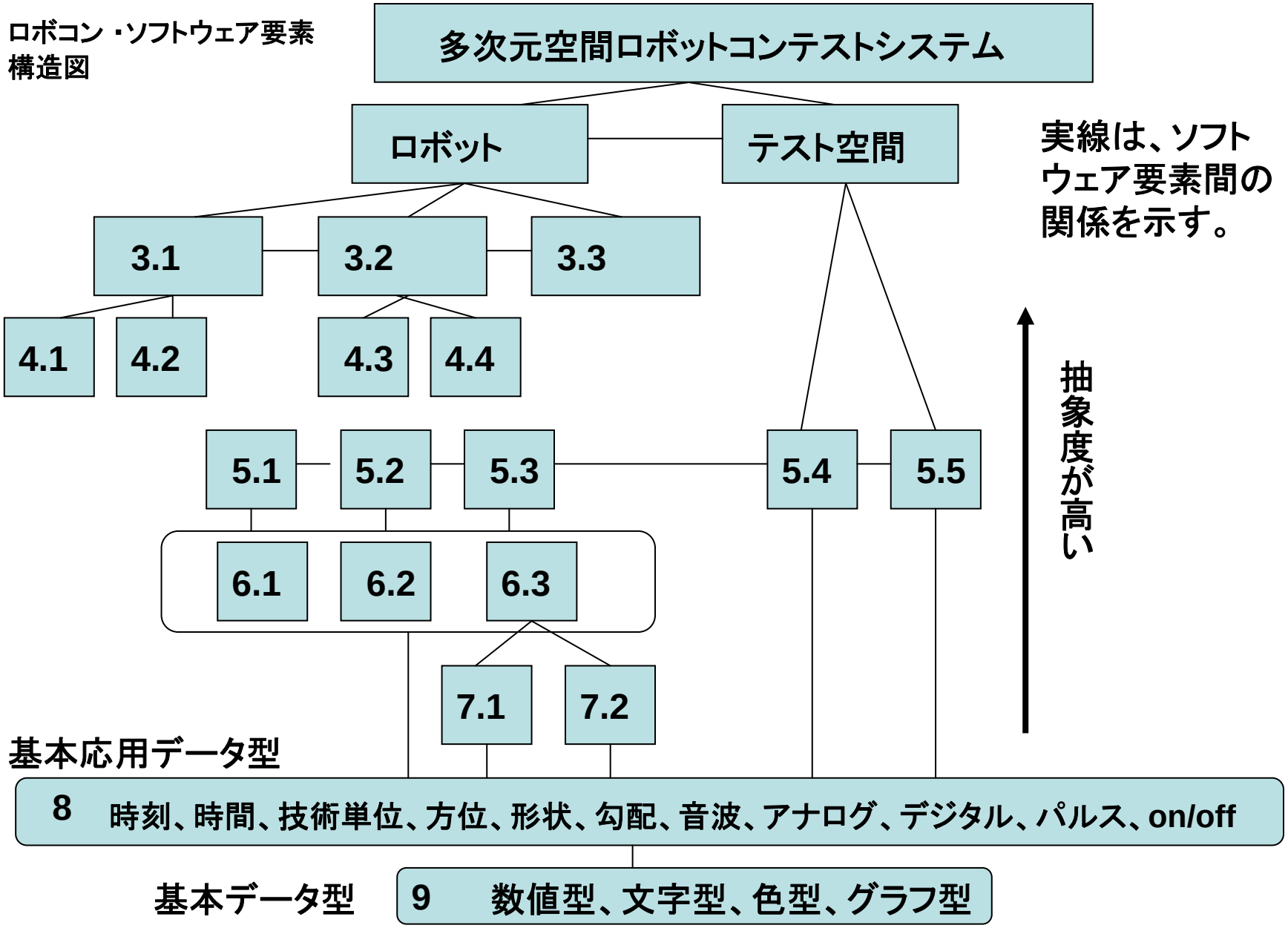
第1ステップ （例によって説明）

- ETロボコン・ソフトウェアと、MDDロボコン・ソフトウェアを、ひとつのプロダクトラインとする。
- 過去に開発したソフトウェア要素を収集、共通性、依存性(dependency)を分析して、グルーピングする。
- グルーピングされたソフトウェア要素を、再利用資産とする。

第2ステップ (例によって説明）

- ソフトウェア要素がもつ抽象度を識別し、抽象レベルによって視覚化する構造図を作成する。構造図の図法に決まりはない。たとえば、つぎのようにレベル分けする。
  - 0レベル： 多次元空間ロボットコンテストシステム
  - レベル1： ロボット、テスト空間
  - レベル3： ロボット静的特性3.1、 ロボット航行3.2、 ロボット保守3.3、
  - レベル4： 概念特性4.1、 物理特性4.2、手動航行4.3、 自動航行4.4
  - レベル5： 表示5.1、 制御5.2、 記録5.3、空間静的特性5.4、 コンテスト条件5.5
  - レベル6： 時間計測6.1、物理量計測6.2、操作6.3、
  - レベル7： 駆動操作7.1、操舵操作6.4
  - レベル8： 時刻、時間、技術単位、方位、グラフ、勾配、音、アナログ信号、デジタル信号、ON/OFF、パルス、表
  - レベル9： 数値型、文字型、色型、グラフ型

