HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

科学的モデリングのススメ

カーネギーメロン大学ソフトウエア工学研究所パートナー
HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.
橋本隆成

アジェンダ

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

はなしの流れ

- 製造業の現在と未来
- 製造業のソフトウエア開発の命題
- ソフトウエア開発の自動化
- ETロボコンの参加意義と活用

製造業の現在と未来
～今開発現場で何が起きているのか？～

日本に何が起きているのか？

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

経営的視点の直面する課題

- 国内GDPの大幅な減少
- 新興国の台頭
- 製品価格の下落
- 海外への市場のシフト

技術視点の直面する課題

- 人件費の抑制
- 品質の確保
- 生産性の向上
- 国際標準への対応

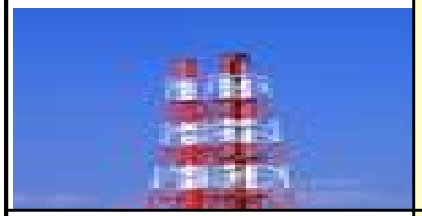
分野毎の国際標準規格①

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

分野		規格
	鉄道/輸送	• EN 50128--Railway applications Software for railway control & protection systems
	航空	• RTCA DO-178B—Software Considerations in airborne and equipment certification requirements
	医療/ 医療関連	• FDA—General principles of software validation
	軍事/防衛	• DEF STAN 00-55(Part2)—Requirements for safety related software in defense equipment • MIL STD 498—Software Development and Documentation

分野毎の国際標準規格②

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

分野		規格
	自動車	• Automotive SPICE • ISO26262
	原子発電	• IEC 60880—Software for computers important to safety for nuclear power plants
	一般/共通	• SEI-CMMI for Development • PMI-PMBOK • IEEE 1008—Standard for software unit testing • IEEE 1012—Standard for verification and validation • IEEE 829—Standard for software test documentation • IEC 61508—Functional safety of electrical /electronic /programmable safety-related systems

国際標準で特に注意したいポイント①
形式手法

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

名称	分野
IEC61506　SIL4 (JIS C 0508)	• 電子機器の機能安全に関する国際規格 • 2000年に刊行された電気・電子関連の国際安全規格 • 日本では「機能安全規格」と呼ぶ • 欧州主導で策定 • 安全度水準をSIL(Safety Integrity Level)で4段階に区分 　• SIL2以上で形式手法を推奨,SIL4で適用
ISO/IEC15408 EAL5-7 JIS X 5070	• IT製品のセキュリテに関する国際標準 • 情報セキュリティの実現度をEAL(Evaluation Assurance Level：評価保証レベル)で表現 • 形式表現適用の規定 　• EAL5以上で「準形式的表現」／EAL7では「形式的表現」が必須

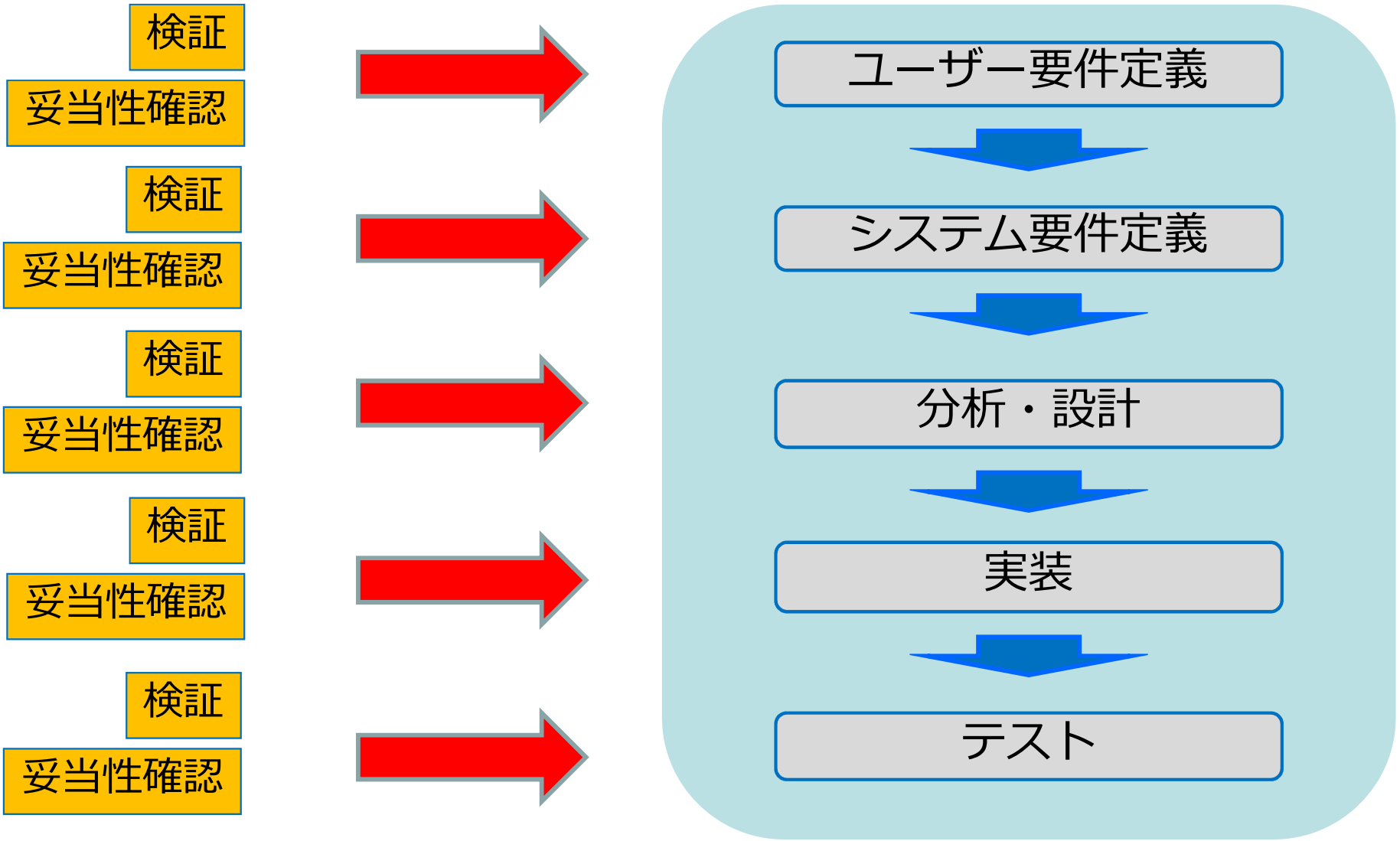
国際標準で特に注意したいポイント② 形式手法の例

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

タイプ	特徴	記述言語
モデル規範	仕様記述に論理学,集合理論を用いる	B,VDM,Z
性質規範型 (代数仕様)	仕様記述に代数学を用いる	OBJ
モデル検査	システムの振る舞いモデルを作成し,特定の性質を検査する手法	SMV,SPIN
定理証明	定理証明を目的として,検証を完全に行うことを特徴とする	HOL
プロセス代数	並行プロセスの振る舞いを代数的に記述する手法	CSP,CCS
同期型言語	クロックに同期する記述言語	Esterel,Lustre
時相理論	時間を取り扱う論理の記法	CTL,LTL
簡易形式手	バグ出しを目的とする手法	ESC

国際標準で特に注意したいポイント③
上流からの品質確保

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



21世紀のソフトウエア開発の命題

～これからのソフトウエア開発を先取りする～

21世紀のソフトウエア開発の命題

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

科学的(数学的・論理学的・工学的)である

自動化が可能である

メトリクスによる客観的な証拠

国際標準に準拠した開発プロセスである

モデルに要求されること

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

モデルが数学的・論理学的である

モデルの品質をツールにより検証できる

モデルの品質を数値化し制御できる

オブジェクト指向の起源は？

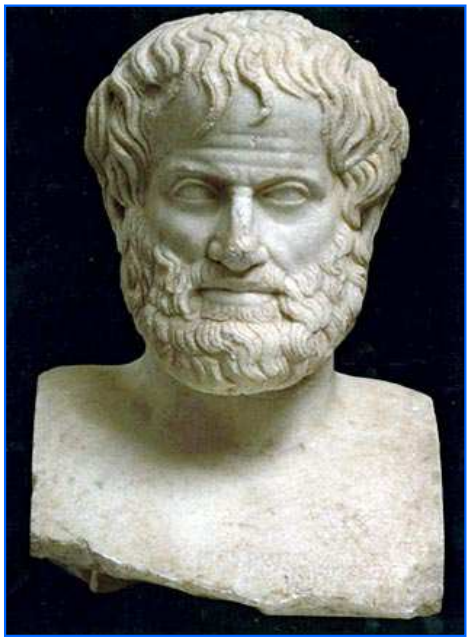
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

オブジェクトとクラスの起源

- オブジェクトやクラスによる考え方は何が起源か？
- 科学的(数学的・論理学的)とはどういうことか？

Who am I ?

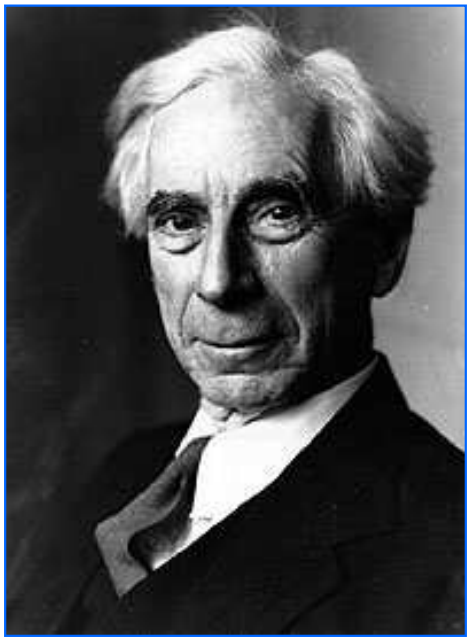
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.



