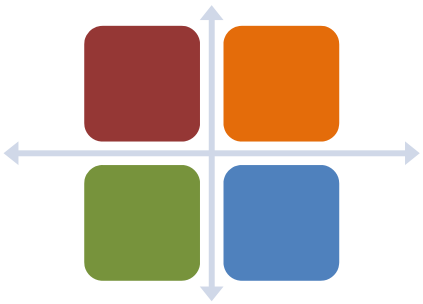


ソフトウェアアーキテクチャの知識の作成と管理

Philippe Kruchten

December 2009

2

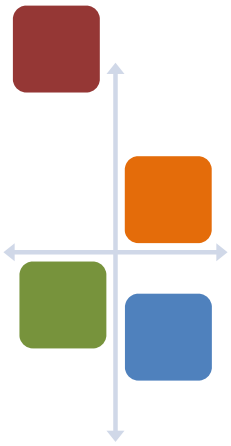
Philippe Kruchten, Ph.D., P.Eng., CSDP
フィリップ・クルーシュテン

Professor of Software Engineering
NSERC Chair in Design Engineering
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC Canada
pbk@ece.ubc.ca
+1 604 827-5654

Founder and president

Kruchten Engineering Services Ltd
Vancouver, BC Canada
philippe@kruchten.com
+1 604 418-2006

3



アウトライン

- ソフトウェアアーキテクチャ
- 知識マネジメント
- 動機
- アーキテクチャ表現
- AK = AD + ADD
- プロセス
- ツール
- まとめ

4

ソフトウェアアーキテクチャ

ソフトウェアアーキテクチャは、以下のような大きな設計判断を含んでいる

- ソフトウェアシステムの組織,
- 構造的要素の選択と(それらの要素の間のコラボレーションとして仕様化される振る舞いから作られる)システムに対するそれらの要素のインタフェース,
- それらの要素から段階的により大きなサブシステムの作成,

Grady Booch, Philippe Kruchten, Rich Reitman, Kurt Bittner; Rational, circa 1995
(derived from Mary Shaw)

ソフトウェアアーキテクチャ(続き)

- …
- アーキテクチャの組織、要素、インターフェース、コラボレーション、構成を導くアーキテクチャスタイル.
- ソフトウェアアーキテクチャは構造と振る舞いだけではなく、その使い方、機能、性能、回復力、再利用、包括性、経済的や技術的な制約やトレードオフ、美学も対象にしている.

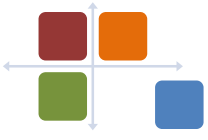
 6

ソフトウェアアーキテクチャ…

- アーキテクチャ = { 要素, 形式, 根拠 } *

Perry & Wolf 1992

- 骨格
- 構造以上のもの
- 多くの“～性(ilities)”を具体化、解決する
- 実行可能で、それゆえ検証可能
- 現れる? …. 時として…



 7

知識

- 経験や教育を通して獲得された専門的知識とスキル; 主題についての理論的または実践的理解
- 特定の分野または全体的に知られていること; 事実と情報
- 事実や状況に対する経験により得られる自覚と精通

- プラトン: 知識は “正当化された本当の信念” である

 8

資産としての知識

- “知的資本”
- 知識労働者
- “知識は力だ”

- 知識管理:
  - 組織の知識を共有、分配、生成、補足、理解する

 9

"知的資本の大きな問題は足を持ち、日々帰宅することである."

Rus & Lindvall 2002

 11

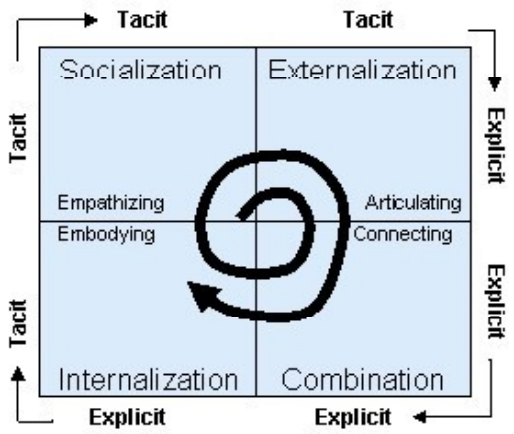
暗黙知

vs.

形式知

 12

SECIモデル



Nonaka & Takeuchi 1995

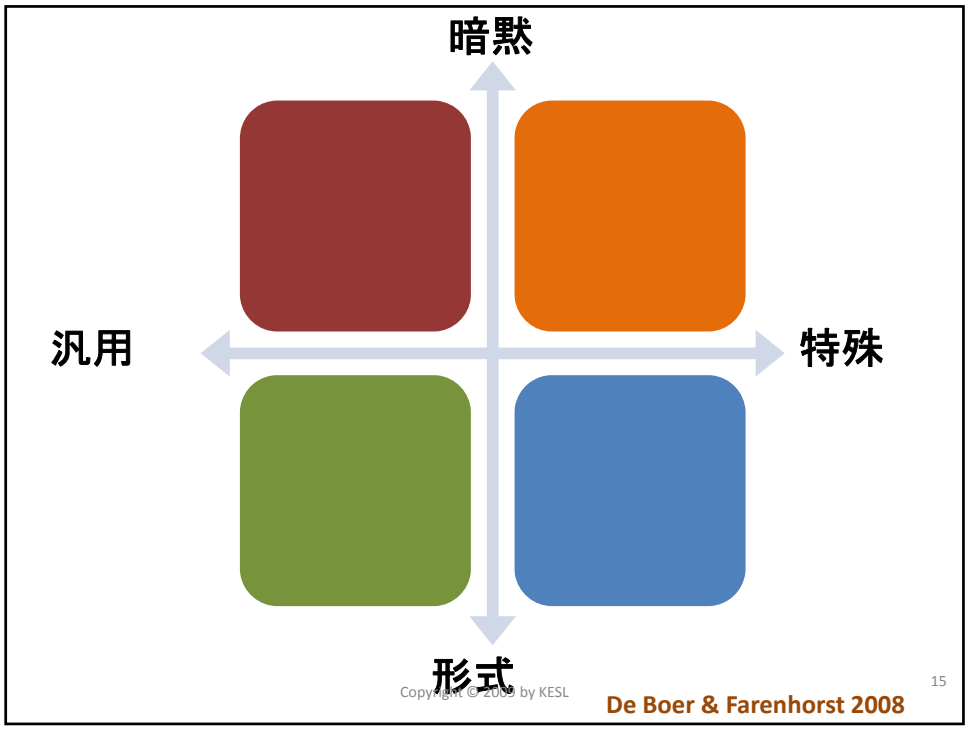
13

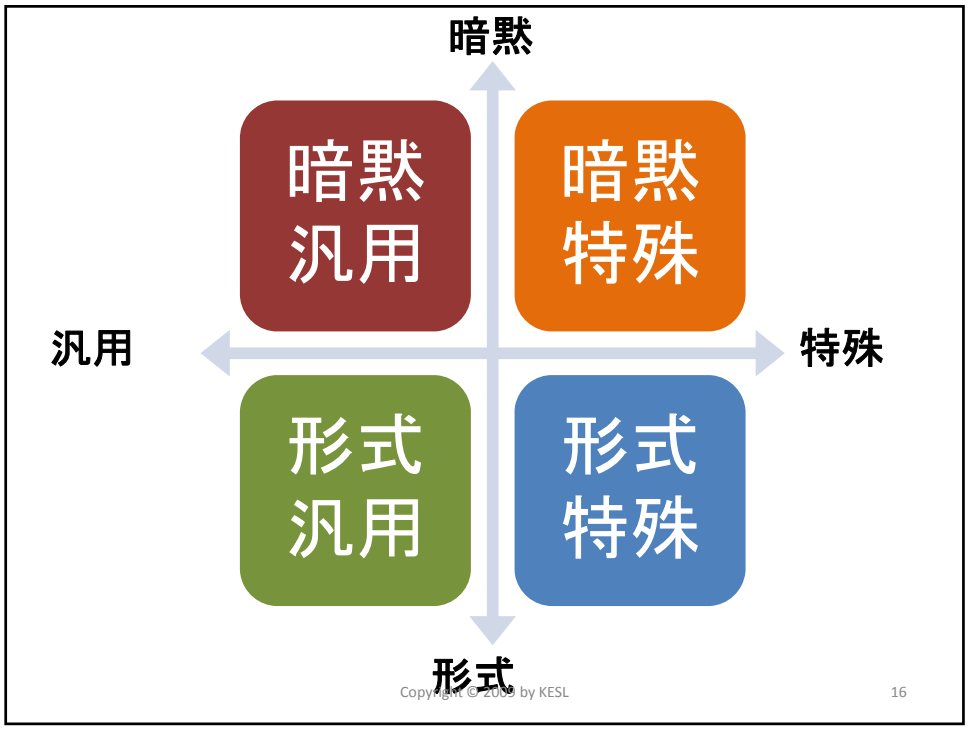
システム – 特殊な知識

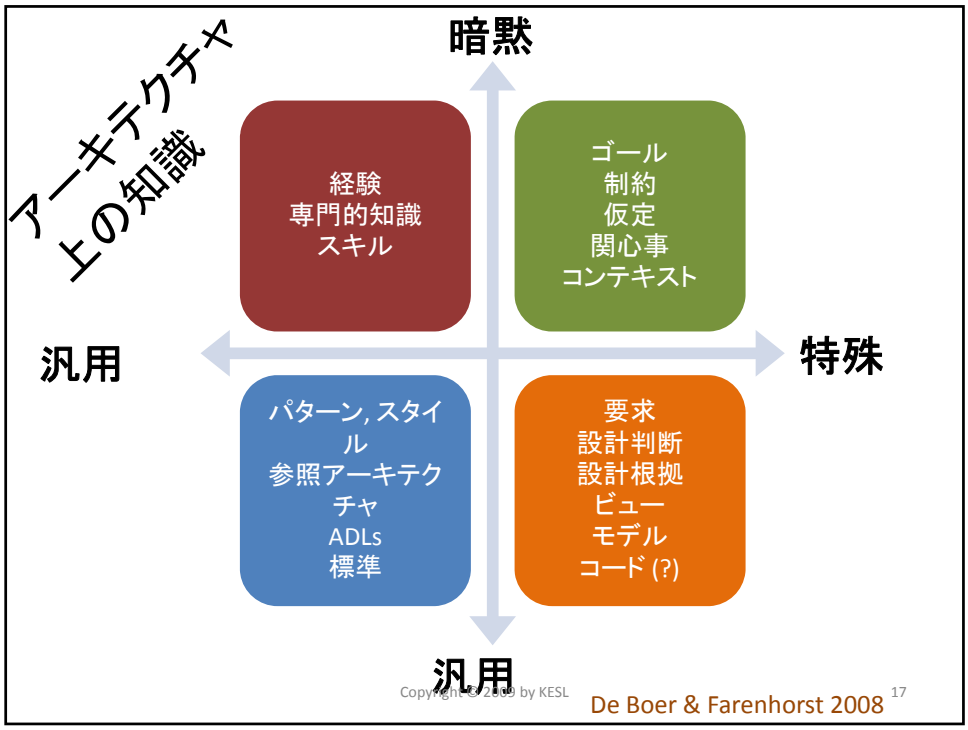
vs.