ロボコン・ソフトウェア要素
構造図



第3ステップ

- ソフトウェア要素構造図を、上下逆転し、クラス図に変換する。クラス図では、上位にあるクラス属性を下位のクラス属性が継承する。
- ソフトウェア要素構造図の抽象レベルとクラス図の継承レベルは、合致していることが望ましい。
- 各抽象レベルを、あらかじめ予定している（過去の資産）ソフトウェアアーキテクチャに写像する。
- ソフトウェアアーキテクチャを、フレームワーク（たとえば、.NET）の対応する層（tier）にそれぞれ写像する。
- このステップでは、資産のなかにあるソフトウェアアーキテクチャ・パターンを利用する。

3層フレームワークの例

Microsoft's Windows Mobile 5.0
Microsoft's Compact Framework 2.0

User Experience

Business Logic

Data & Web Svc

Management

Security

Connectivity

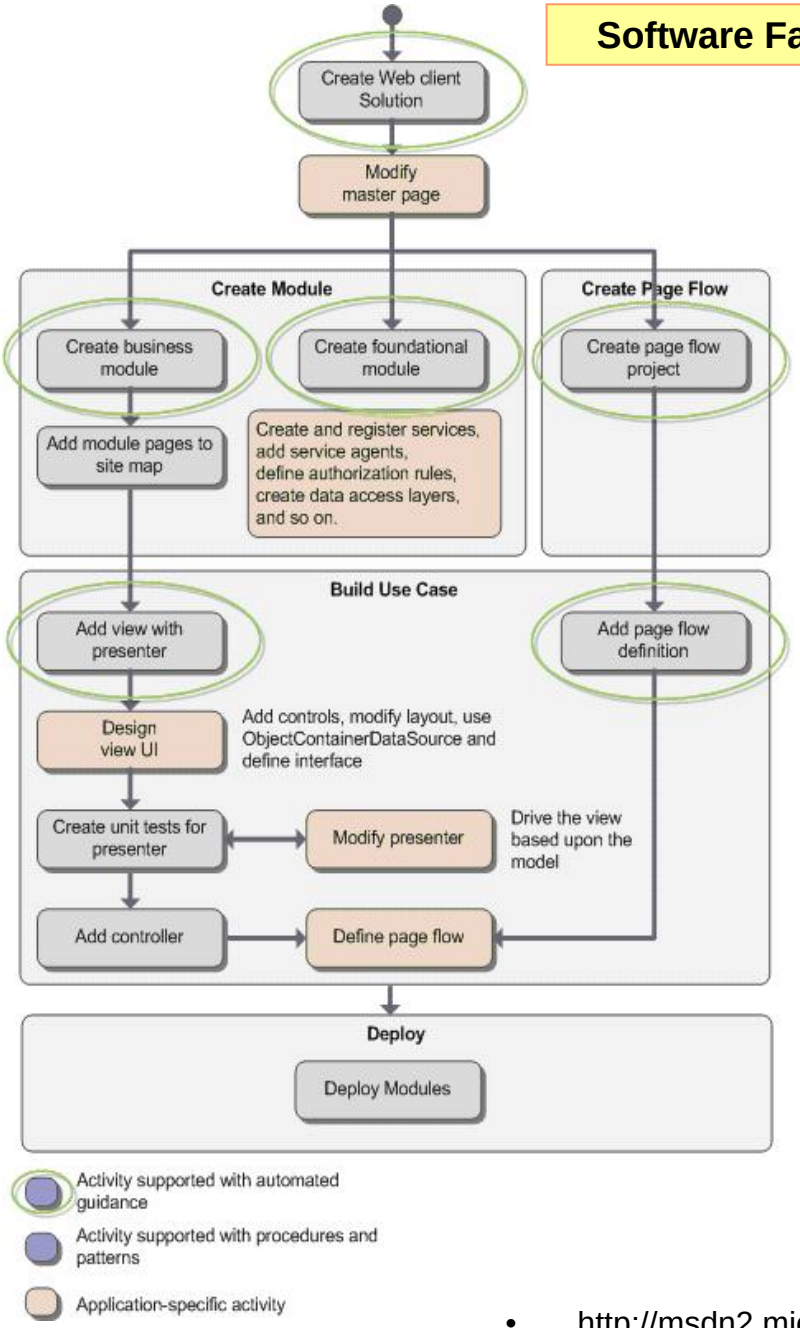
Infrared radiation communication

Taken from Microsoft's web site

第4ステップ

- クラス図のなかの属性と定義する。ここでは、GOFデザインパターンで推奨されてる「テンプレート・パターン」を利用して、共通部と可変部を分けて定義する。
- 共通部を、テンプレート言語で記述する①。
- テンプレート記述のなかで定義された可変要素に、値を割り付ける②。
- ①からプログラムジェネレータを自動生成する。
- ②の記述を、プログラムジェネレータに入力して、目的プログラムを自動生成する。
- 上記2項の自動生成に必要なツールは、MSソフトウェアファクトリのなかで提供されている。
- 以下のステップは、つぎのスライドを参照されたい。

Software Factoriesを用いたアプリケーション開発手順 (1)