アリストテレス

ライプニッツ

カント

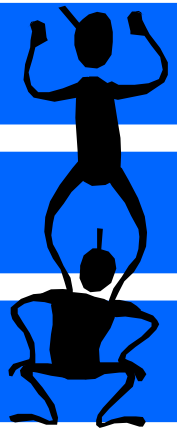


ラッセル

存在論・論理学・数学が土台

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

オブジェクト指向開発アプローチ



存在論(形而上学的)

述語論理学

集合・写像(群論・圏論)

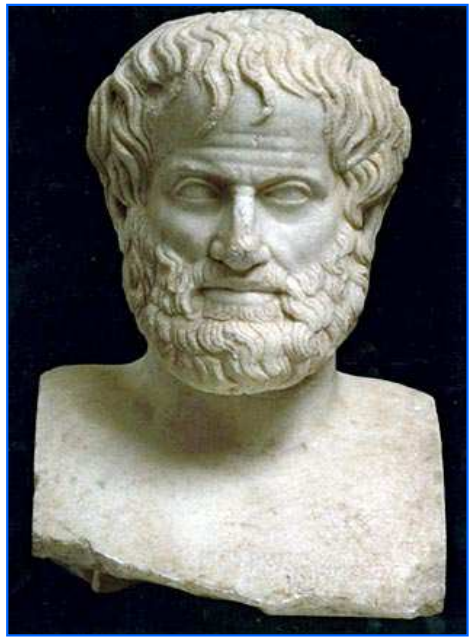
過去の知的資産を利用できる

数学・論理学の資産(考え方や定理)を活用できる

開発環境を活用でき自動化が可能となる

存在論って何？

- 人間,岩,星,犬はなぜ存在するのだろうか？
- 存在の理由は何か？
- 私たちの思考の対象は何か？
- なぜ全く同じ実体でも区別がつくのか？



哲学者の概念の分類カテゴリー
～オブジェクト指向の基礎となっている

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

哲学者	カテゴリー(範疇)
アリストテレス	• 実体／量／能動／受動／所属(所有)／状態／場所／時間／関係 • アリストテレスの「**存在論**」からの定義
カント	• 量／質／関係／様相 • さらに細かな分類がある • カントの「**認識論**」からの定義
ヴント	• 対象／属性／状態／関係

概念はいろいろな立場から分類されるが,分類する
立場の相違によって,異なった種類の概念が存在する

オブジェクト指向開発方法論の比較

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

方法論	SM法	Ross	Coda／Yourdon	アリストテレス
オブジェクト／クラスの候補	• 物理 • シミュレーション • 仕様 • 出来事 • 相互作用 • 役割	• 人 • 場所 • 物 • 組織 • 概念 • 事象	• 構造 • 他のシステム • 装置 • 記憶されるものや事象 • 演じられる役割 • 操作手順 • 場所 • 組織単位	• 実体 • 量 • 能動／受動 • 所属(所有) • 状態 • 場所 • 時間 • 関係

古典的(哲学的)なカテゴリーを基礎に置く方法と呼ばれる

論理学の用語

用語	説明
判明	• 概念(集合,クラス)の内包を明らかにすることを「判明」するという

定義	• 内包を判明する時の論理的手続き「定義」と呼ぶ
	  • 定義は種類が存在する
	  • 実質的定義／発生的定義／唯名定義／指示的定義／操作的定義

明晰	• 概念の外延を明らかにすること「明晰(めいせき)」にするという
名辞	• 概念が言葉で表現されたもの
主語	• 主張(説明)される対象
述語	• 主語について主張(説明)される事柄
周延／不周延	• 名辞(主語または述語)の外延全部に命題が論及する場合は周延,否の場合は不周延

概念間の関係種類①

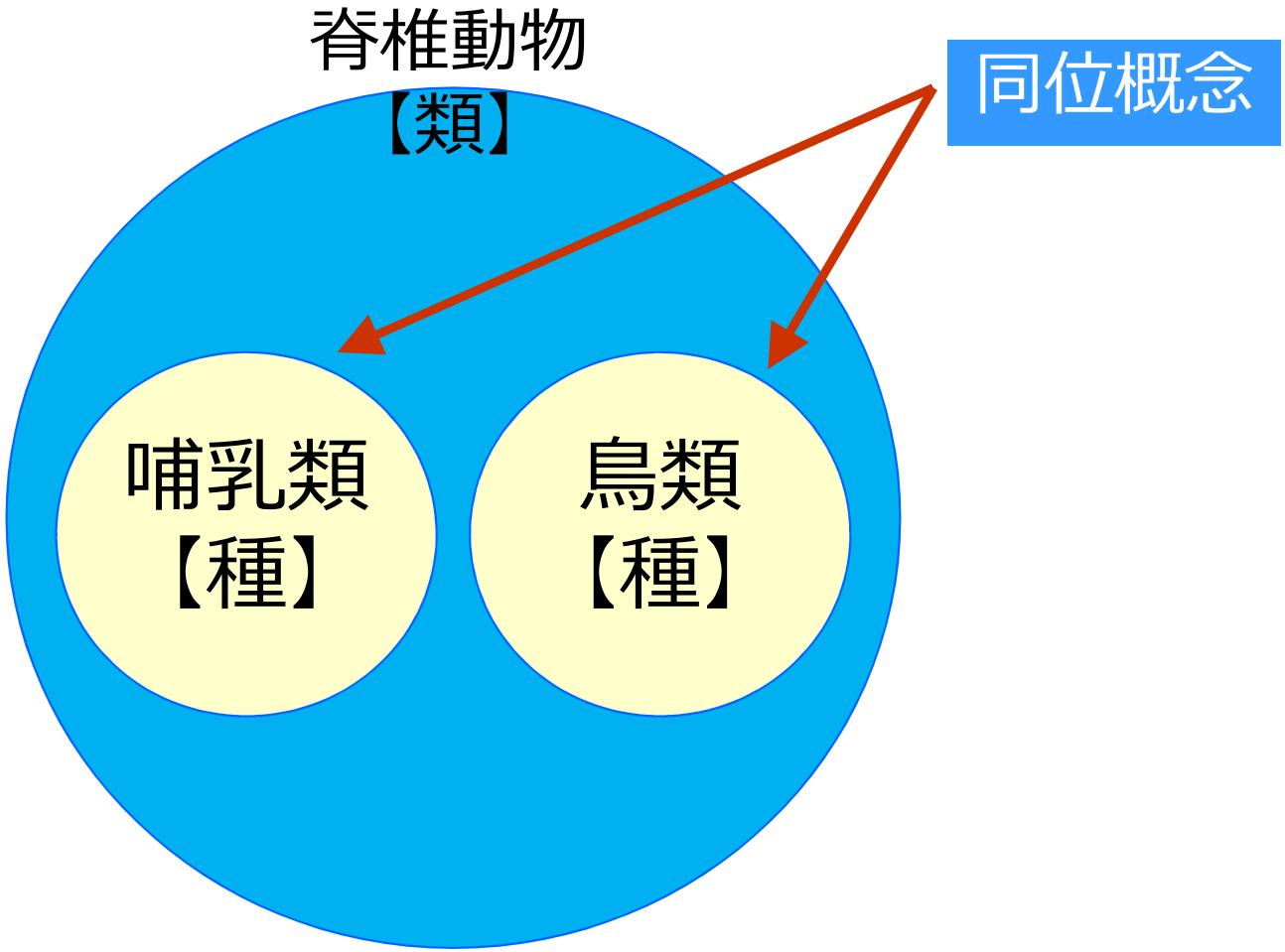
HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	説明
同一概念	• 内包,外延ともに同じ概念を同一概念という

等値概念	• 内包の違う二つの概念が同じ対象を示しているか,またはその外延が等しい場合,その２つの概念を等値概念という
	  • 「二等辺三角形と二等角三角形」
	  • 「総理大臣と○○党代表」など
	  • 一般に「ＡはＢであり且つＢはＡである」という命題が真であるときＡとＢは等値概念である

上位概念 (類概念)	• 包含関節がなりたっている時,他の集合を内に包んでいる方の概念を上位概念または類概念という
下位概念 (種概念)	• 他の集合の中に包まれている方の概念を下位概念または種概念という

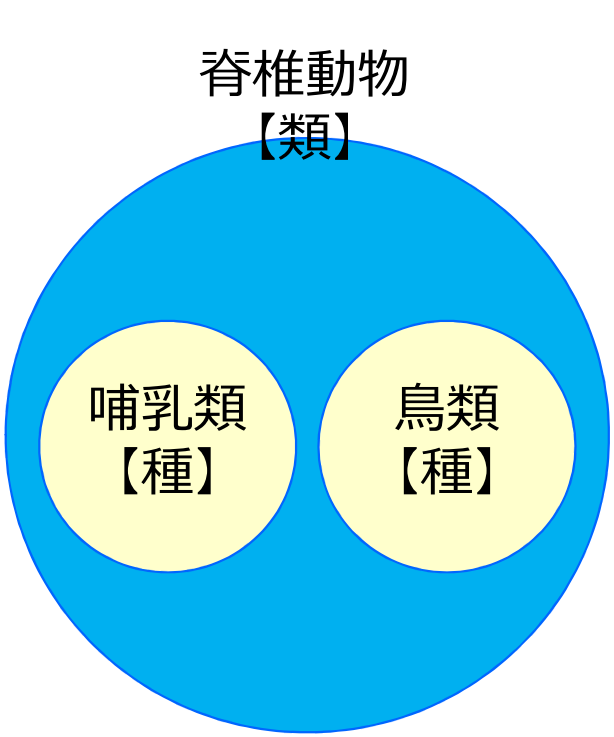
集合の包含関係と類・種①



同一の類概念内に種概念があるとき互に同位概念

集合の包含関係と類・種②

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.



用語	意味

限定	• 概念の内包に新しい内包をつけ加えて外延を縮小することを「**限定**」
	  • 概念を限定することにより対象の本質が精緻になる
	  • UML用語の「**特化**」に該当

概括	• 内包を減らして外延を拡げることを「**概括**」という
	  • 概括することにより細分化されていた知識が統一できる
	  • UML用語の「**汎化**」に該当

概念とその種類を論理学的に理解することは
正確なモデリングと分析にとても重要!!