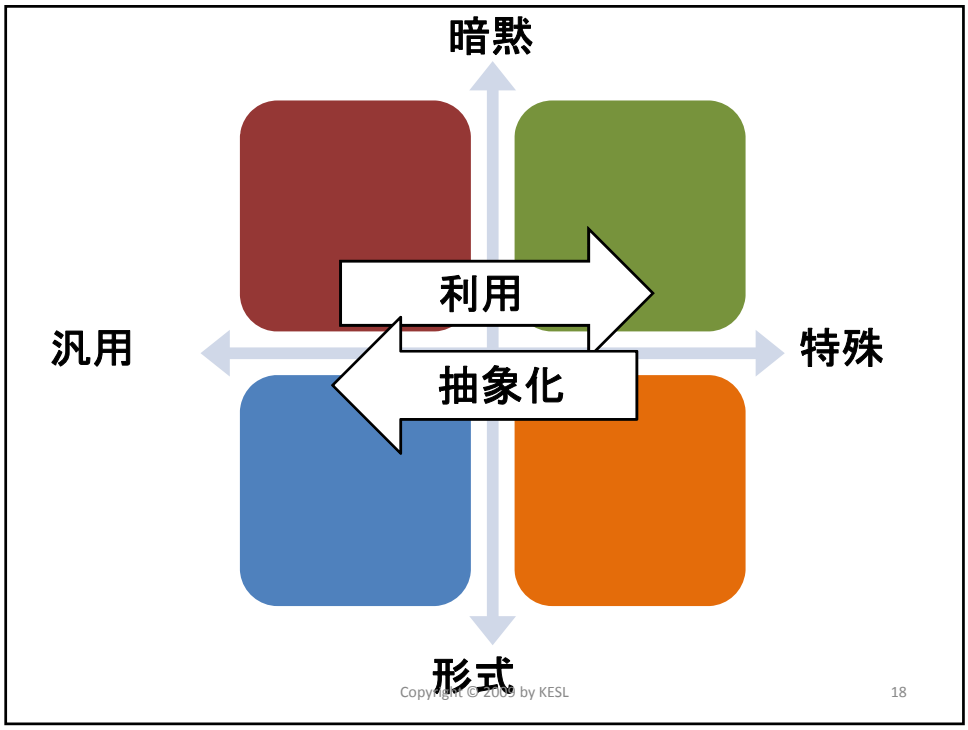
システム – 汎用的な知識

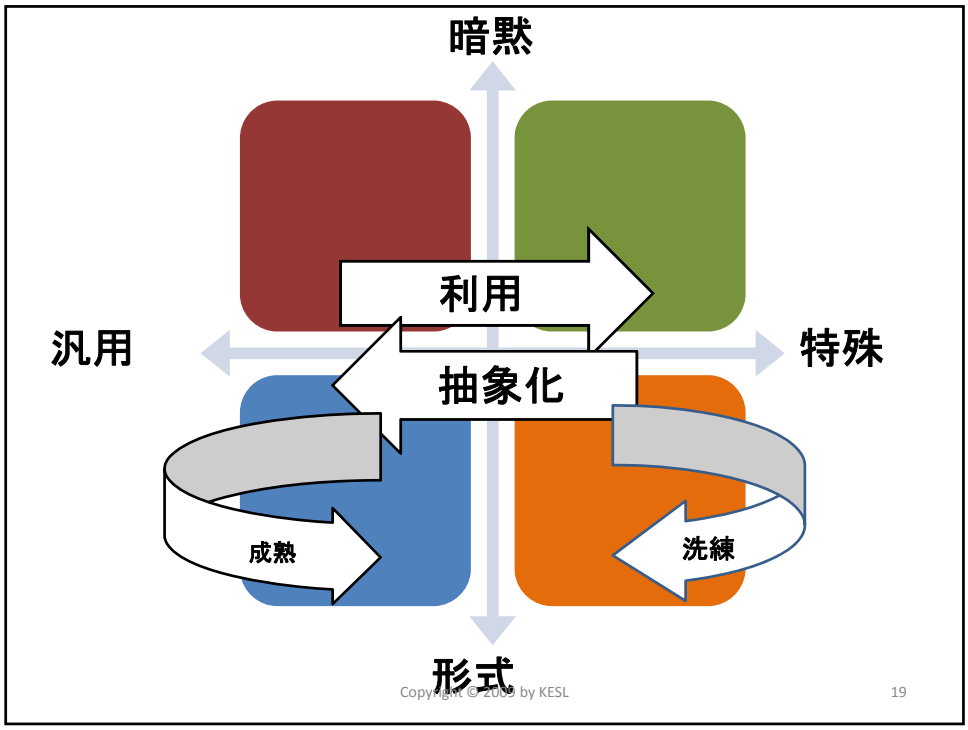
14


暗黙
汎用
特殊
形式
Copyright © 2009 by KESL
15
De Boer & Farenhorst 2008


暗黙
暗黙
汎用
暗黙
特殊
汎用
特殊
形式
汎用
形式
特殊
形式
Copyright © 2009 by KESL
16


暗黙
アーキテクチャ上の知識
経験
専門的知識
スキル
ゴール
制約
仮定
関心事
コンテキスト
汎用
特殊
パターン, スタイル
参照アーキテクチャ
ADLs
標準
要求
設計判断
設計根拠
ビュー
モデル
コード (?)
汎用
Copyright © 2009 by KESL
De Boer & Farenhorst 2008
17


暗黙
利用
汎用
特殊
抽象化
形式
Copyright © 2009 by KESL
18

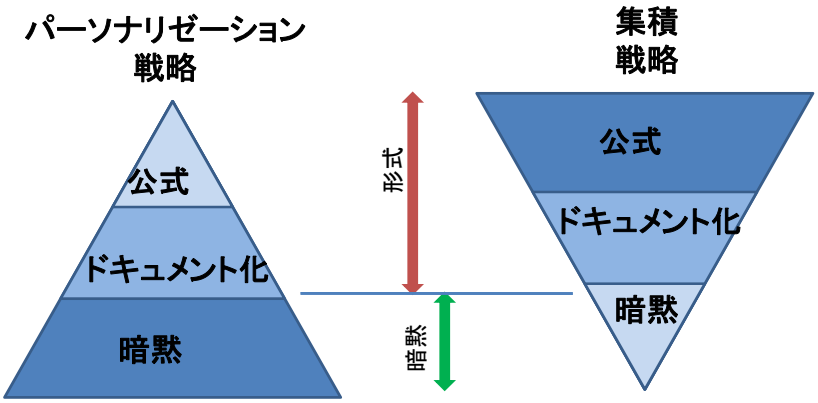


知識マネジメントの戦略

- 集積
  - 情報を補足し、保管し、取り出す
- パーソナリゼーション
  - 誰が何を知っているかを定義する, “イエロページ”

Hansen et al. 1999

戦略 & 知識タイプ



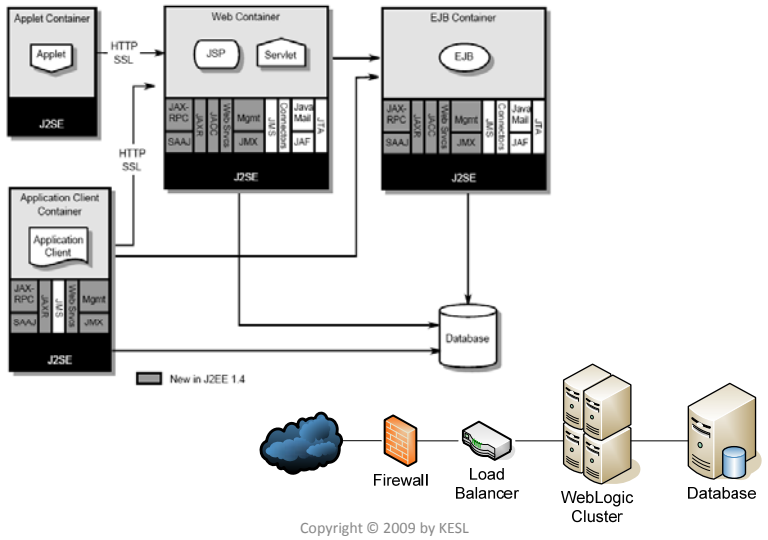
 21

ソフトウェアアーキテクチャの歴史

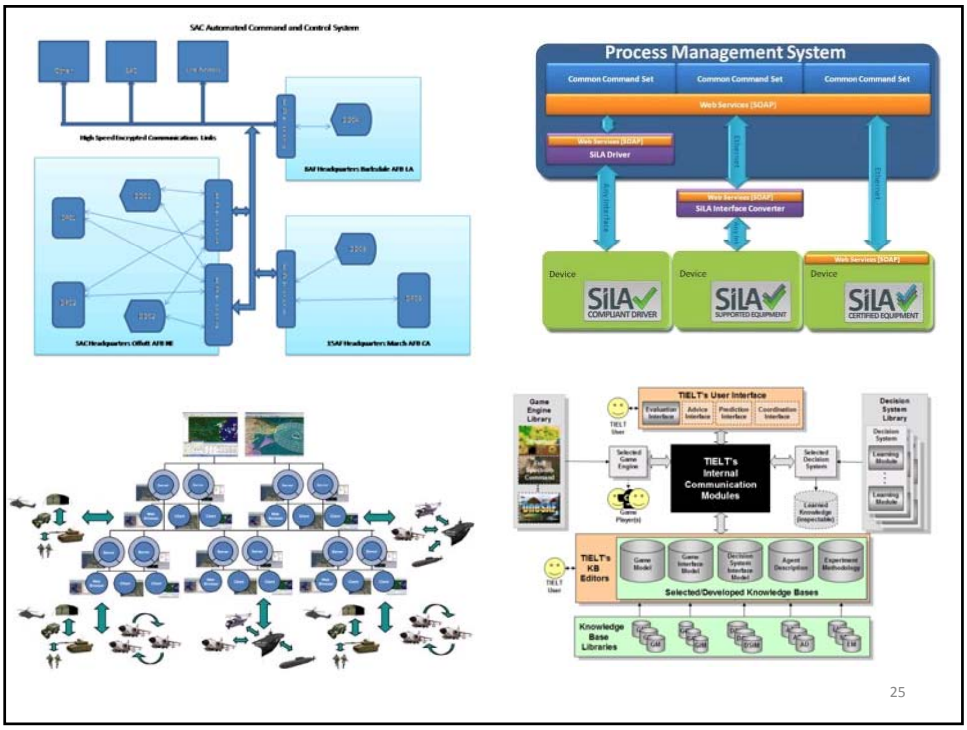
- NATOカンファレンス(1969)
- 箱（ボックス）と矢印(1960s-1980s)
- ビューとビューポイント(1990s-2000)
- ADLs (1980s-2000s)
- アーキテクチャ設計手法(1990s-2000s)
- 標準, 参照アーキテクチャ(1995-…)
- アーキテクチャ設計判断(2004-…)