1. GAT (guided automation toolset) から、Web Client Solutionというフレームワークを選び、提示されるmaster page patternをカスタマイズし、master pageを開発する。
2. GATからbusiness module patternで類似のものを選び、カスタマイズし、business moduleを開発し、必要なページをサイトマップに加える。
3. GATが支援していないモジュールは、自分で開発する。ただし、これらモジュールは、サービスとして形成し、登録することが望ましい。サービスに対しては、サービスエージェントを作る必要があり、この作業はGATによって支援される。
4. モジュールおよびサービスに関しては、authorization rulesをGATの支援下で作成する。
5. モジュールおよびサービスがデータ管理（データベースなど）である場合に、データアクセス層を作成する。

- http://msdn2.microsoft.com/en-us/library/bb264518.aspx

Software Factoriesを用いたアプリケーション開発手順 (2)

6. GATを利用してページフローを作成する。

-- 以上がファクトリを構成する基本要素の設定プロセスであり、これ以下のプロセスは、顧客ニーズに直結するユースケースの構築（ファクトリの垂直系統開発）プロセスである。

7. VMC (view-model-control) モデルに従って、まずviewを定義し、そのpresenterを設計する。viewの組み込みはGATが支援。

8. view UIを設計する。これにはGATの支援がないので、手作りである。controlsを設定し、配置を定義し、ObjectContainerDataSourceを使ってコンテナで包む。最後にインタフェースを定義する。