概念間の関係種類②

用語	一つの類概念に含まれる種概念の集合の包含関係の種類
選言概念	• 同一類概念に属するいくつかの種概念の集合に共通の元がないとき,それらの集合に対応する概念を選言概念または熔接概念という
交叉概念	• 外延の一部に共通な部分があるものを交叉概念という

反対概念	• 大と小,白と黒などのように同一の量或いは性質において殻大の相見を示す二つの概念を反対概念という
	• 反対概念は次の矛盾概念と追って,その中間に第３の概念を入れることができる
	  • 例：「中」や「灰色」
	  • 反対概念は同時に肯定することはできないが,同時に否定することができる

概念間の関係種類③

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	一つの類概念に含まれる種概念の集合の包含関係の種類
矛盾概念	• 同じ類概念に属していて一方が他方の否定となっている概念を矛盾概念という • 類概念の集合の中に一つの種概念の集合を取ったとき,その補集合を示す概念はこの種概念の矛盾概念である
絶対概念	• どんな事物も他の事物と何か関係を有するから,一般に概念は本来他の概念と何らかの関係をもっている • ところが,一応他の概念とは無関係で独立して一つの意味を有すると考えられる概念を絶対概念という
相対概念	• 相対概念は昼と夜,兄と妹などのように一つむ関係原理に基いて互いに相対的な関係を他との関係を考えることによって,概念の意味がよく理解される

概念の種類と特徴を理解して「概念モデル」を作成すると非常に論理的かつ優れたモデルになるよ

概念も視点によって色々分類されている

概念間の関係種類④

用語	説明
相関概念	• 相対概念の中で特に相対関係が深いもの • 二つの相対概念ＡとＢとがあってＡからＢを,またＢからＡを理解して,初めてそれ白身の意味がはっきりするとき両者をそれぞれ他に対して相関概念という 　• 例：上と下,親と子,右と左,夫と妻などは互いに相関的である
単独概念	• 固有名詞が示す概念で,その集合に唯一つの元しか存在しない概念を単独概念または個体概念という • 単独概念は固有名詞だけでなく「世界最大の都市」とか「この人」などのような特定の個物を示す概念も単独概念となる
一般概念	• 大や猫などのように普通名詞の示す概念を一般概念とか普遍概念またはクラス概念という
集合概念	• 集合の元が集って一つの団体を組織し,これを全体として一括して初めて意味を有する概念をいう 　• 例：組合,読売ジャイアンツなど

概念間の関係種類⑤

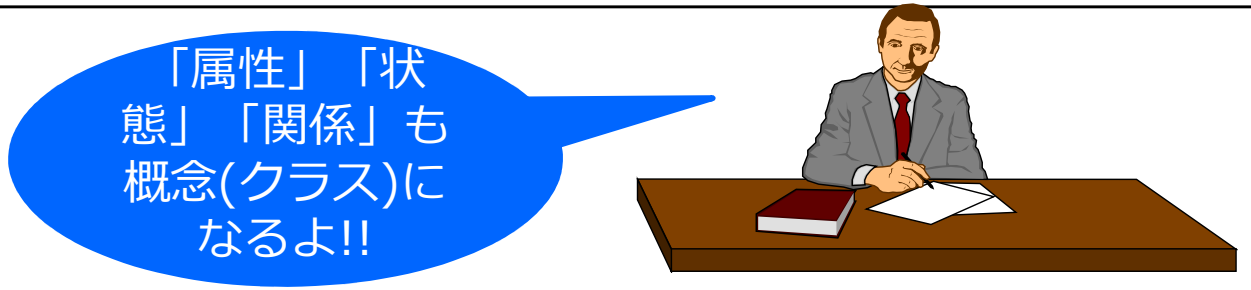
用語	説明
単純概念	• 概念の内包が最小となりこれ以上分析することができない概念を単純概念という • 例：「存在」「質」「量」などであってこれが「範疇(カテゴリー)」である
複合概念	• 色々な本質を持ち,内包の分析が可能である概念を複合概念という • 人や動物など日常用いられる概念は殆んどすべて複合概念である

隔離概念	• 二つの概念の間に類と種の包含関係も,また同位概念間に成立するいかなる関係も成立しない,全く内包間に等しいものが認められない概念を離隔概念または乖離概念という
	  • 例：青と正直,人間と風,道徳と三角形など全く比較することのできない概念
	  • 「月とすっぽん」「提灯と釣鐘」などもそれに近い概念

概念間の関係種類⑥

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

用語	説明
対象概念	• 判断において主語となる概念である • 人間性とか白さなど実体の属性を表すわす形容詞から転じた抽象名詞の類も対象概念に含まれる
属性概念	• アリストテレスの量および質に相当し,文法的には形容詞およびこれから転じた副詞の類を含む • 例：「白い」「冷たい」「10m」など
状態概念	• be,haveをはじめとするすべての動詞類を含み,対象のあらゆる存在と状態を示す • 属性と状態は別種の存在でなく事物の見方の相異によるものである
関係概念	• 対象相互の関係を表わす概念である • 関係の種類は「時間」「空間」「因果」「目的」および「論理的関係」の五つがある

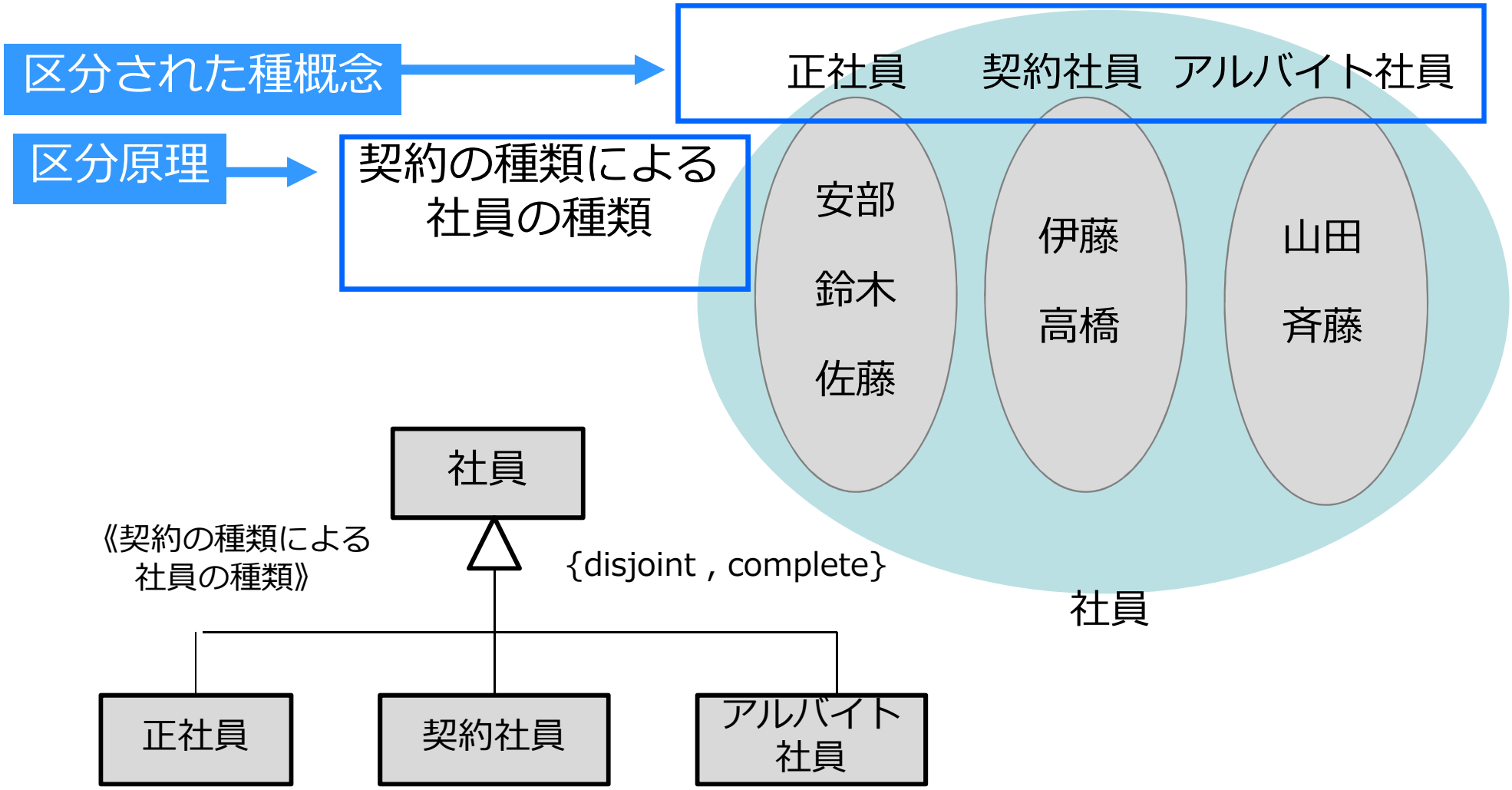


区分

項目	説明
区分	• ある類概念を(その概念が)包括している種概念分けること
区分原理	• 種へと分ける基準
区分肢	• 区分された種概念

区分の規則	• 採用する区分原理の選択は任意
	  • 区分を実施する人間の目的,関心によって自由
	  • 首尾一貫して区分原理は同一でなければならない
	  • 区分を実施するときに無差別に２つの区分原理が用いられていることを「交叉区分」
	  • 区分作業をする際に段階的に行わず,大きな飛躍があるとき「飛躍区分」