 23

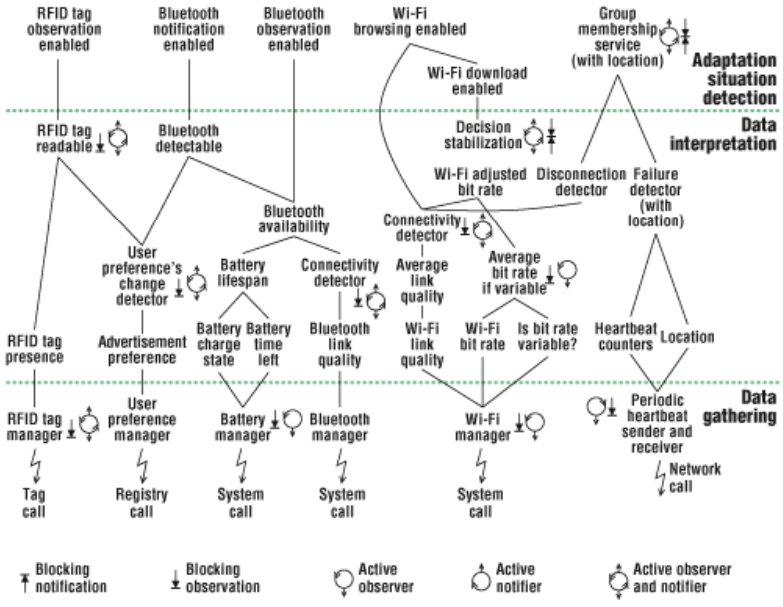
箱（ボックス）と矢印（アロー）



Copyright © 2009 by KESL 24

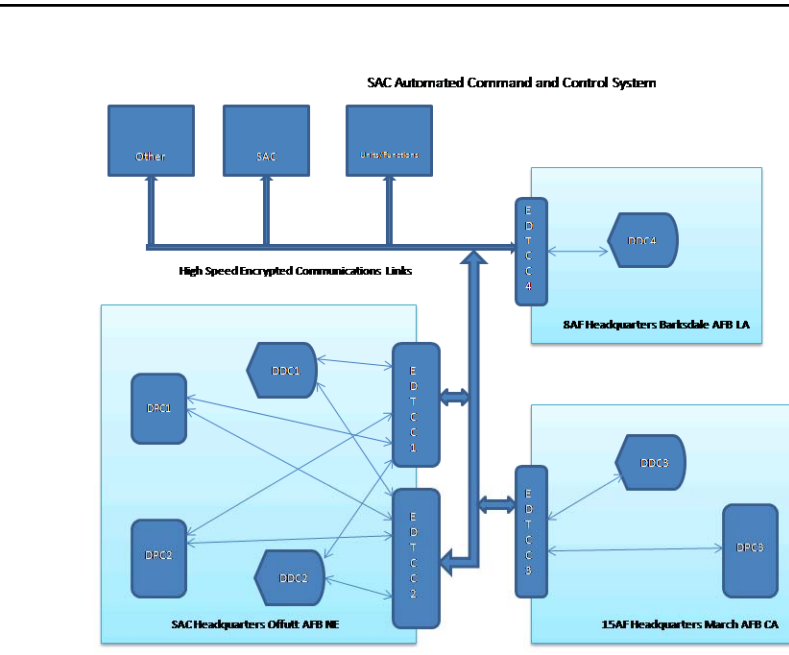


25

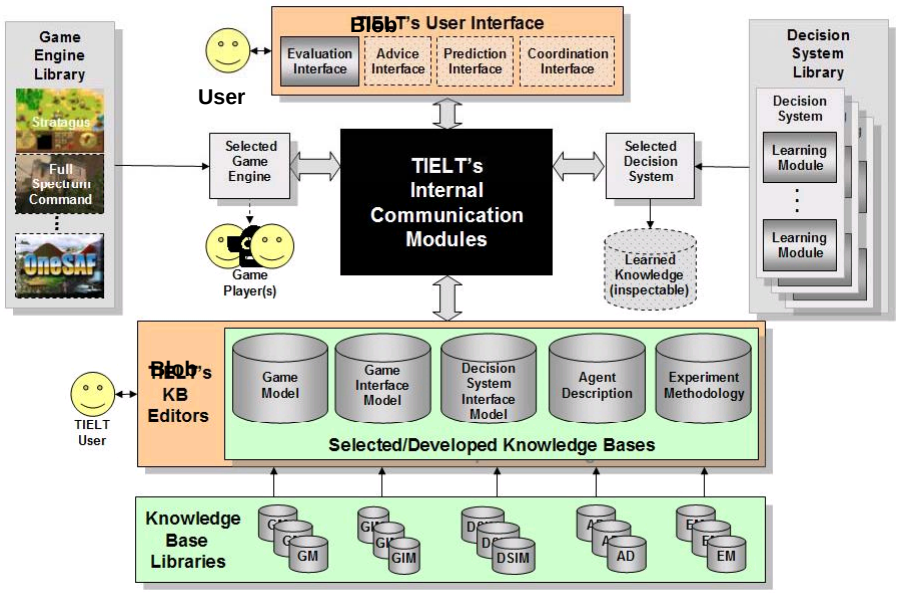


Source: IEEE DNS Online

26



27



28

ボクソロジー（Boxology ）の問題

- 一般的な“メッセージ” やメタファーはOKだが…
- 意味があいまい:
  - 箱は何を表しているのか?
    - 機能, コード, タスク, プロセス, プロセッサー, データ?
  - 矢印は何を表しているのかj?
    - データフロー, 制御フロー, 意味的な依存性, 配線?
- 解釈が発散する
- 多くの異なる関心事や問題に1枚の図で対処する

29

ビュー & ビューポイント

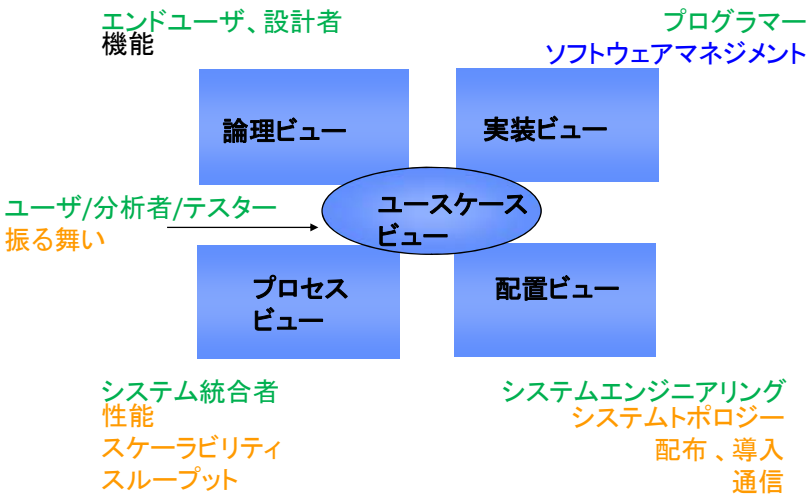
- Rational アプローチ
- S4V（Siemens社）
- BAPO/CAFR（Philips社）

- IEEE Std 1471:2000 Recommended practice for software architecture description
- ISO/IEC 42010: 2007 Recommended practice for architectural description of software-intensive systems
- ISO/IEC 42010: 2010 (?) Architectural description

- Clements et al. 2005, Documenting Software Arch
- Rozanski, N., & Woods, E. (2005). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives.* Boston: Addison-Wesley.

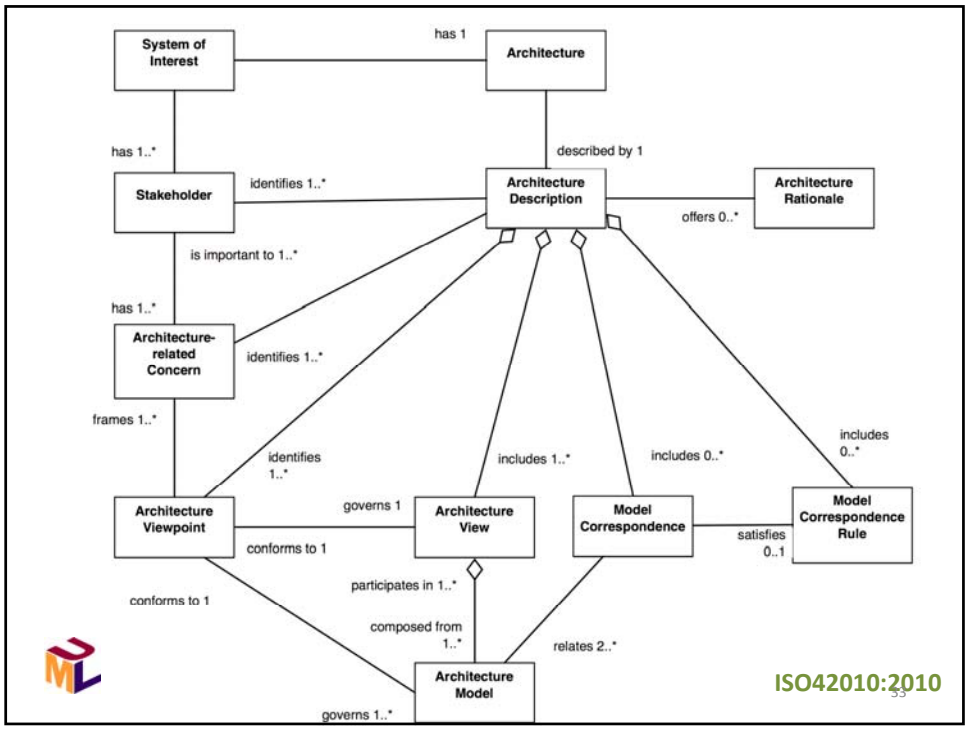
 31

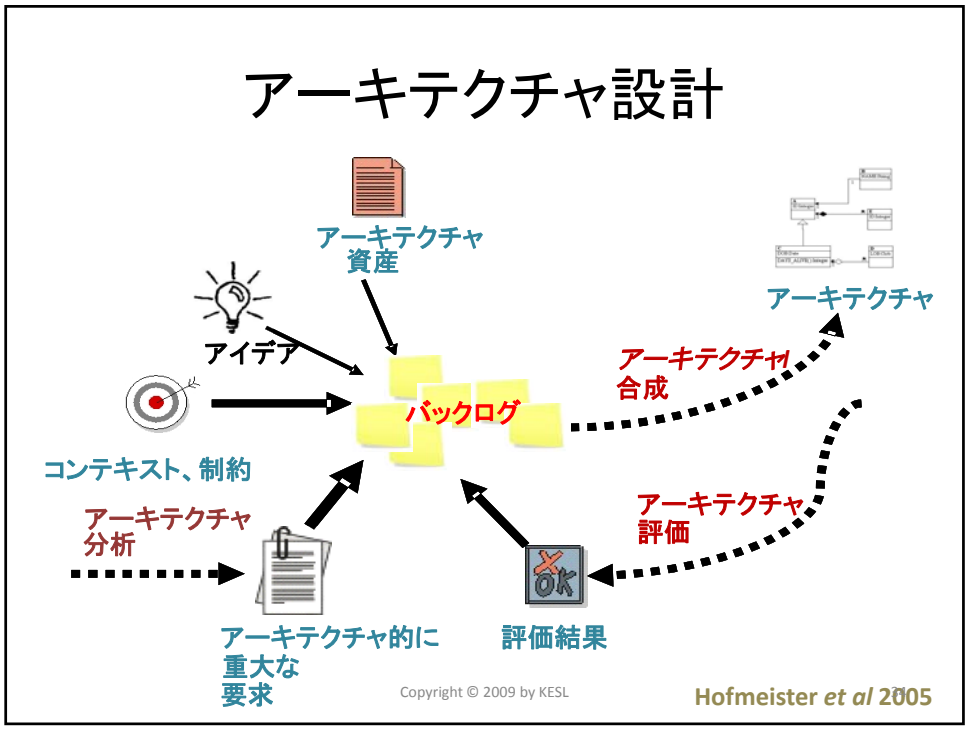
アーキテクチャの4+1ビューモデル



 32

Kruchten 1995

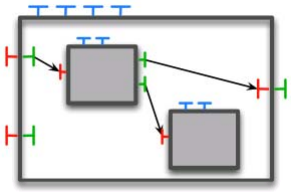

System of Interest
has 1
Architecture
has 1..*
described by 1
Stakeholder
identifies 1..*
Architecture Description
Architecture Rationale
offers 0..*
is important to 1..*
has 1..*
Architecture-related Concern
identifies 1..*
frames 1..*
identifies 1..*
includes 1..*
includes 0..*
includes 0..*
Architecture Viewpoint
governs 1
Architecture View
Model Correspondence
Model Correspondence Rule
satisfies 0..1
conforms to 1
participates in 1..*
conforms to 1
composed from 1..*
relates 2..*
Architecture Model
governs 1..*
ISO42010:2010