9. viewを実行させることによってpresenterをテストする。

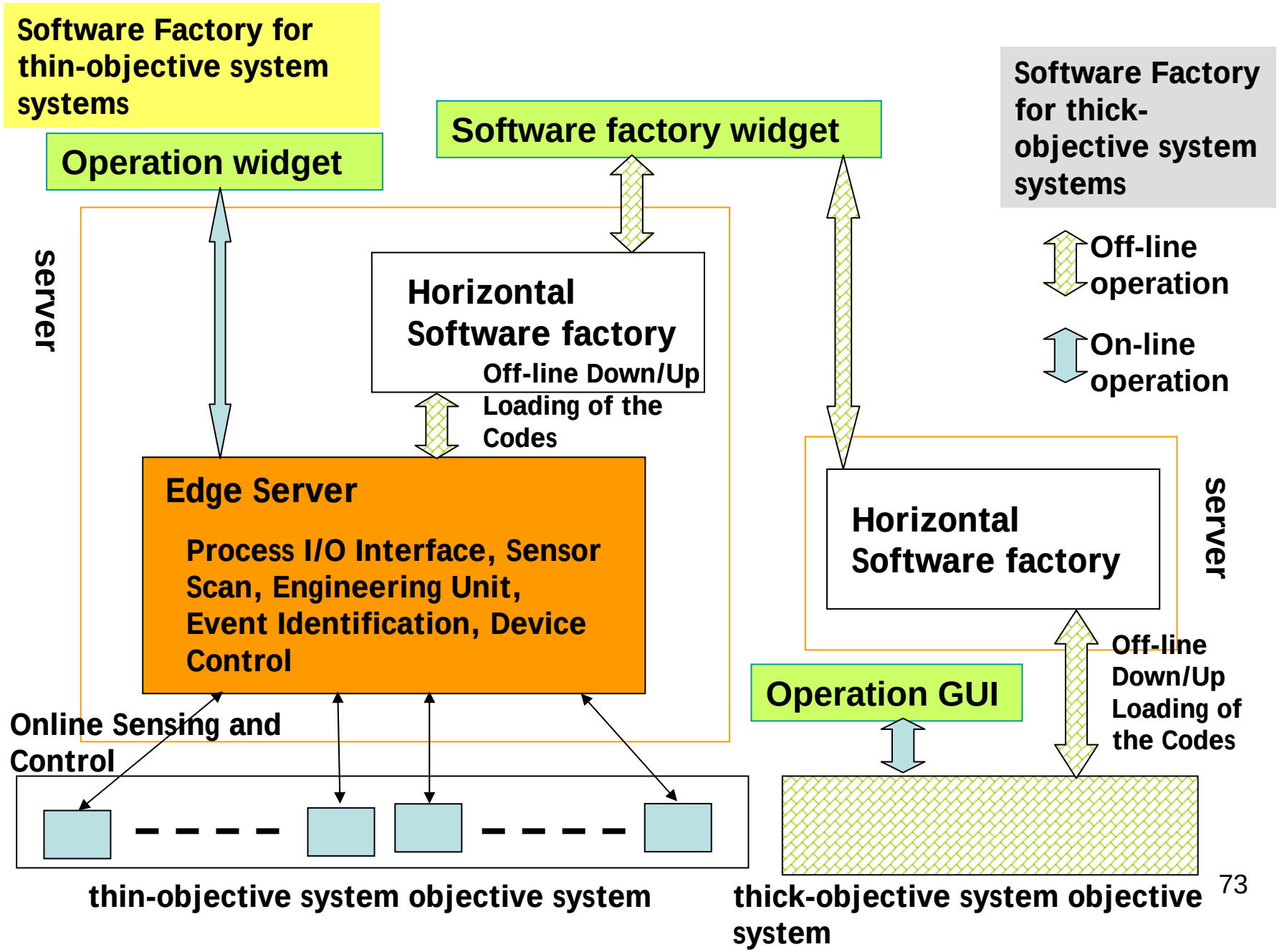
10. controllerを設定する。

11. GATを使って、先に作成したpage flow 定義を設定する。

12. page flowを定義する。

13. 実装モジュールを生成する。

Two types of Software factory



サービス （ソフトウェアファクトリが利用するモジュール）

- サービスとは
  - 人の知的活動による実世界に対する価値創造（改善を含む）を支援するために、第三者によって提供される支援（ここでは、情報および情報通信技術に限定）
- イノベーションとは
  - 社会・経済が沈滞しないようにするために必要な、各種資源（科学、技術、産業、社会、経済など）の新しい結合の定着
  - サービス・イノベーションに関係する領域は：
    - 技術、ビジネス、社会・組織、市場経済など
- サービス・サイエンス
  - サービス・イノベーションを推進するために必要な科学
    - 情報および情報通信技術、組織文化、ビジネスを軸に体系化

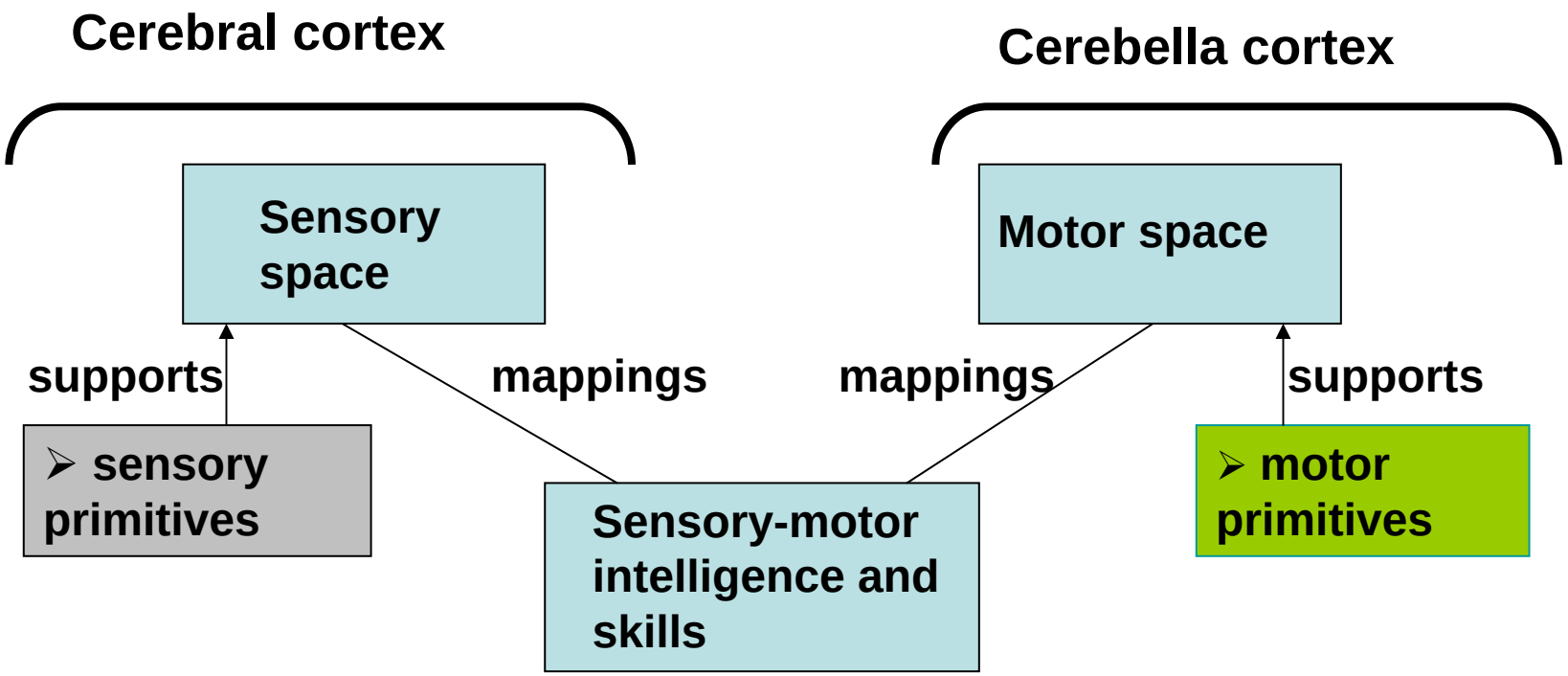
注： 実装方法としては、SOA （Service-Oriented Architecture), SaaS (Software as a Service)