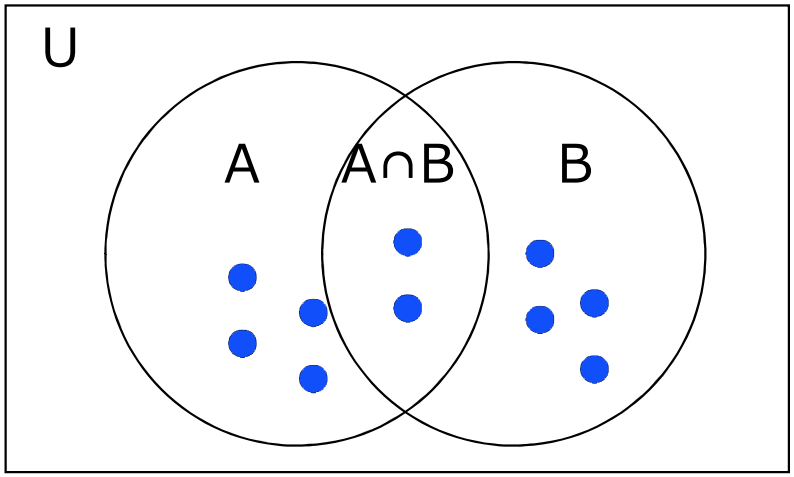
区分と区分原理



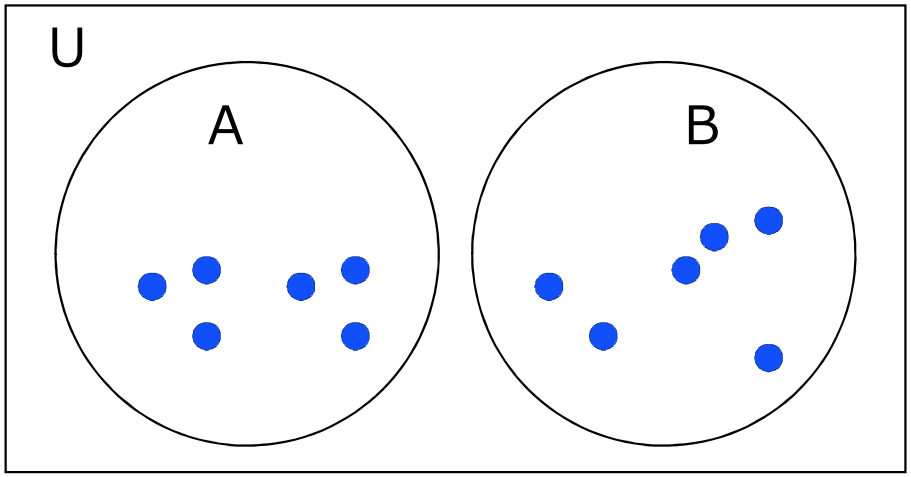
区分肢の交叉(交差)・同値

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

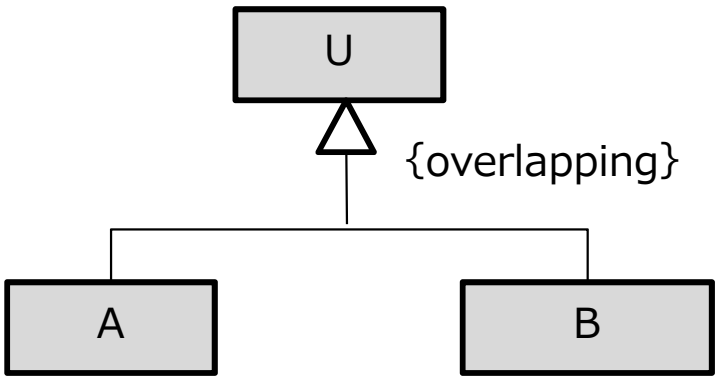
交叉分割(区分)

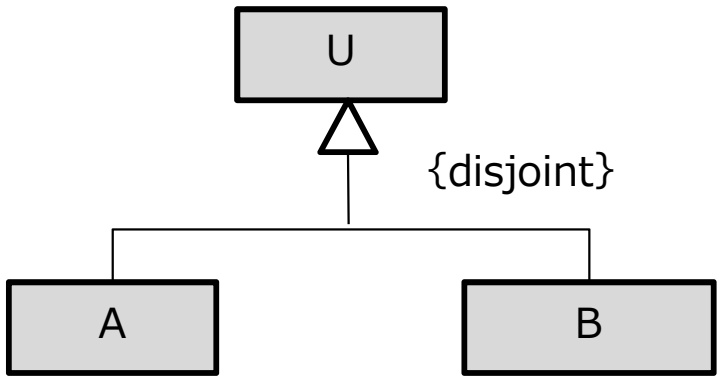


同値分割(区分)



UMLの制約で表現すると下記になる





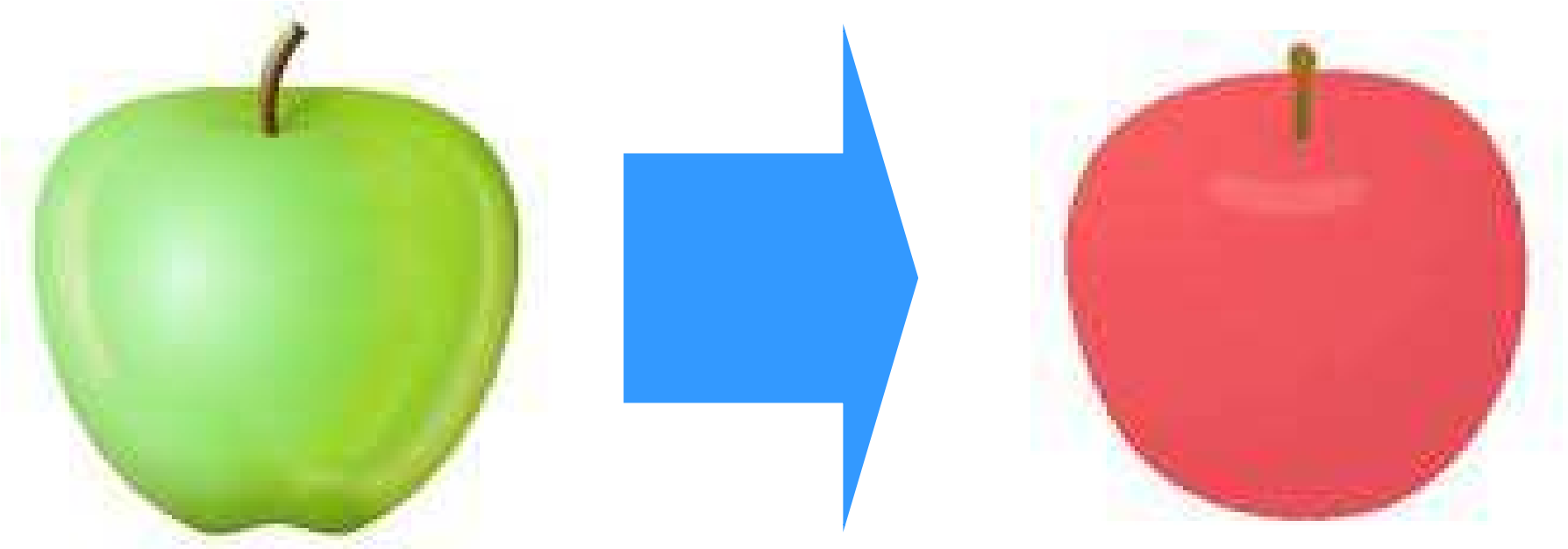
論理学の基本原理(公理)

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

定理	意味
同一律	• 「ＡはＡである」 • 「現象は変化するが,その根底にある実体そのもの『Ａ』は不変である」ことを意味すると考える
矛盾律	• 「Ａは非Ａでない」または「ＡはＢであると同時にＢでないということはあり得ない」 • Ａとその否定である非Ａとは相矛盾する事がらであって同時に成立することはあり得ないということを示す原理
排中律	• 「ＡはＢかまたは非Ｂである」 • Ｂと非Ｂという相矛盾する概念をとると,その中間にＢでもないが,非Ｂでもないという第３の概念は在り得ないし,また考えることも不可能

現代工学の原理となっている

同一律



時間が経過して赤く熟しても,甘みが増しても同じりんご

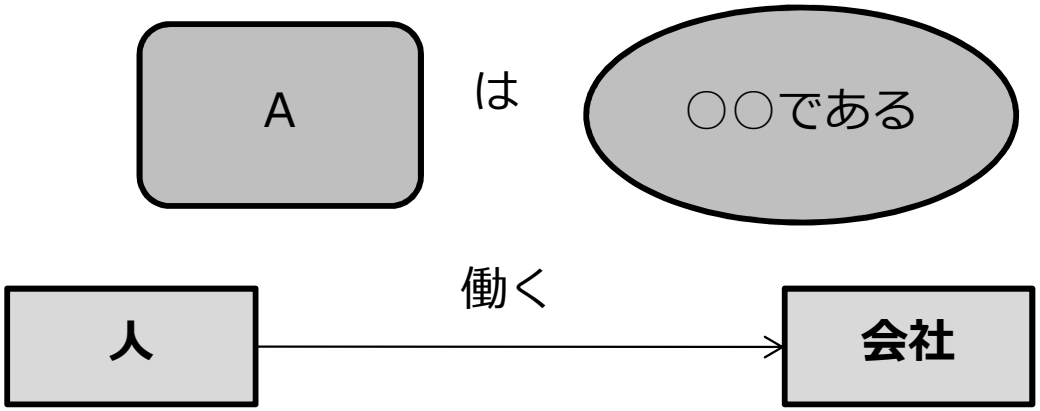
論理学とUMLの対比

論理学の用語	UML用語	補足
分類	classification	{dynamic},{multiple}
継承	inheritance	完全に同じ意味ではない
区分肢	subclass	
区分原理	powertype	《powertype》
交叉区分分割	overlapping	{overlapping}
同値区分分割	disjoint	{disjoint}
完全区分	complete	{complete}
不完全区分	incomplete	{incomplete}

概念(オブジェクト)と判断

概念とは？

- 判断
  - 判断とは概念と概念との間に成り立つ関係を述べるもの
- 判断の形式は二つの概念を結合
  - ＡとＢとを二つの概念とする判断の形式は「ＡはＢである」と表わす
  - Ａを判断の主語といいＢを述語という