アーキテクチャ設計
アーキテクチャ資産
アーキテクチャ
アイデア
アーキテクチャ合成
バックログ
コンテキスト、制約
アーキテクチャ評価
アーキテクチャ分析
アーキテクチャ的に重大な要求
評価結果
Copyright © 2009 by KESL
Hofmeister et al 2005

アーキテクチャ記述言語
Architecture Description Languages

- Rapide (Stanford)

- ACME (CMU)
- Wright (CMU)
- C2 (UC Irvine)
- Darwin (Imp. Coll.) -> Koala
- Archimate
- AADL (based on MetaH)
- etc…



 35

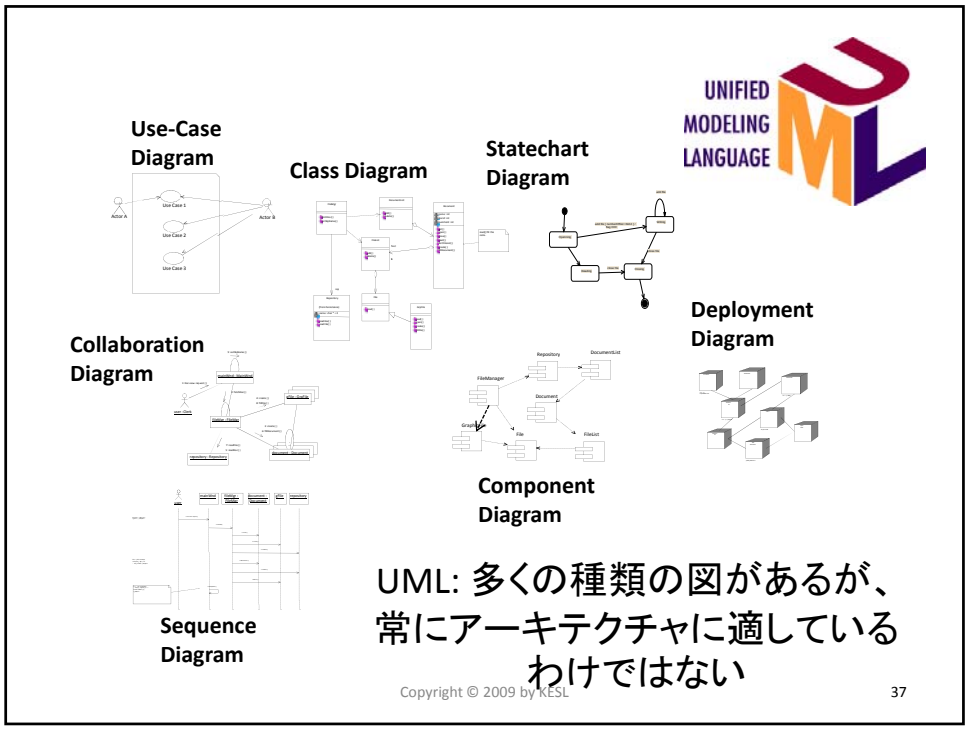
UML 2.0

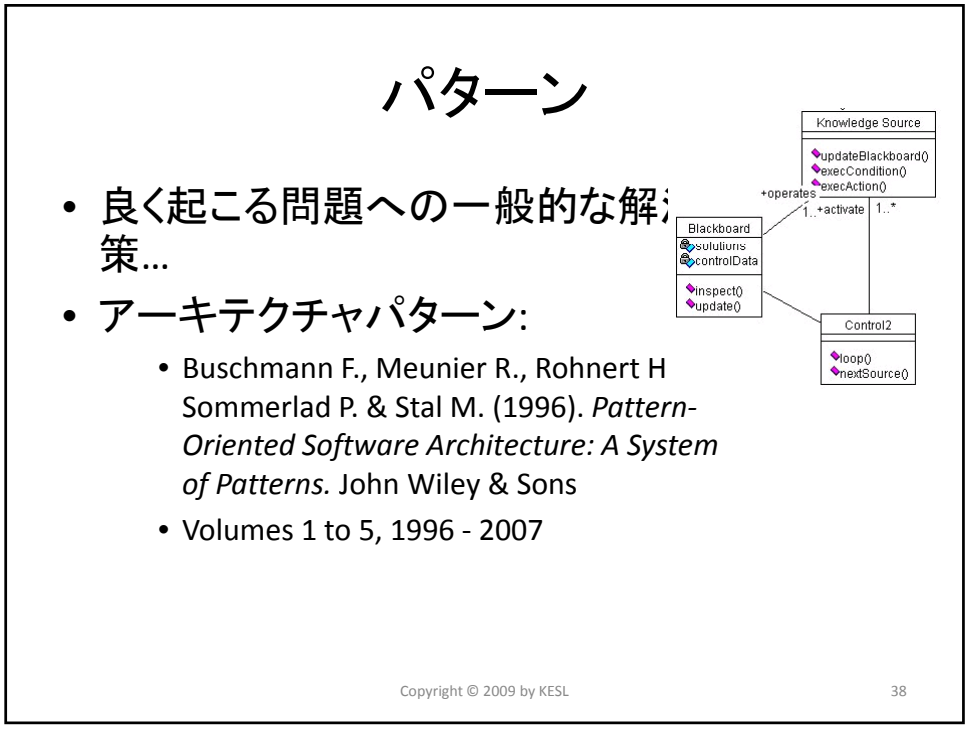
- 記法
- より良い“box and arrows”
- 意味がより明確
- ほとんどADL？

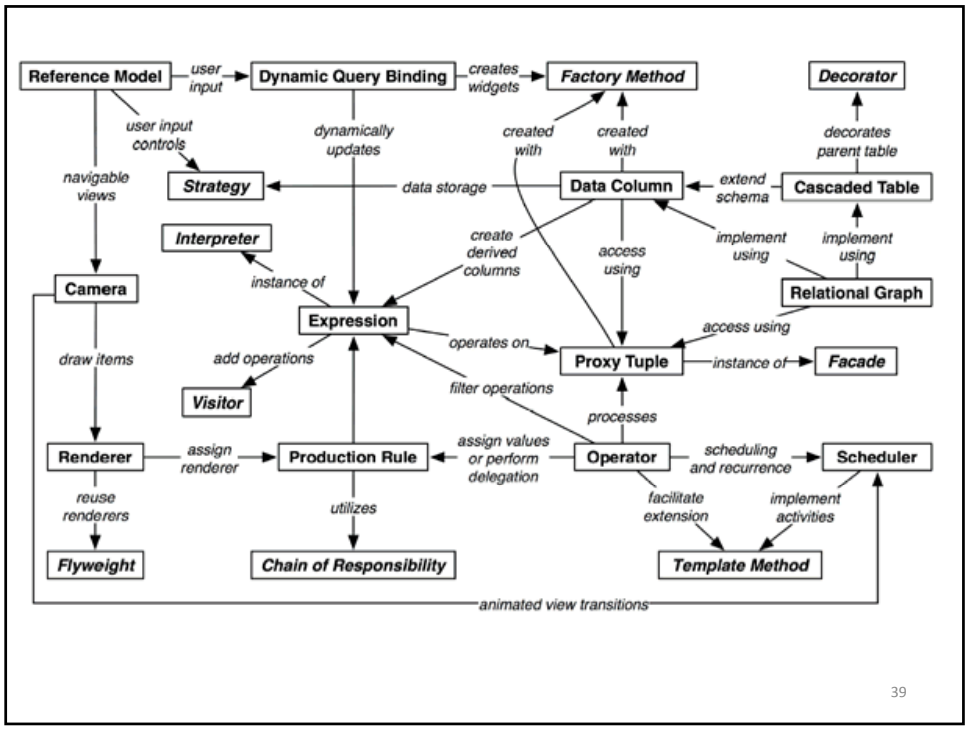
UNIFIED MODELING LANGUAGE

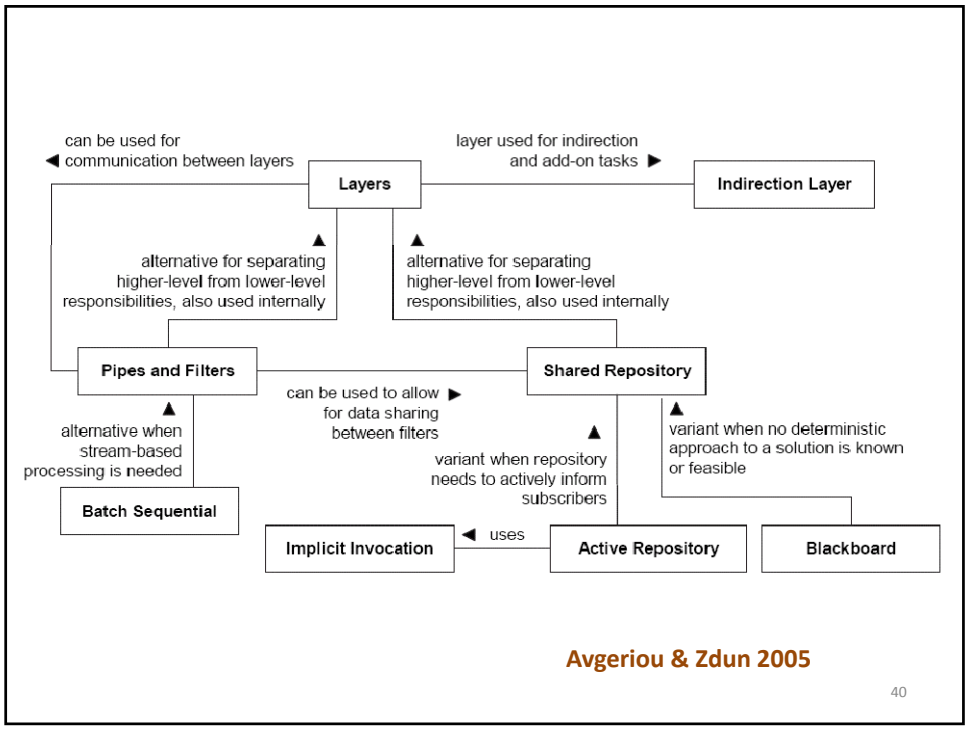
- モデル駆動設計,
- モデル駆動アーキテクチャ.