第6章　フレームワーク1層、可変プラットフォームの場合の
ソフトウェアファクトリ
（東芝ソフトウェアファクトリ）

詳細は、つぎのURLを参照ください。
http://splc2007.jaist.ac.jp/SPLC2007MatsumotoKeynote.pdf

メタモデル（1）

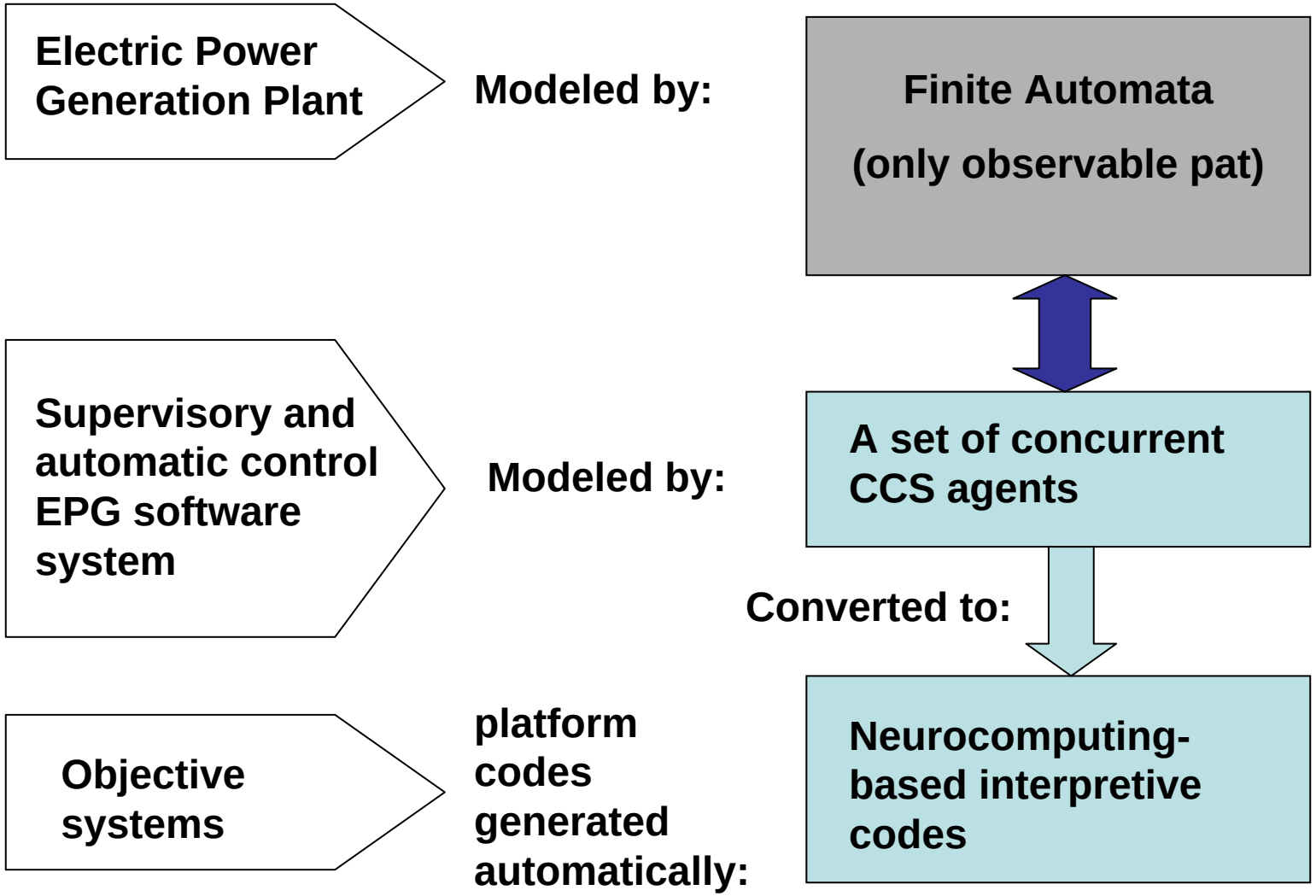
- From the aspect of Human-Centered Computing:
  - our domain engineering ≒ Sensory-motor anthropologic engineering