概念(オブジェクト)と判断

命題論理学／述語論理学

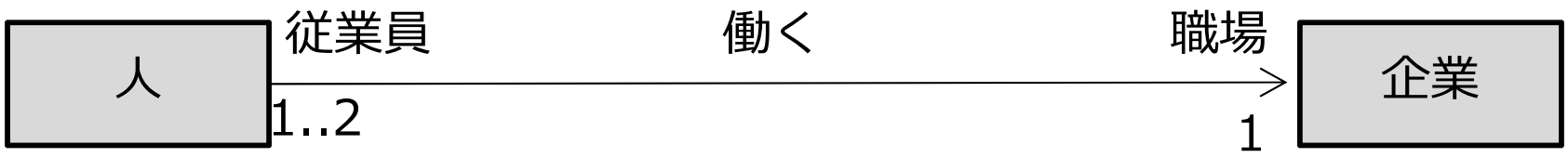
オブジェクト指向／UML／OCL

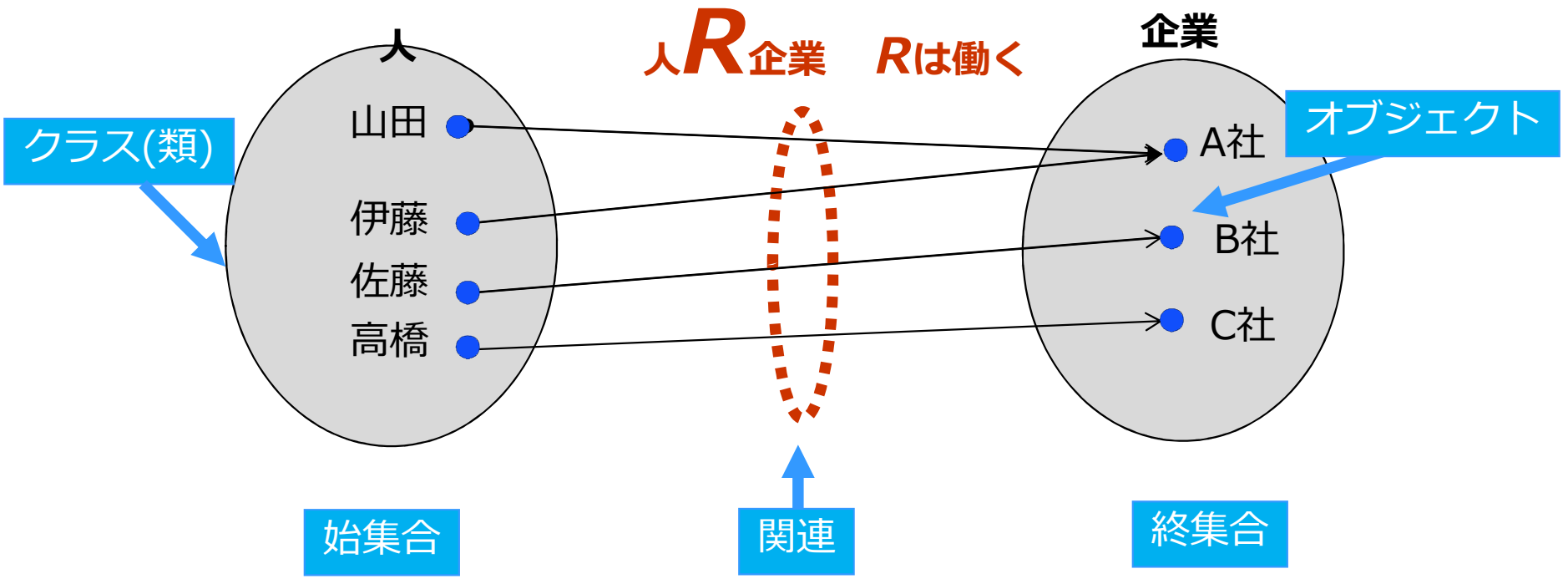
オブジェクト指向の論理的基盤

オブジェクト指向の基盤
• 述語論理／様相理論／時相理論 • グラフ理論 • 集合論と写像(関数) • 型理論(型システム)

“形式的手法”と呼ばれる

クラスの関連は数学の2項関係(aRb)





2項関係(a*R*b)の表現方法

関連(2項関係(aRb))の表現方法

- 2項関係(a*R* b)の表現方法には,表(行列),写像,数式,グラフなどがある

氏名	A社	B社	C社
山田	○		
伊藤	○		
佐藤		○	
高橋			○

2項関係(aRb)の表による表現

	A社	B社	C社
山田	1	0	0
伊藤	1	0	0
佐藤	0	1	0
高橋	0	0	1

行列による表現

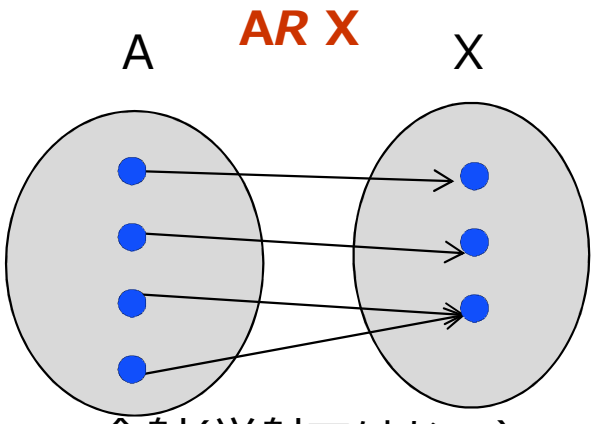
$$\begin{cases} x' = ax + by \\ y' = cx + dy \end{cases}$$

数式による表現

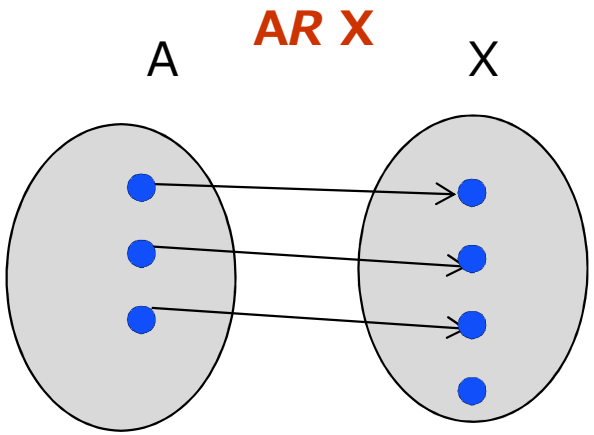
対応と写像のパターン

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

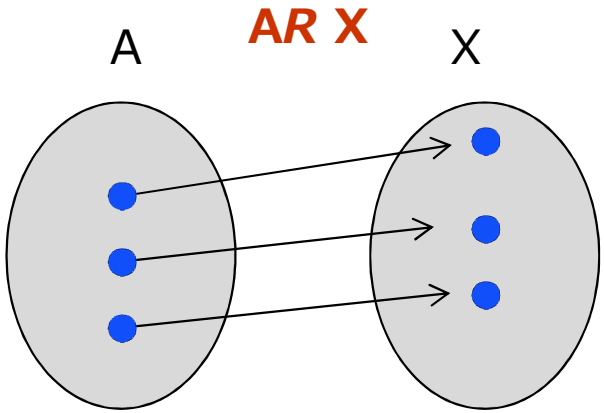
対応・写像の対応関係の基本と用語



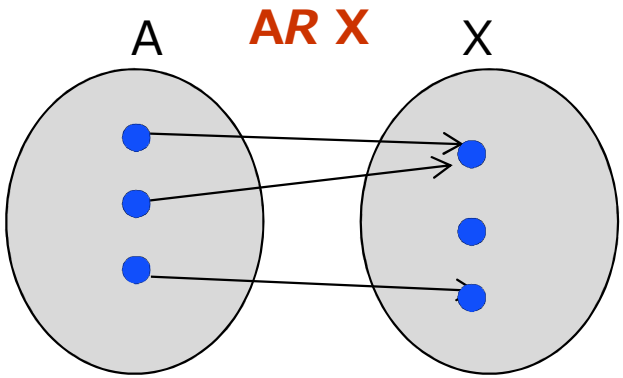
全射(単射ではない)
＊上への写像



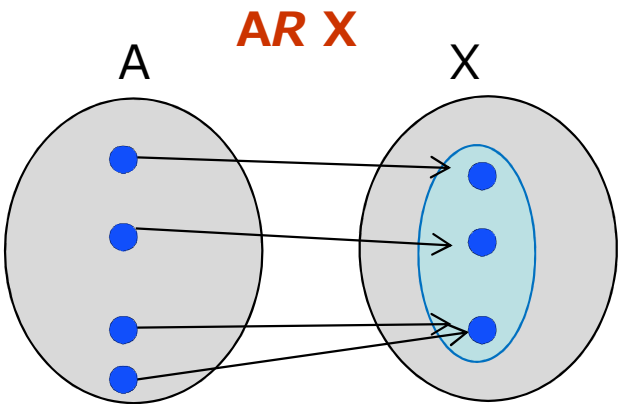
単射(全射ではない)



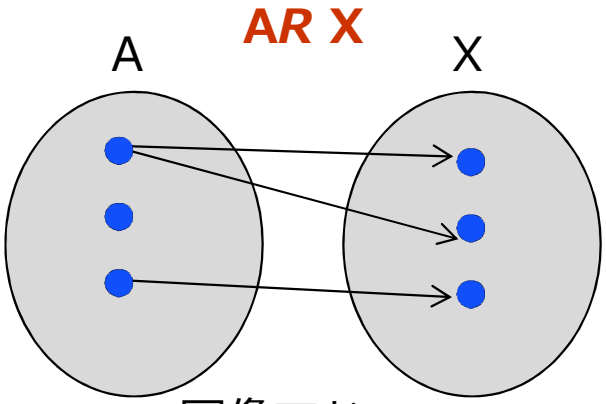
全単射(全射かつ単射)
(逆写像を持つ)