 36


Use-Case Diagram
Class Diagram
Statechart Diagram
UNIFIED MODELING LANGUAGE
Deployment Diagram
Collaboration Diagram
Component Diagram
Sequence Diagram
UML: 多くの種類の図があるが、常にアーキテクチャに適しているわけではない
Copyright © 2009 by KESL
37


パターン
良く起こる問題への一般的な解決策…
アーキテクチャパターン:
Buschmann F., Meunier R., Rohnert H Sommerlad P. & Stal M. (1996). Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons
Volumes 1 to 5, 1996 - 2007
Knowledge Source
updateBlackboard()
execCondition()
execAction()
+operates
1..+activate
1..*
Blackboard
solutions
controlData
inspect()
update()
Control2
loop()
nextSource()
Copyright © 2009 by KESL
38


Reference Model
user input
Dynamic Query Binding
creates widgets
Factory Method
Decorator
user input controls
dynamically updates
created with
created with
decorates parent table
navigable views
Strategy
data storage
Data Column
extend schema
Cascaded Table
Interpreter
create derived columns
access using
implement using
implement using
Camera
instance of
Relational Graph
Expression
access using
operates on
draw items
add operations
Proxy Tuple
instance of
Facade
Visitor
filter operations
processes
Renderer
assign renderer
Production Rule
assign values or perform delegation
Operator
scheduling and recurrence
Scheduler
reuse renderers
utilizes
facilitate extension
implement activities
Flyweight
Chain of Responsibility
Template Method
animated view transitions
39


can be used for communication between layers
layer used for indirection and add-on tasks
Layers
Indirection Layer
alternative for separating higher-level from lower-level responsibilities, also used internally
alternative for separating higher-level from lower-level responsibilities, also used internally
Pipes and Filters
Shared Repository
can be used to allow for data sharing between filters
alternative when stream-based processing is needed
variant when repository needs to actively inform subscribers
variant when no deterministic approach to a solution is known or feasible
Batch Sequential
Implicit Invocation
uses
Active Repository
Blackboard
Avgeriou & Zdun 2005
40

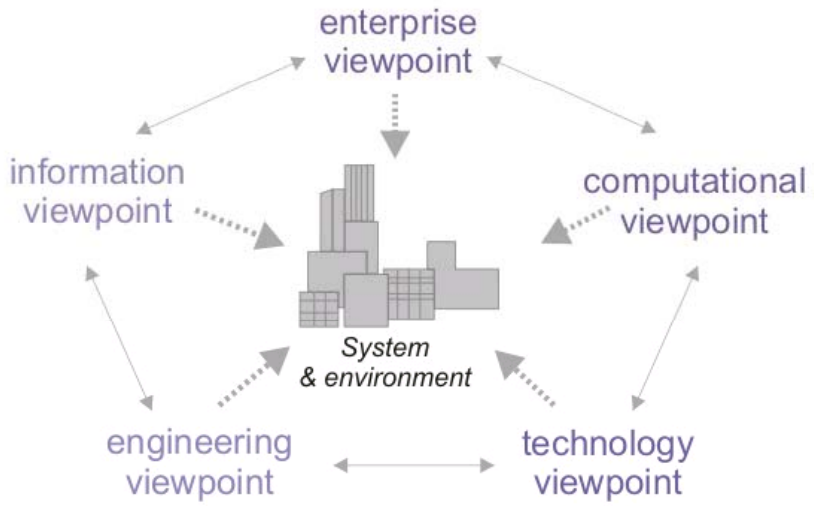
標準, 参照アーキテクチャ

- 汎用的な知識を集積
- IEEE 1471, ISO 42010 architecture representation

- RM-ODP = ISO 10746
- TOGAF (The Open Group)
- MoDAF, DoDAF, <xyy>AF
- ISO 19439 エンタープライズモデリングのフレームワーク

 41

RM-ODP or ISO/IEC 10746



42

手法

- ADD, ATAM, QAW (SEI)
- RUP (IBM)
- SAV,… (Siemens)
- BAPO/CAFR (Philips)
- Etc.

- Software Architecture Review and Assessment (SARA) handbook

43

メタファー

- メタファーは形式に意味を与え, 概念的なシステムの土台を与える.
- 認識の変換: ソースドメインからターゲットドメインへの
  - <ターゲット> は、<ソース>である

Lakoff and Johnson (1980) *Metaphors we live by*

 44

メタファー

- オントロジー的なメタファー:
  - クライアント/サーバ, レイヤー, パイプとフィルター, ショッピングカート
- 構造的なメタファー
  - 空間的: 最上位に, 並列に, 並んで, フォアグラウンド/バックグラウンド
  - ネットワーク, ウェブ, 階層
  - コンテナー: パッケージ, リポジトリー, ライブラリ, ボリューム…

 45

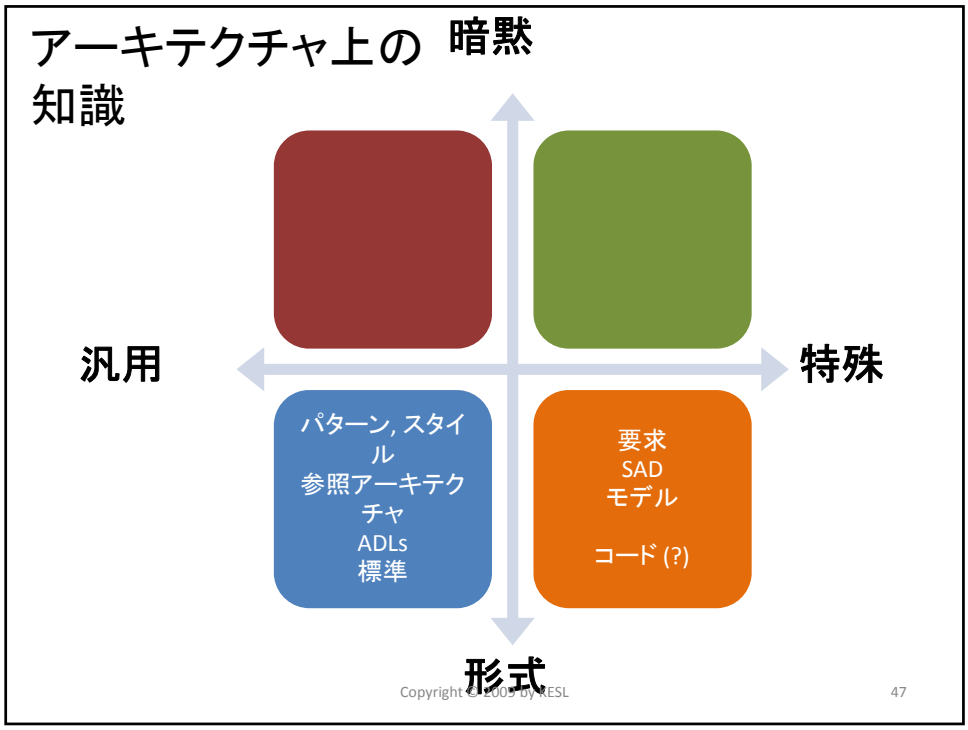
メタファーを超えて: Blends

- 2つのソース空間を使う“スーパーメタファー”
- デスクトップメタファーは実はblendである
  - コンピューターコマンド
  - オフィス要素

Imaz & Benyon 2007

“Good news.
The test results show it’s a metaphor.”

46



足りないもの

- ソフトウェアアーキテクチャ説明書
- 知識を移転する
  - 他のシステムへ
  - 他の人へ
  - 他の組織へ
- 根拠: なぜ?
- 判断.... ?

 48

ソフトウェアアーキテクチャ

ソフトウェアアーキテクチャは、以下のような**大きな設計判断**を含んでいる

- ソフトウェアシステムの**組織**,
- 構造的要素の選択と(それらの要素の間のコラボレーションとして仕様化される振る舞いから作られる)システムに対するそれらの要素のインタフェース,
- それらの要素から段階的により大きなサブシステムの作成,

Grady Booch, Philippe Kruchten, Rich Reitman, Kurt Bittner; Rational, circa 1995
(derived from Mary Shaw)

AK = AD + DD

アーキテクチャ上の知識

=

アーキテクチャ設計

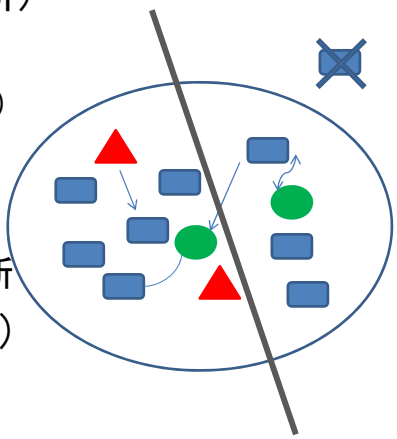
+

設計判断

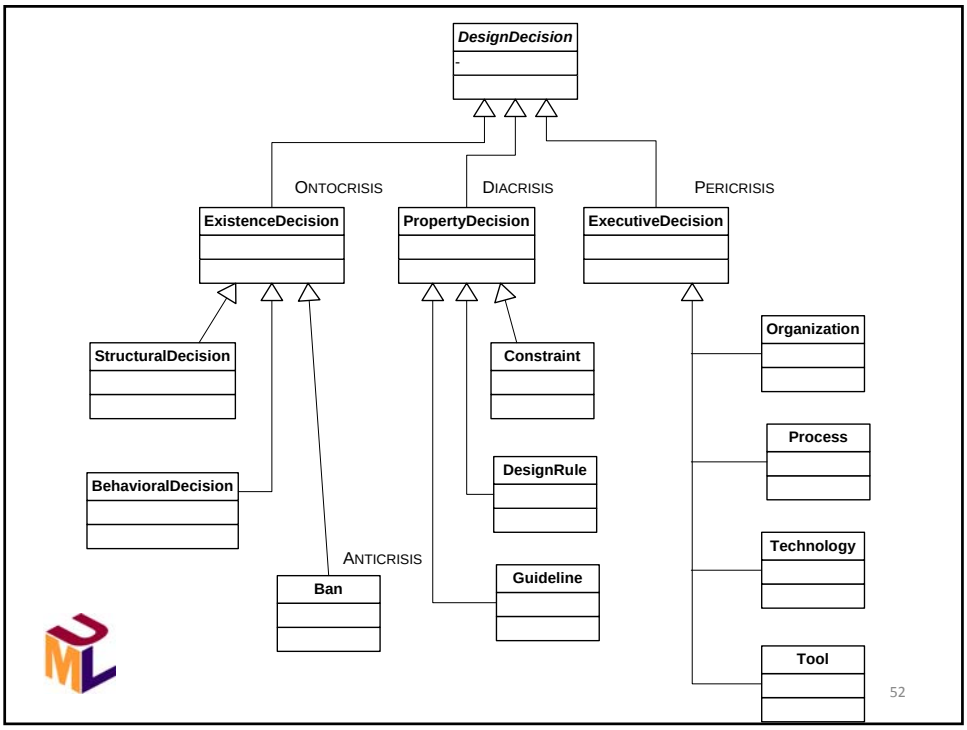
50

判断の種類

- Ontocrises（存在の判断）
  - 存在するかの判断
  - Anticrises（反対の判断）
- Diacrises（横断的判断）
  - プロパティに関する判断
- Pericrises（周囲の判断）
  - 重役の判断