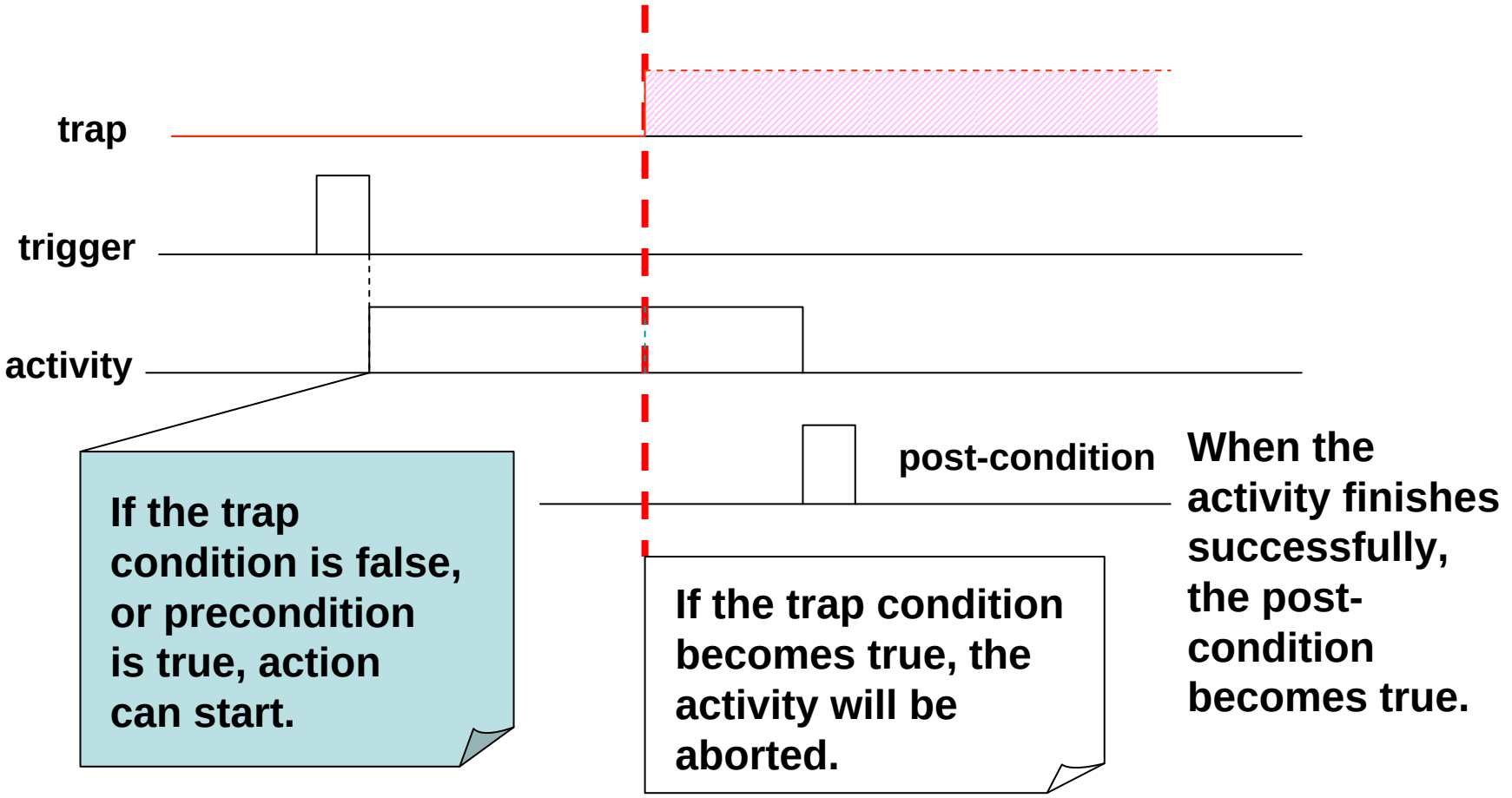
The core assets = Sensory-motor representation, sensory and motor primitives, mappings between both spaces and the execution platform

メタモデル（2）

- **CCS: Milner, R., A Calculus of Communication Systems, LNCS-92 (1980)**

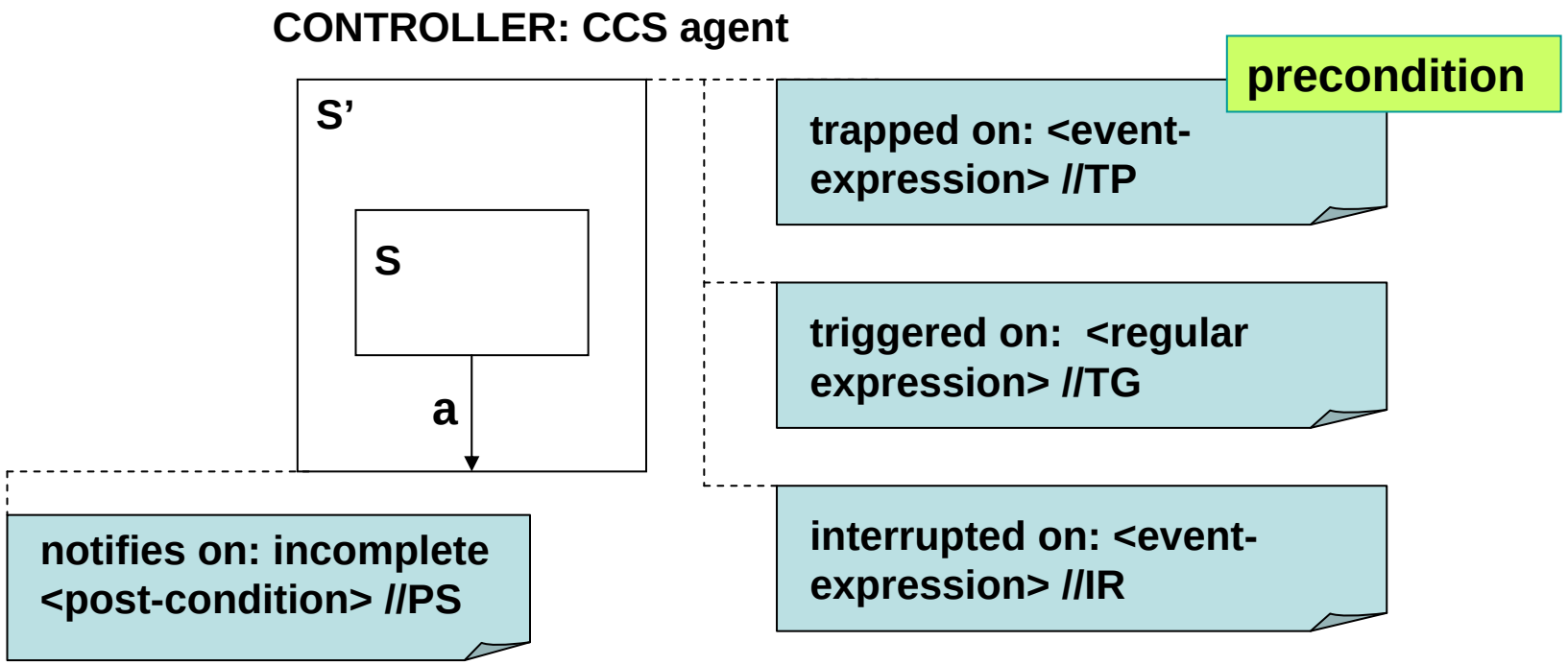


メタモデル（3）



Matsumoto, Y., A Method of Software Requirements Definition in Process Control, Proc. COMPSAC77, pp.128-132 (1977)