全射でも単射でもない



上への写像



写像でない

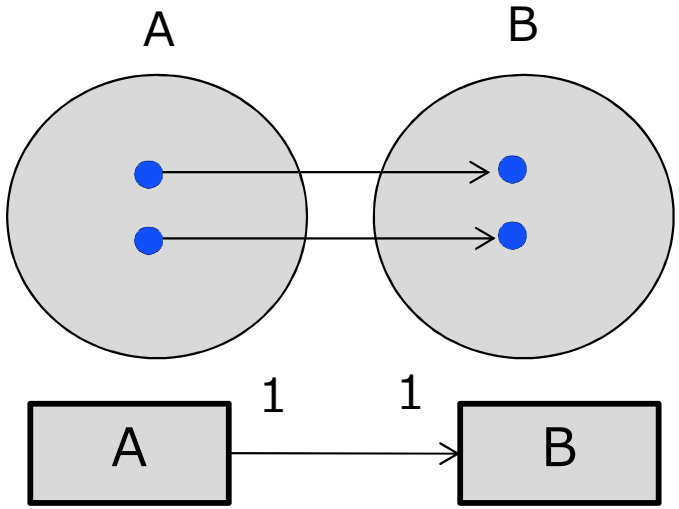
対応と写像の基本と用語

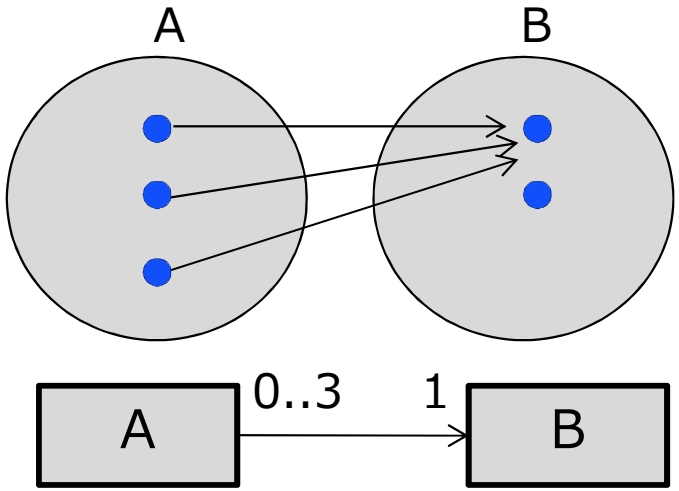
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

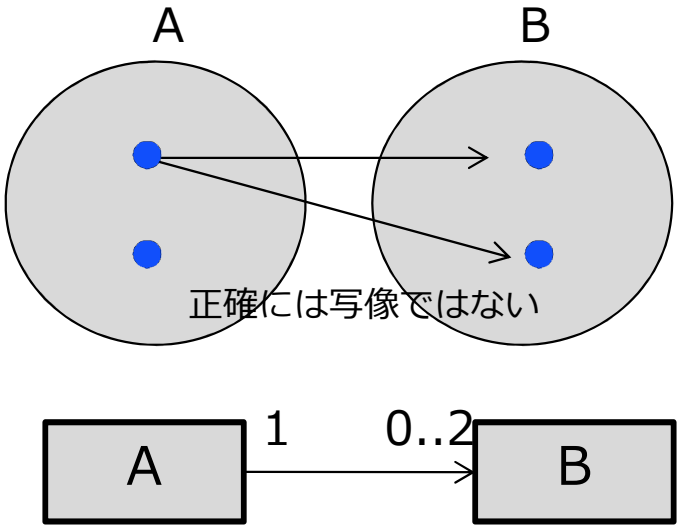
	説明
写像(mapping)	• 始集合の元(要素)が,必ず終集合の１つの元に対応する(１価性)
単射(injection)	• 始集合の元(要素)が終集合へとそれぞれ１つ対応する • (一対一対応の写像)
全射(surjection)	• 上への写像であるが,一対一対応の写像でない • 上射ともいう
全単射 (bijection)	• 単射かつ全射(一対一対応の写像かつ上への写像)(逆写像を持つ) • 双射ともいう
全射でも単射でもない	• 上への写像でもなく,かつ一対一対応の写像でもない
上への写像 (onto mapping)	• 始集合の元(要素)が全て終集合へと対応し,かつ終集合全体に対応する • 全射のとき • 逆に言えば,任意の終集合の元は,ある始集合の元の像になっている
一対一対応の写像	• 単射のこと