 51



52

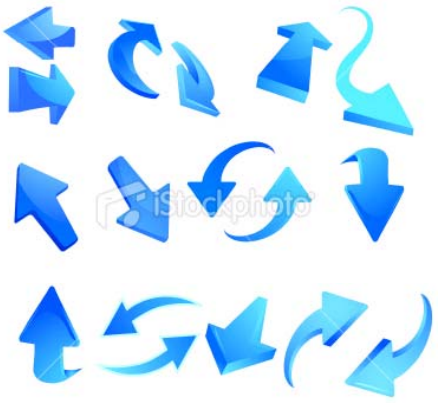
判断の属性

- 典型 テキスト
- 根拠 テキスト or 参照
- スコープ テキスト
- 状態 列挙
- 履歴 (タイムスタンプ + 執筆者 + 変更) のリスト
- コスト 価値
- リスク 露出レベル

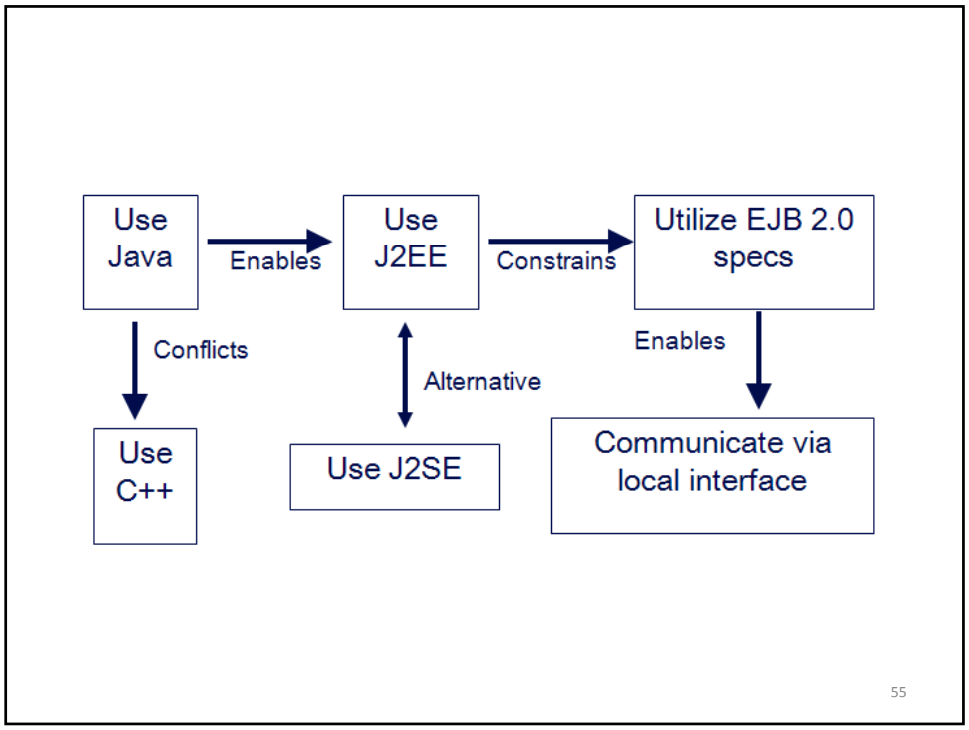
53

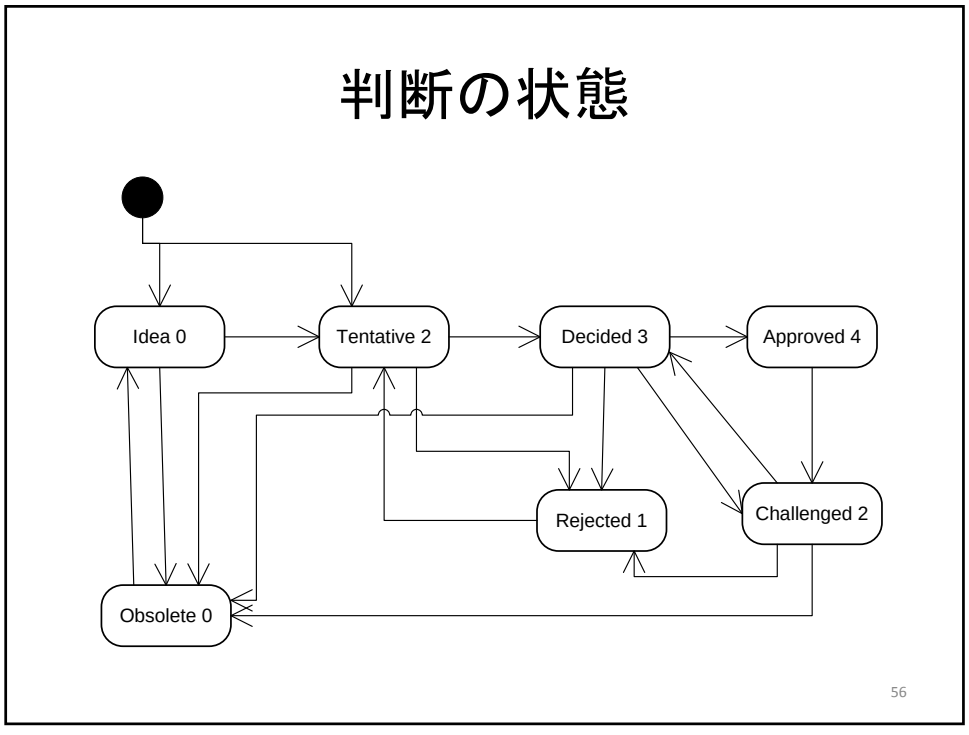
判断の間の関係

- 制約する
- 禁ずる
- 可能にする
- 組み込む
- 衝突する
- 覆す(Override)
- でできている
- に向かっている
- の代りとなる
- に関係する
- 辿れる
- を順守しない

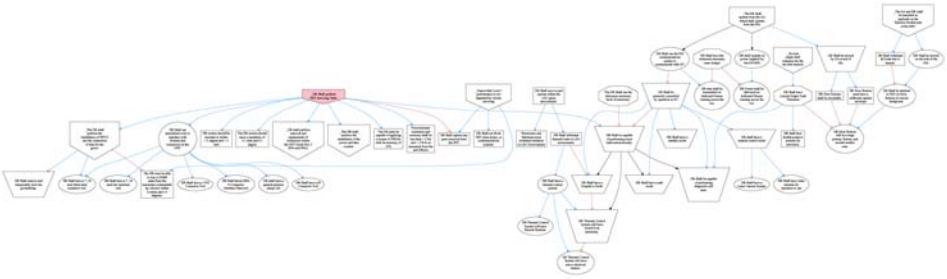


54


Use Java
Enables
Use J2EE
Constrains
Utilize EJB 2.0 specs
Conflicts
Alternative
Enables
Use C++
Use J2SE
Communicate via local interface
55


判断の状態
Idea 0
Tentative 2
Decided 3
Approved 4
Rejected 1
Challenged 2
Obsolete 0
56

実験: Spar Aerospace Dexterous Robot

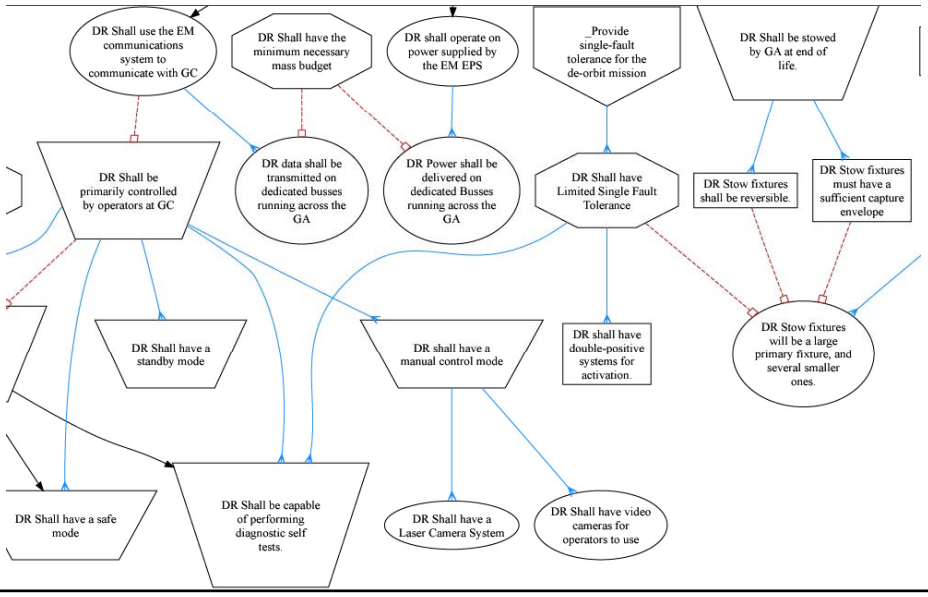


- DR = Dexterous Robot = Hubble望遠鏡を修理するためにアームが使われた
- MySQL Database, SVG + GraphViz

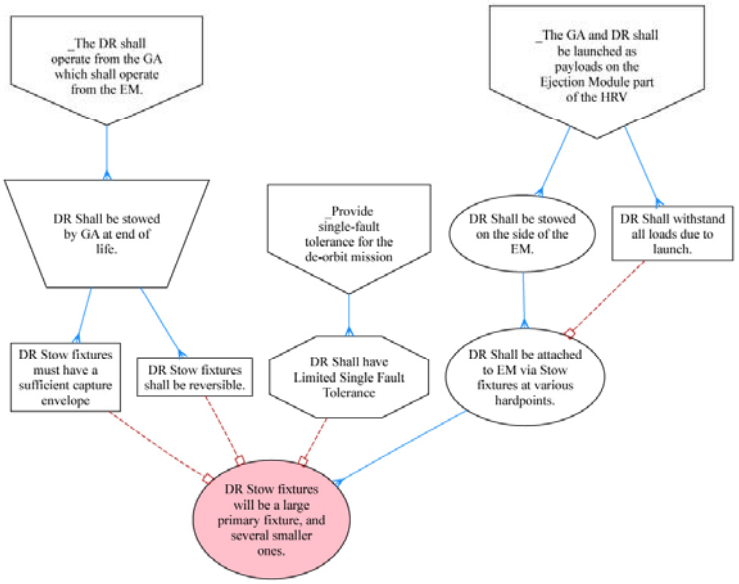
Michael Trauttmansdorff & Nicolas Kruchten