メタモデル（4）



Yoshihiro Matsumoto, A Guide for Management and Financial Controls of Product Lines, Proceedings of SPLC2007, IEEE Computer Society (2007)

DSL

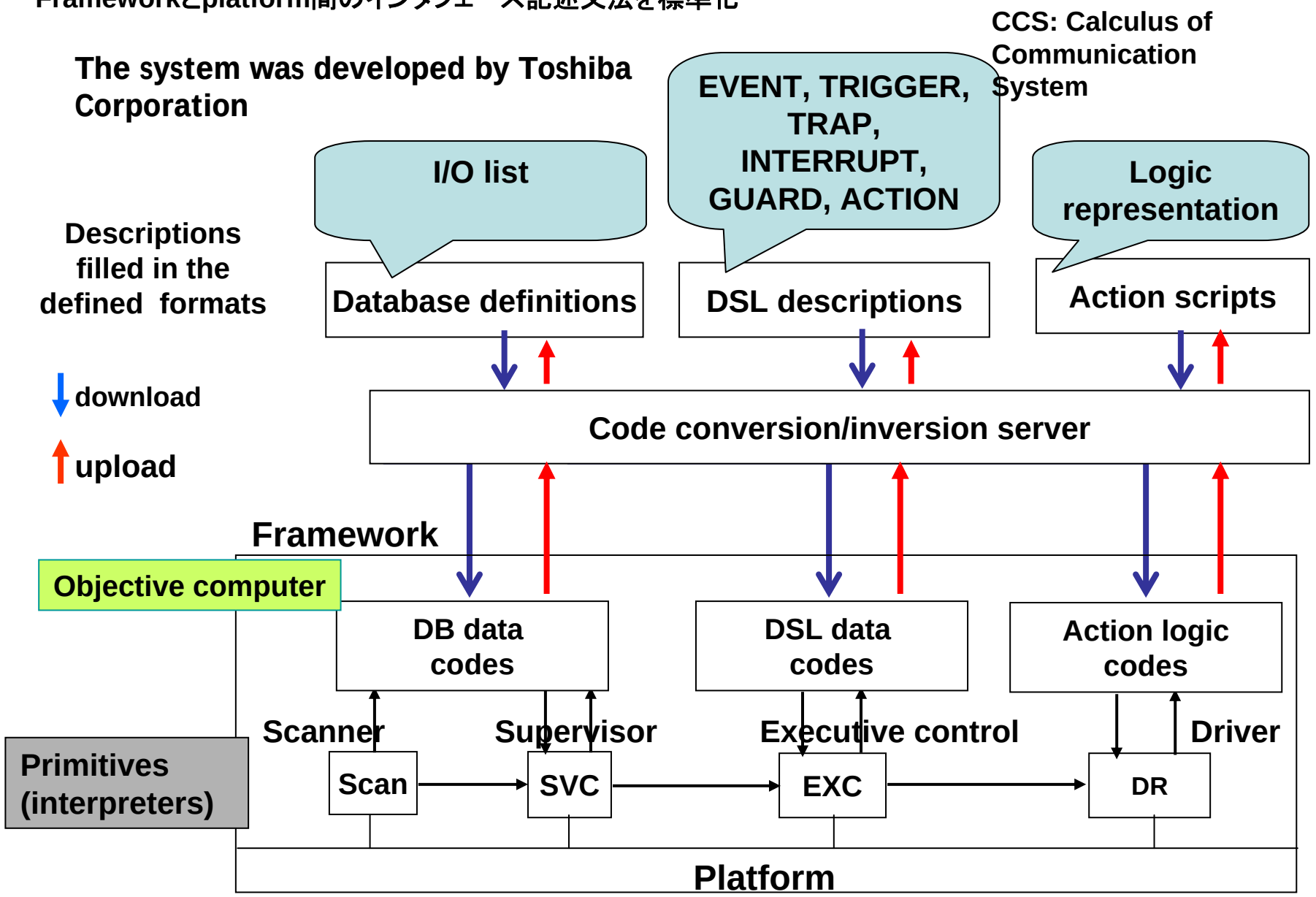
More than 6 thousands sheets of tables were produced per project.

The agents are described using a set of table formats. The described tables are called "plant tables".