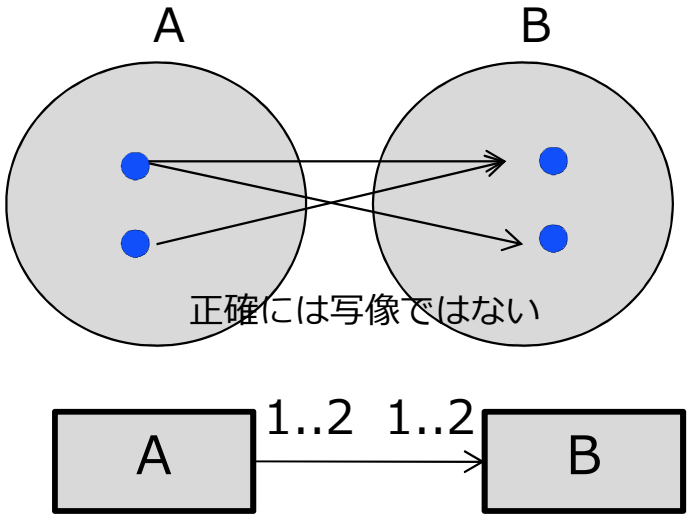
写像と多重度

写像の多重度の基本



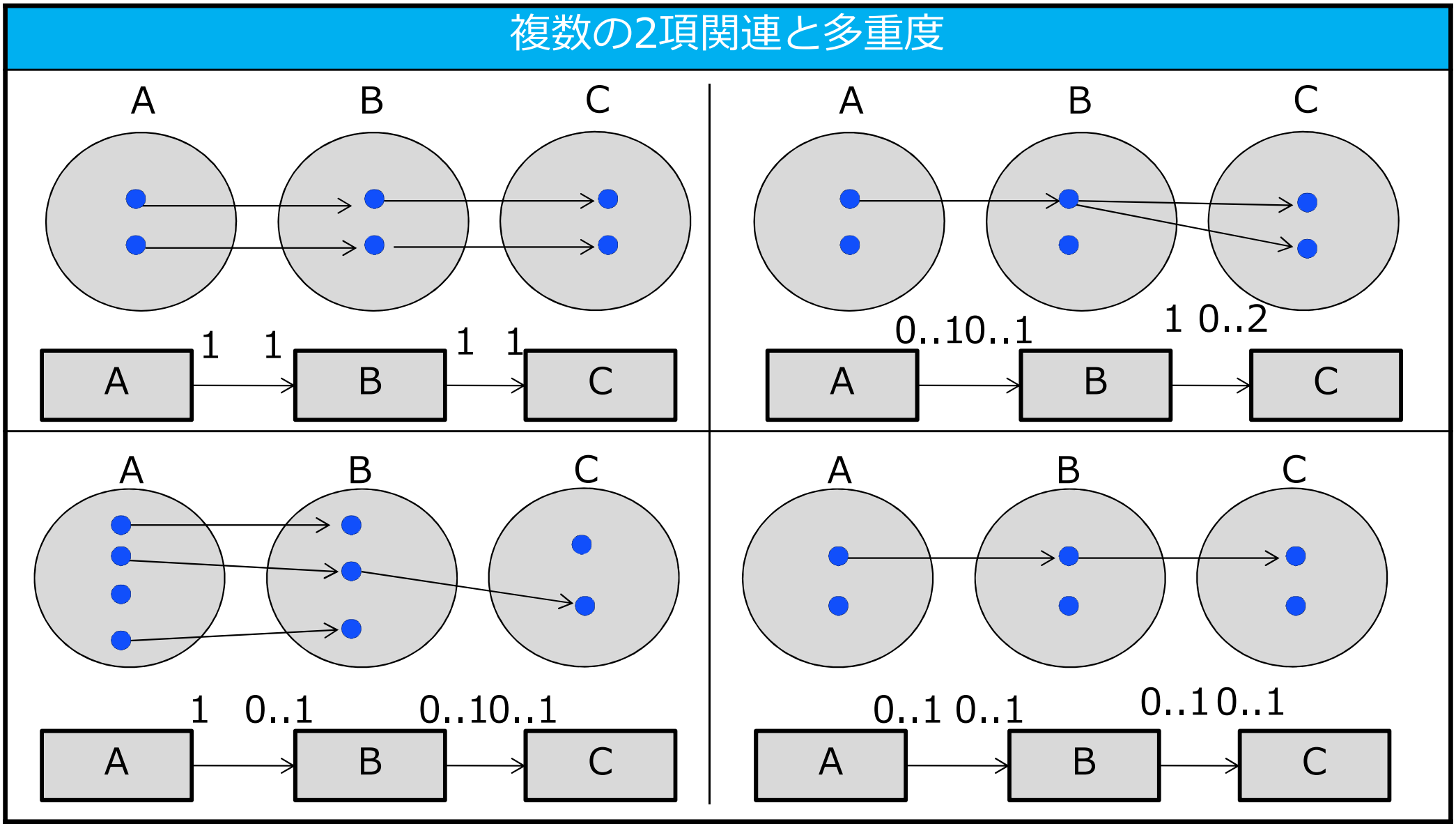






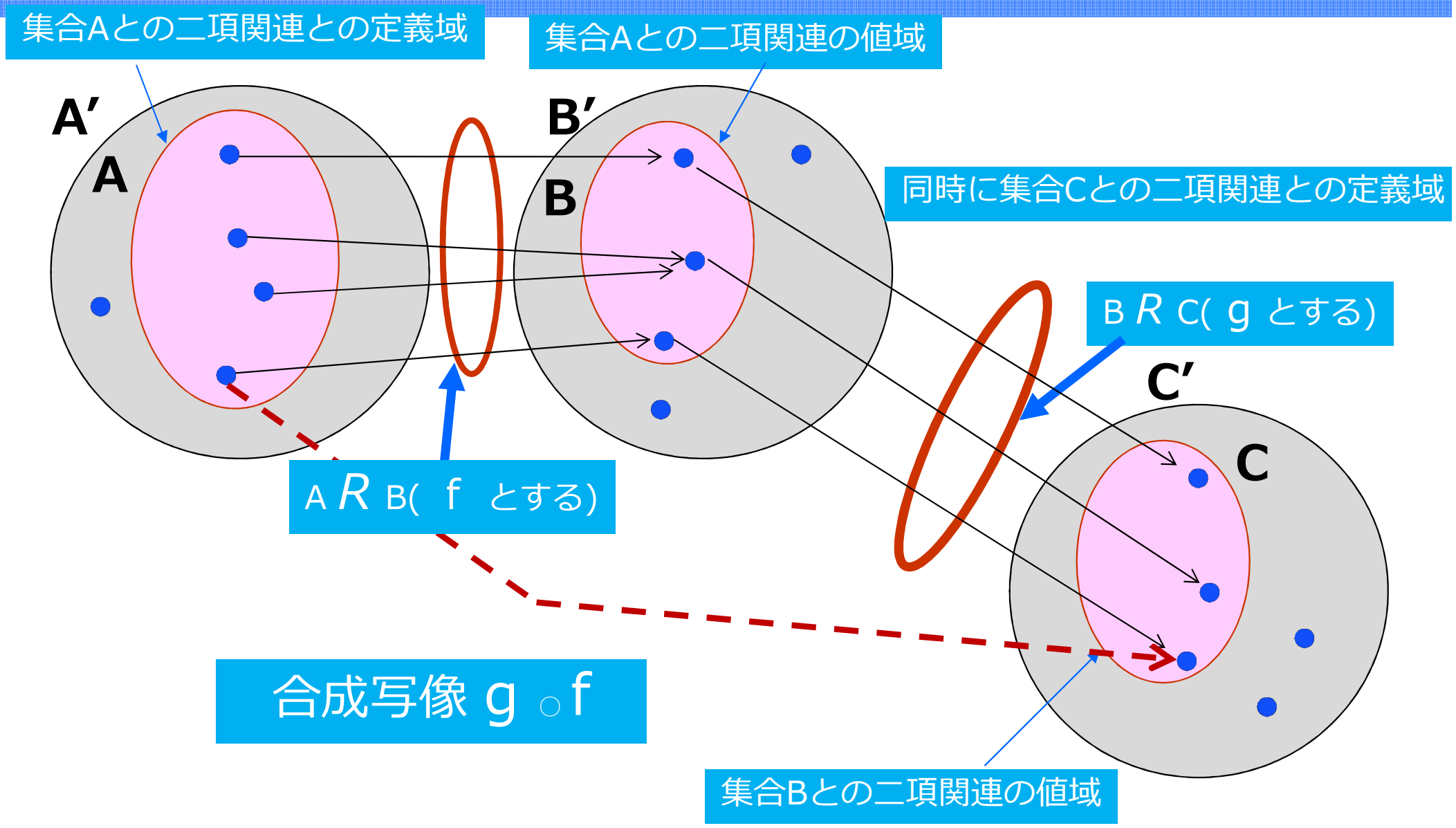
複数の2項関連と多重度

複数の2項関連と多重度



合成写像の定義域と値域

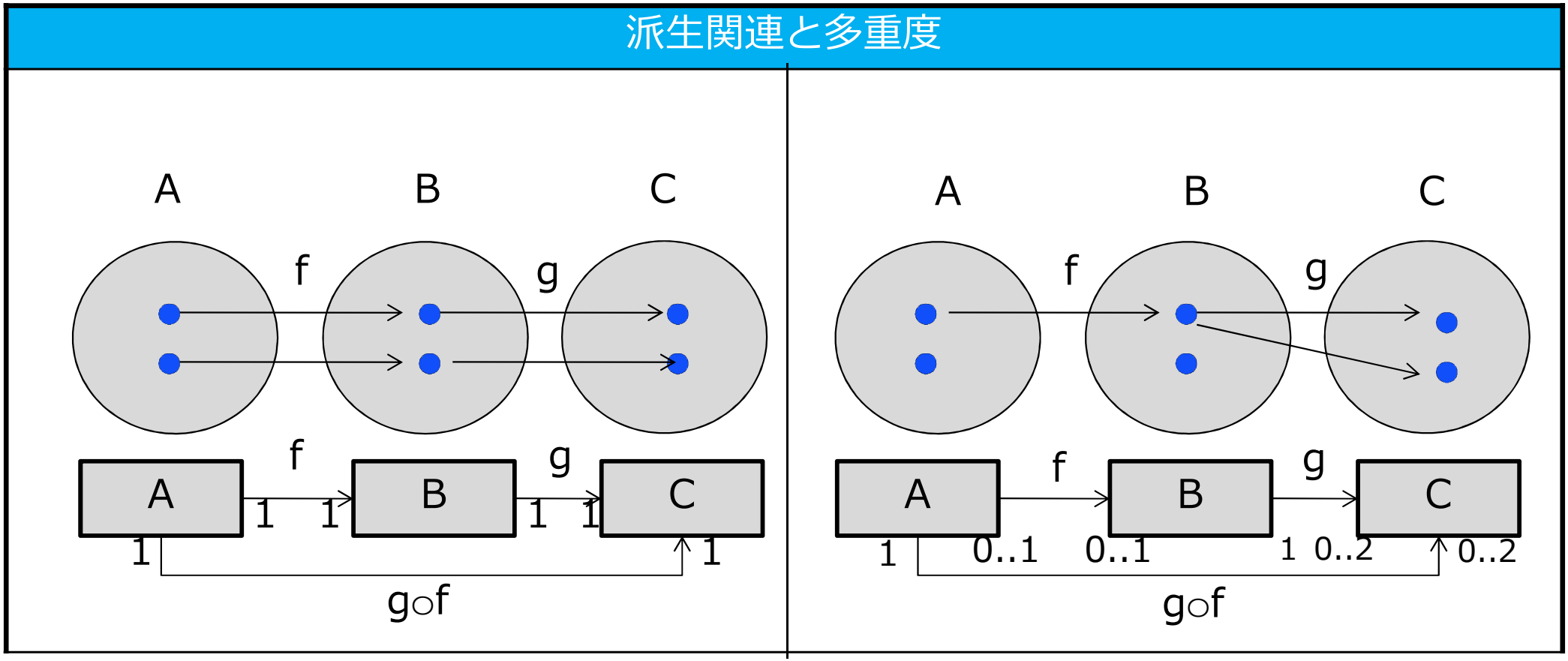
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.



派生関連の多重度

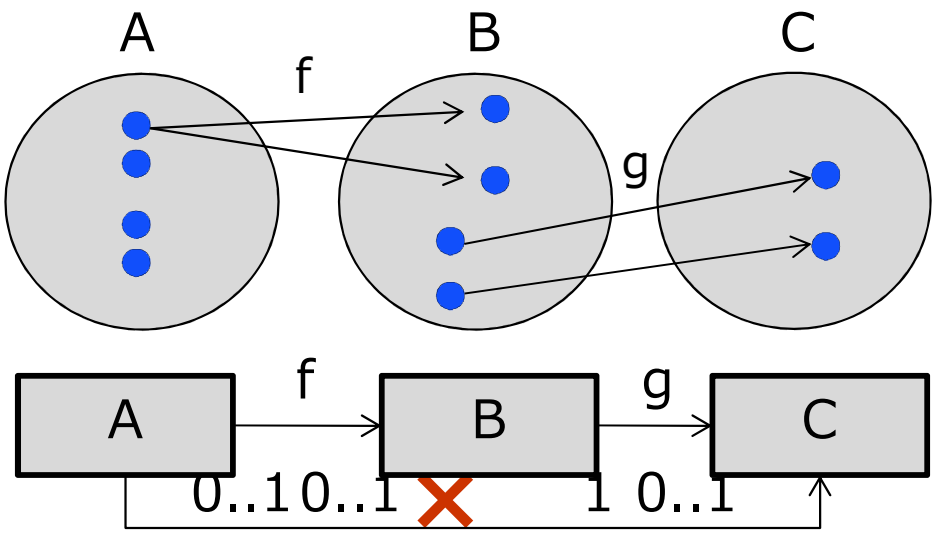
HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