57

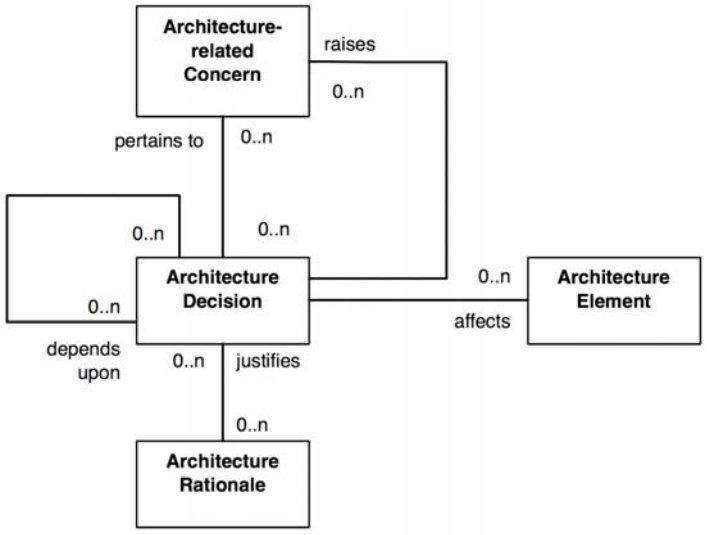
拡大された一部



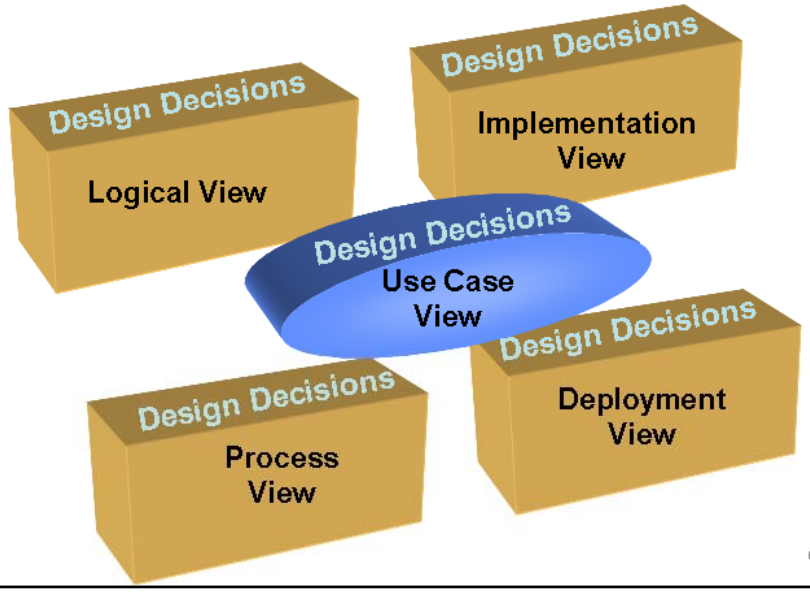
抽出, around Stow Fixture



ISO 42010に追加されたもの:



5+1 ビュー?



62

設計判断を捕捉する

- “設計根拠サポートシステムは、捕捉するためのオーバーヘッド故にソフトウェア業界からのいかなるレベルのサポートも得られなかった”(Jintae Lee)
  - QOC, DRL, InfoRAT, IBIS その他.
  - すぐに十分な価値が得られないので、捕捉する動機が生まれない
  - 退屈なプロセス, 静的な図

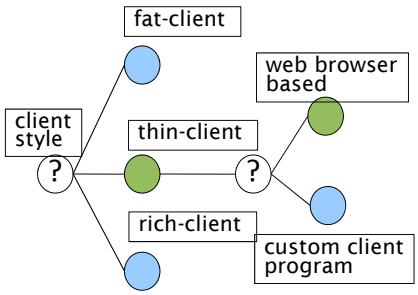
63

Aside: 判断支援, 判断プロセス

- これらのシステムの多くは、合理的な判断プロセスを支援するように設計されていた
  - 問題を確立する
  - 選択肢を列挙する, 長所と短所を探す
  - ランク付け, 優先度付け
  - 選択
  - 選択理由の文書化
- 実際には: 直感的であり, アイデアが先に思い浮かぶ
  - そして理屈付け, あるいは破棄や再作業

64

合理的な判断



基準:
ポータビリティ?

65

捕捉に異なる方法で挑む…?

- デーモン(エージェント)で捕捉を自動化する
  - 判断の元に仕込む:
    - 設計ツール
    - 要求管理ツール
    - 欠陥追跡ツール
    - 構成及び変更管理ツール
    - マネジメントツール(タスク割り当て, 問題点/アクション項目)
- Waypointing
- 今捕捉し, 後で整理する

66

アーキテクチャ知識マネジメントのためのツール

- 集積またはパーソナリゼーションまたは両方?
- “中央リポジトリ”の神話
  - 官僚的な学校
- 新しい問題を解決する追加のツールの神話
  - ツールベンダー と/あるいは 卒業生
- “けものを飼う(＝金をつぎ込むことが目的になる)”
- 時間的ズレ: 製造 – ニーズ
  - 動機なし, ‘粘り’なし

67

ツール

- 誰のための?
  - アーキテクト
  - レビュー者, 監査者
  - 要求エンジニア., 分析
  - 保守担当者
- 何を行うため?
  - AKの作成
  - AKの取得
  - アーキテクチャ評価
  - 判断を下す
  - 他者を教育する
  - ....

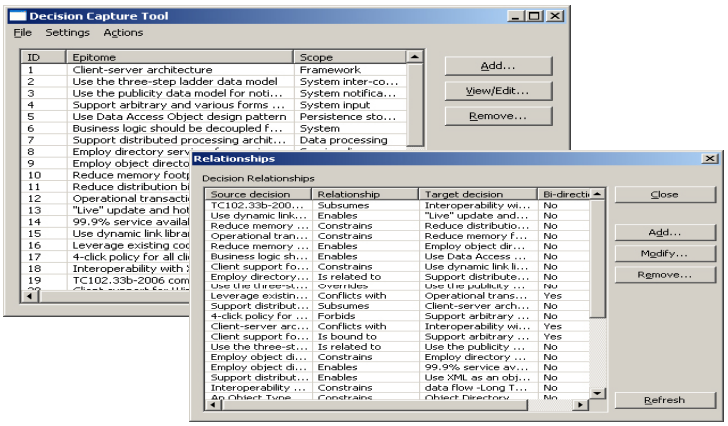
68

ツール

- ADDS: Rafael Capila
- Archium: Jansen & Bosch
- AREL: Tony Tang
- Knowledge architect
- IBM's Architect's Workbench
- SEURAT: Burge

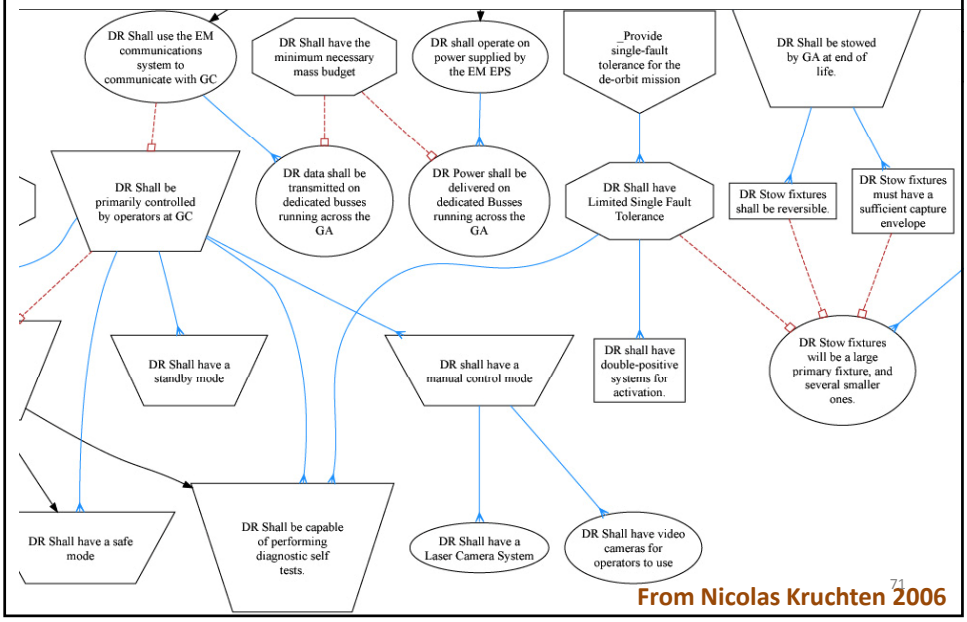
69

表形式



70

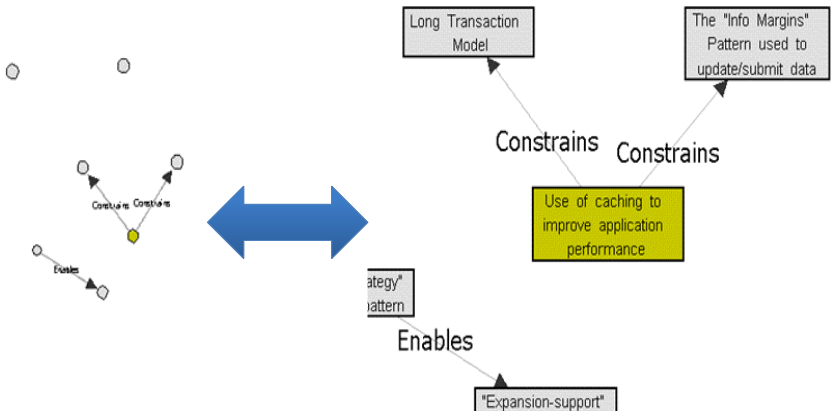
図形式



71

From Nicolas Kruchten 2006

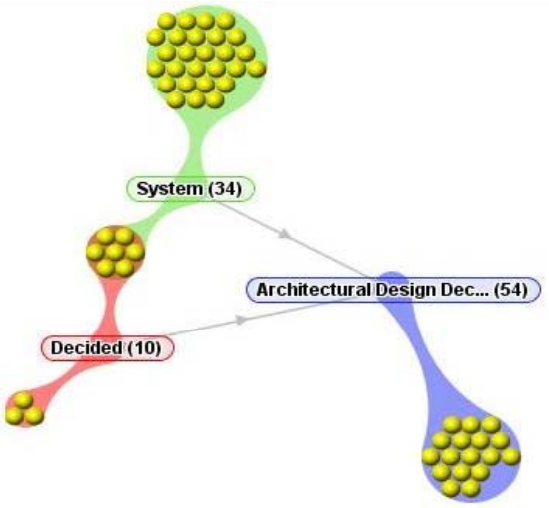
展開と省略



Lee & Kruchten 2008

72

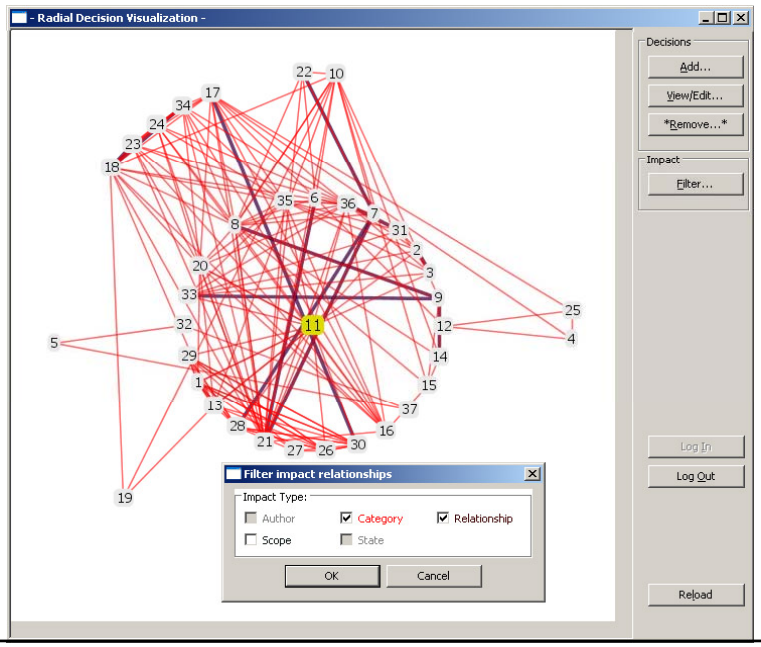
クラスタリング (with Aduna)