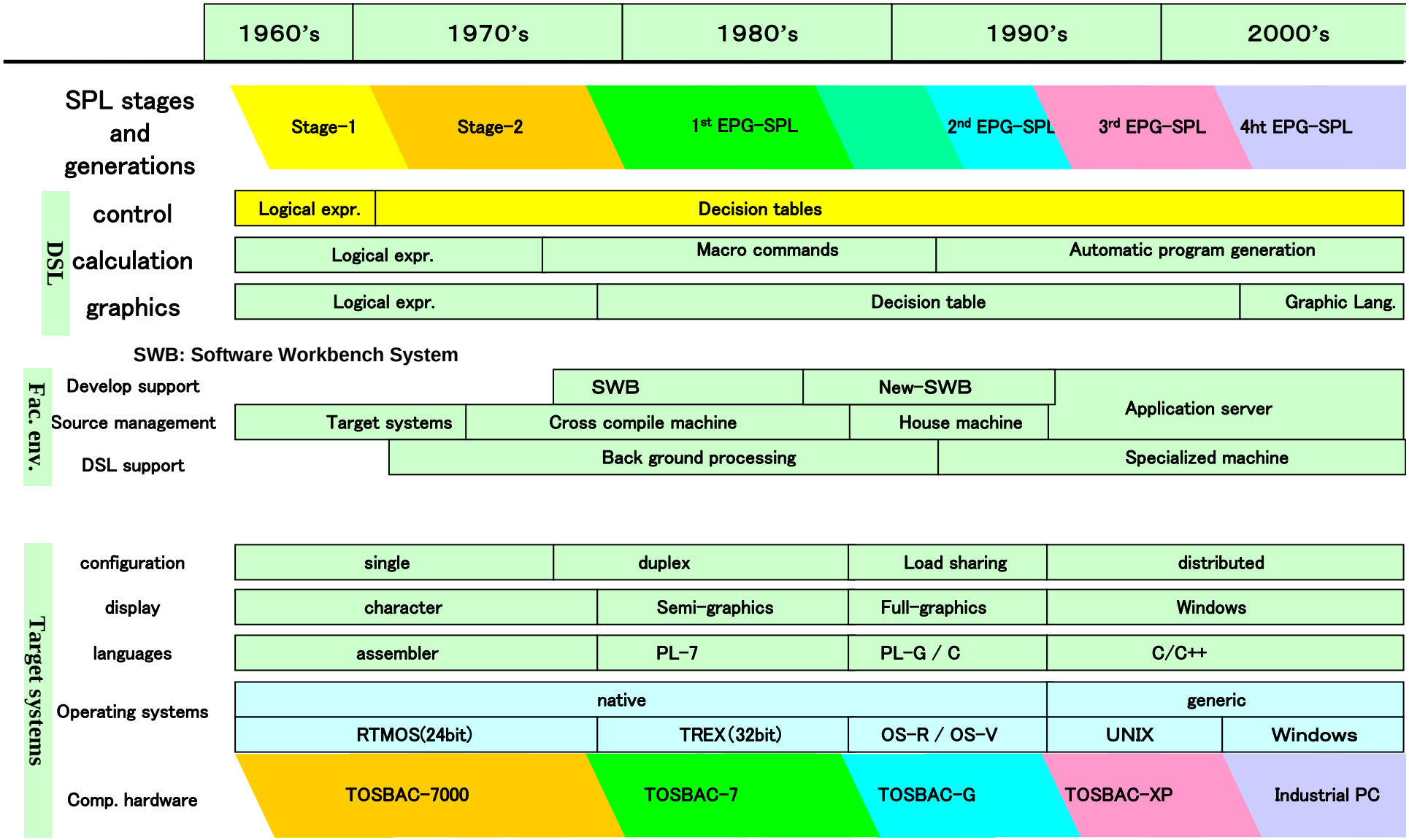
The six kinds of tables, to be described in the fill-in-the-blank formats are:

- **✓Plant Master Status table (PMS: TRAP) – event notation,**
- **✓Macro Status Determiner (MSD: EVENT) – macro event notation,**
- **✓Master Control Sequencer definitions (MCS: TRIGGER) – trigger notation,**
- **✓Input/Output List (I-O List: cerebral data),**
- **✓ Alarm Group definitions (ALG: ACTION) – action notation for alarm, and**
- **✓Operation Block definitions (OB: ACTION) – action notation for control.**

Frameworkとplatform間のインタフェース記述文法を標準化



Generations of the EPG products



Windows: Microsoft's trademark

- 解 (solution) に関するメタモデルとしてCCMを採用しているため、解が、数学的に検証可能であること。
- DSLは、このメタモデルをベースに設計されているので、恒久性が維持できる。
- プラットフォームが7~8年ごとに変化するが、DSLを含む垂直ソフトウェアファクトリは、まったく変更する必要がないように、変化をフレームワークで吸収する構造となっている。
- フレームワークを変更するプロセスは工業化されており、水平ソフトウェアファクトリが変更を支援する。

むすび

- 小さなIT企業しか存在しない国（たとえばメキシコやタイ）がSPI (software Process improvement) 活動を行い、その結果、ISO/IEC JTC1 WG24 *Software Life Cycles for Very Small Enterprises*を編成し、新しい標準を作ろうとしている。
- ソフトウェア事業のオフショア化、国際化を進展させるために、80年代からソフトウェア工業化に実績があり、また新しくソフトウェア工業化を進めている日本の企業が連携して、工業化プロセス標準の開発を行い、ISOへ提案することが考えられないであろうか。