Kruchten et al 2005

73

What if分析



74

パーソナリゼーション (& ハイブリッド)

- 知識共有ネットワーク
- WIKIs
- Web 2.0

- セマンティクweb, セマンティクwikis…
- EAGLE at VU
- PAKME

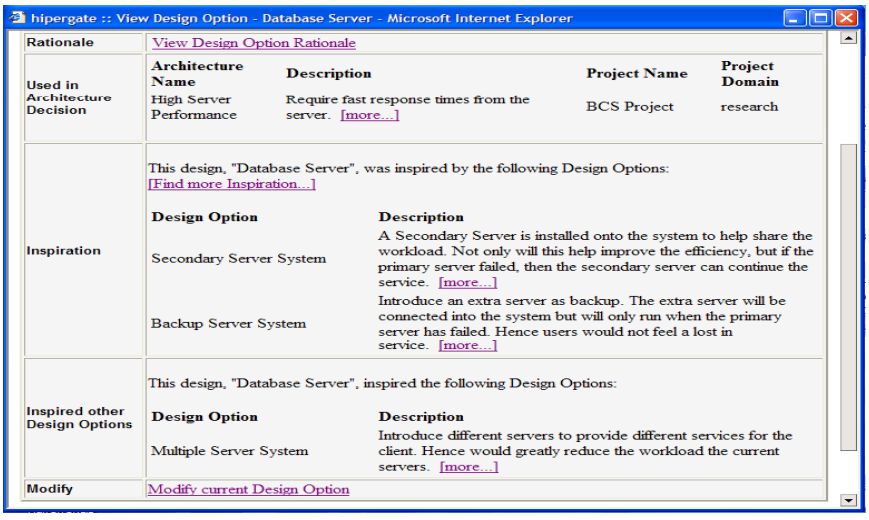
75

PAKME

- M. Ali Babar, LERO, Limerick Ireland

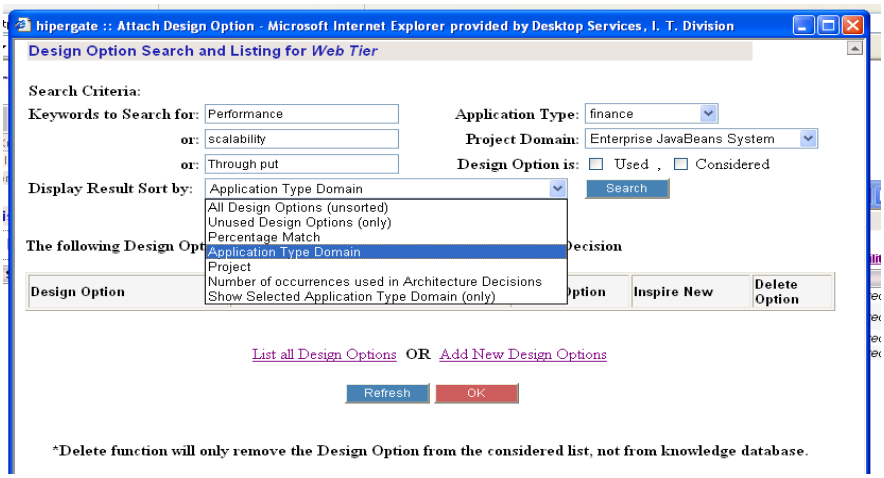
76

設計判断をケースとして捉える



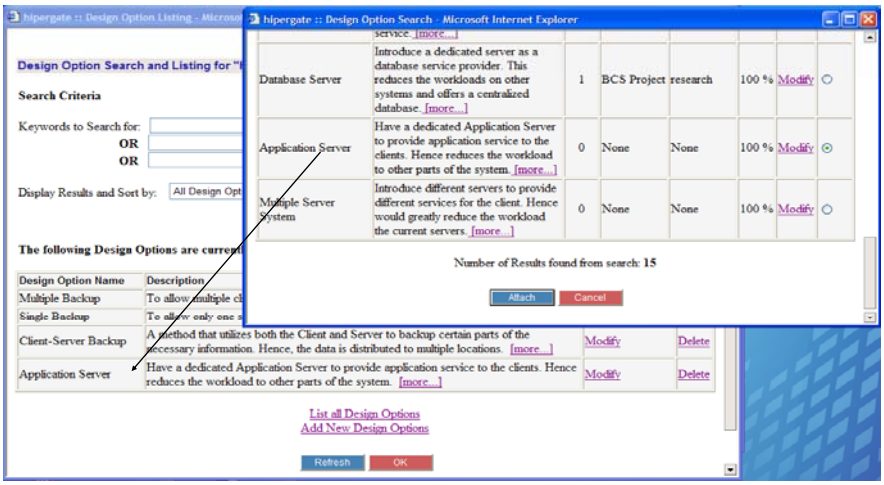
77

設計判断ケースを検索する



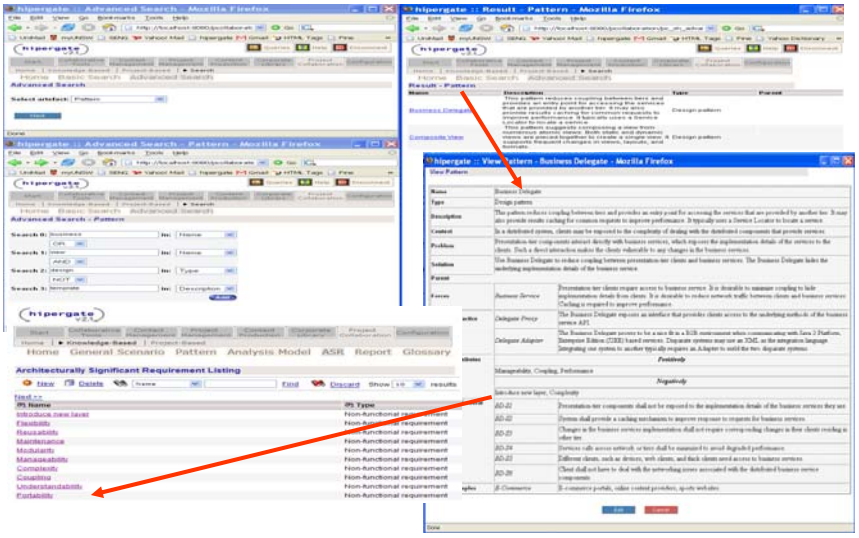
78

設計判断ケースを使う



79

知識ベースをナビゲートする



80

パターンを補足、表現するための テンプレート

hipergate :: View Pattern - Business Delegate - Microsoft Internet Explorer provided b...

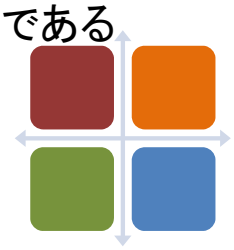
View Pattern

Name	Business Delegate
Type	Design pattern
Description	This pattern reduces coupling between tiers and provides an entry point for accessing the services that are provided by another tier. It may also provide results caching for common requests to improve performance. It typically uses a Service Locator to locate a service.
Context	In a distributed system, clients may be exposed to the complexity of dealing with the distributed components that provide services.
Problem	Presentation-tier components interact directly with business services, which exposes the implementation details of the services to the clients. Such a direct interaction makes the clients vulnerable to any changes in the business services.
Solution	Use Business Delegate to reduce coupling between presentation-tier clients and business services. The Business Delegate hides the underlying implementation details of the business service.
Parent	*No Parent Available*
Forces	1) Business Service
Tactics	1) Delegate Proxy 2) Delegate Adapter
Affected Attributes	***Positively***
	1) Performance
	Negatively
	1) Complexity 2) Introduce new layer
General Scenario	1) BD-S6 2) BD-S2
Usage Examples	1) E-Commerce

81

まとめ

- アーキテクチャは、結果として得られたアーキテクティングの設計以上のものである
- 暗黙, 形式知
- 汎用, 特殊な知識
- 集積, パーソナリゼーション
- メタファーの威力
- 一級の市民としての判断
- ツールのサポート(さらに改良が必要)



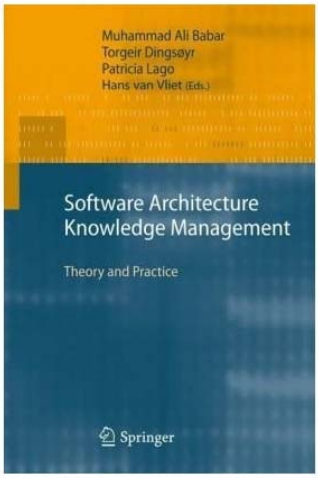
83

Slides at: pkruchten.wordpress.com/talks/

質問?

84

恥知らずの自己宣伝



M. Ali Babar, T. Dingsøyr, P. Lago, H. van Vliet, *Software Architecture Knowledge Management*, Springer Verlag, 2009

3 章を書きました!

85

References (1)

- Avgeriou, P., & Zdun, U. (2005). Architectural patterns revisited:a pattern language. Paper presented at the 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), Irsee, Germany.
- Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice (2nd ed.).* Reading, MA: Addison-Wesley.
- Buschmann F., Meunier R., Rohnert H. & Sommerlad P. & Stal M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns.* John Wiley & Sons.
- Gladwell, M. (2005). Blink: The Power of Thinking Without Thinking. New York: Little, Brown and Company.
- Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., & America, P. (2007). A General Model of Software Architecture Design derived from Five Industrial Approaches. *Journal of Systems & Software,* 80(1), 106-126.
- Imaz, M., & Benyon, D. (2007). *Designing with blends: conceptual foundations of human-computer interaction and software engineering.* Cambridge, MA: The MIT Press.
- Kruchten, P., Capilla, R., & Dueñas, J. C. (2009). The role of a decisions view in software architecture practice. *IEEE Software*, 26(2).

86

References (2)

- Kruchten, P. (1995). The 4+1 View Model of Architecture. *IEEE Software*, 12(6), 45-50.
- Kruchten, P. (2004, December 3-4). An Ontology of Architectural Design Decisions. Paper presented at the 2nd Groningen Workshop on Software Variability Management, Groningen, NL.
- Kruchten, P. (2009). Documentation of Software Architecture from a Knowledge Management Perspective--Design representation. In: M. Ali Babar, T. Dingsøyr, P. Lago & H. van Vliet (Eds.), *Software Architecture Knowledge Management: Theory and Practice* (pp. 29-58). Berlin: Springer-Verlag.
- Lakoff, G., & Johnson, M. (1980). *Metaphors we live by.* Chicago: The University of Chicago Press.

87

References (3)

- Nonaka, I., & Takeuchi, H. (1995). *The knowledge creating company: how Japanese companies create the dynamics of innovation*. New York: Oxford University Press.
- Obbink, H., Kruchten, P., Kozaczynski, W., Hilliard, R., Ran, A., Postema, H., et al. (2002). *Report on Software Architecture Review and Assessment (SARA)*, Version 1.0. At http://philippe.kruchten.com/architecture/SARAv1.pdf.
- Perry, D. E., & Wolf, A. L. (1992). Foundations for the Study of Software Architecture. *ACM Software Engineering Notes*, 17(4), 40-52.
- Rozanski, N., & Woods, E. (2005). *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives.* Boston: Addison-Wesley.
- Vitruvius Pollio, M. (25 B.C.). *De Architectura*.

88