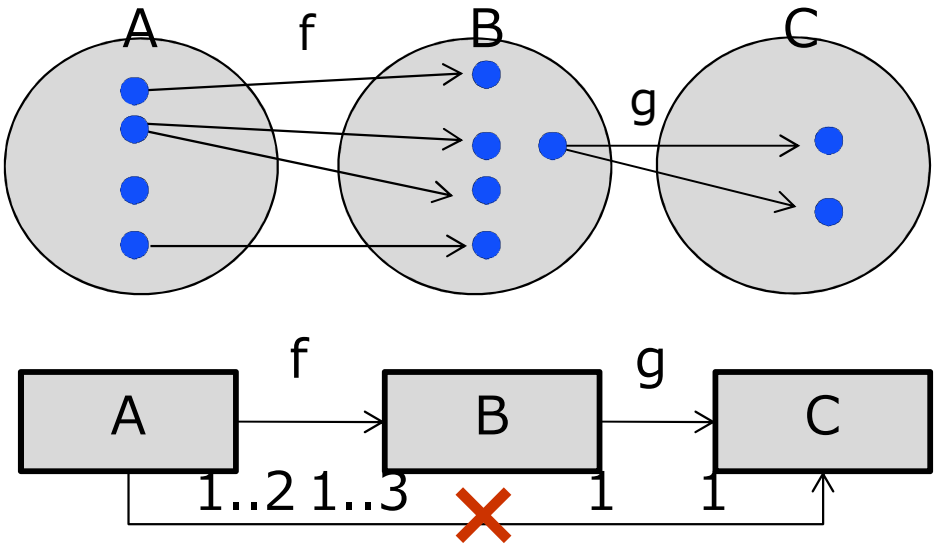
派生関連と多重度



派生関連が無いケース

派生関連が無いケース

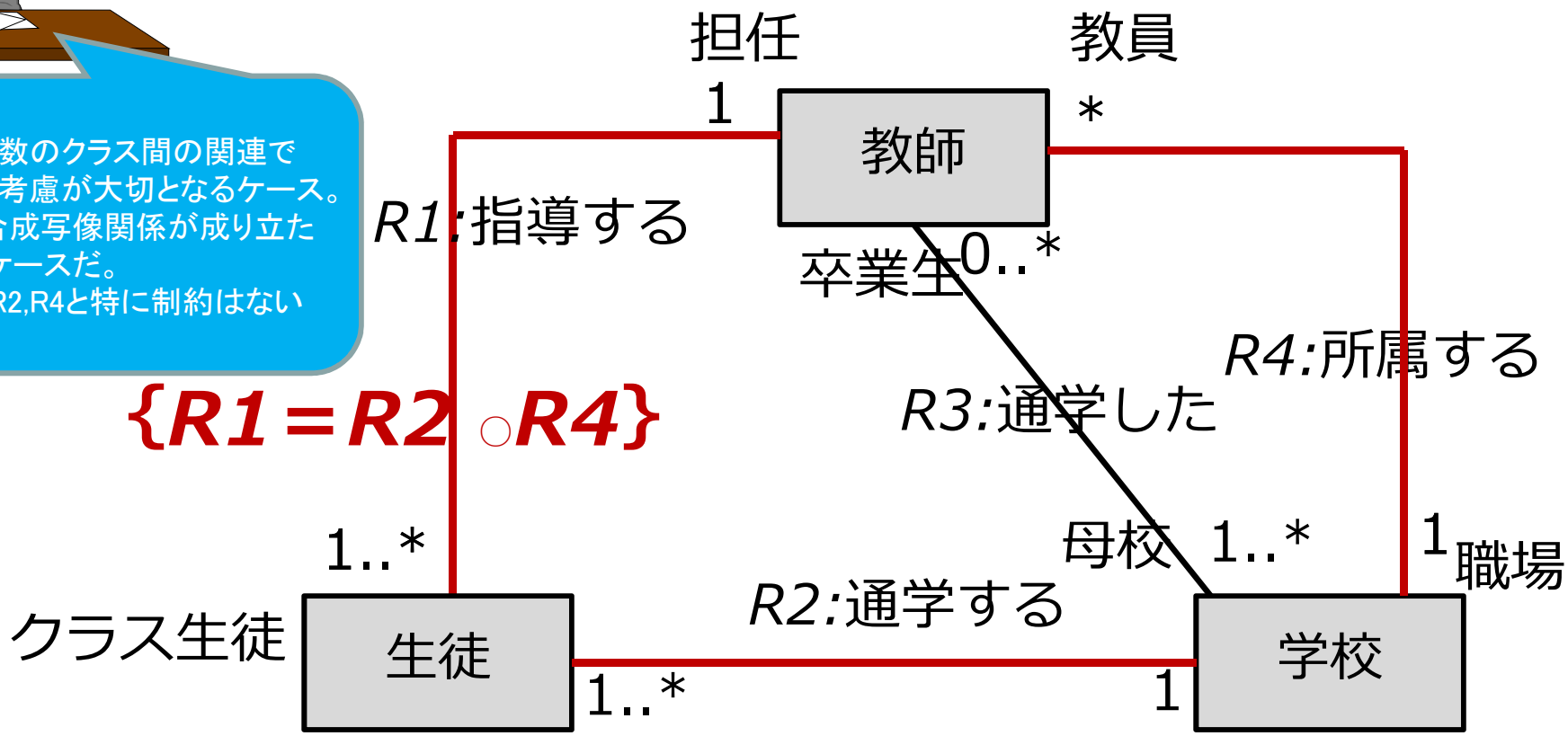




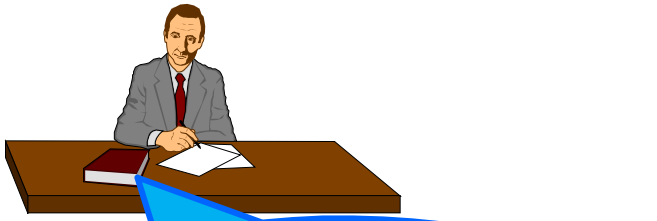
合成写像が重要となるケース①

- 担任教師は指導する生徒は自分が勤めている学校の生徒である必要がある
- 教師は自分が過去に卒業した学校に指導する生徒が通学する必要はない

- このケースは,複数のクラス間の関連で定義域と値域の考慮が大切となるケース。
- R1はR2とR4の合成写像関係が成り立たないといけないケースだ。
- R3の関連はR1,R2,R4と特に制約はない



合成写像が重要となるケース②



集合(クラス)の要素間のつながりを意識してモデル化しよう

教師

*R1:*指導する

*R4:*所属する

{*R1*＝*R2*　○*R4*}

*R3:*通学した

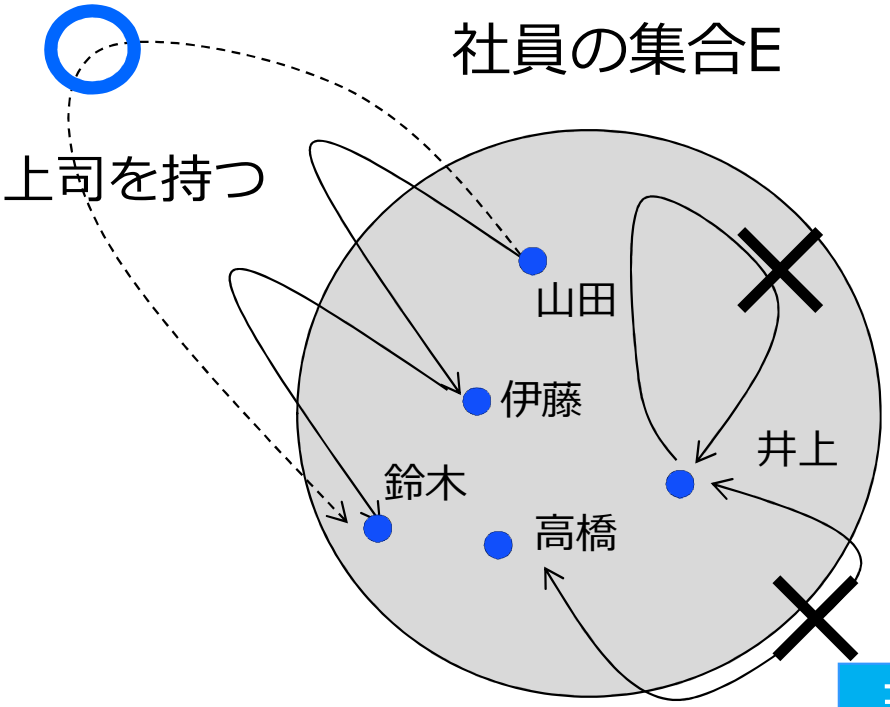
生徒

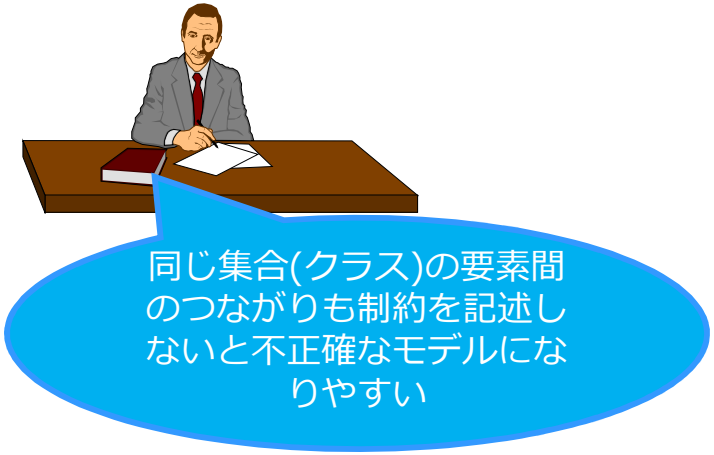
*R2:*通学する

学校

同じ集合への写像(再帰関連)

推移的に関係が成り立つ



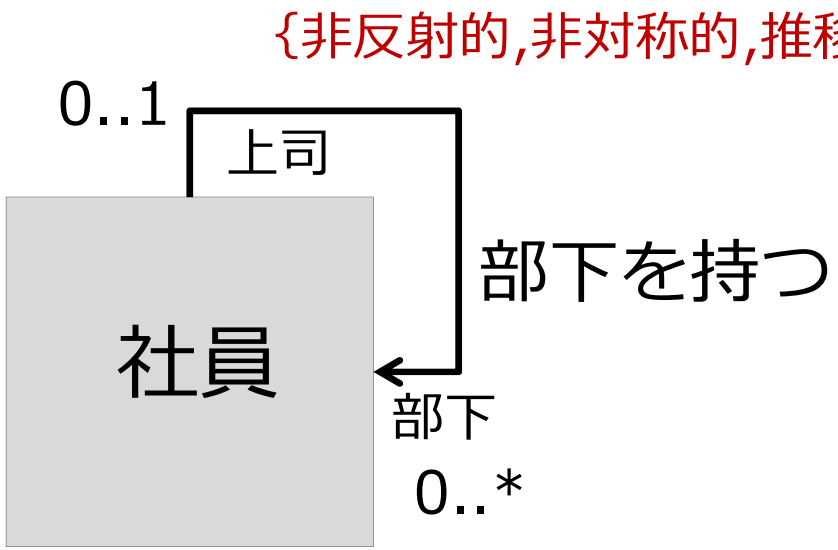


非反射：自分自身は
上司・部下を兼務できない

非対称：上司・部下の関係
は対称性はない(非対称的)

再帰関連と制約①

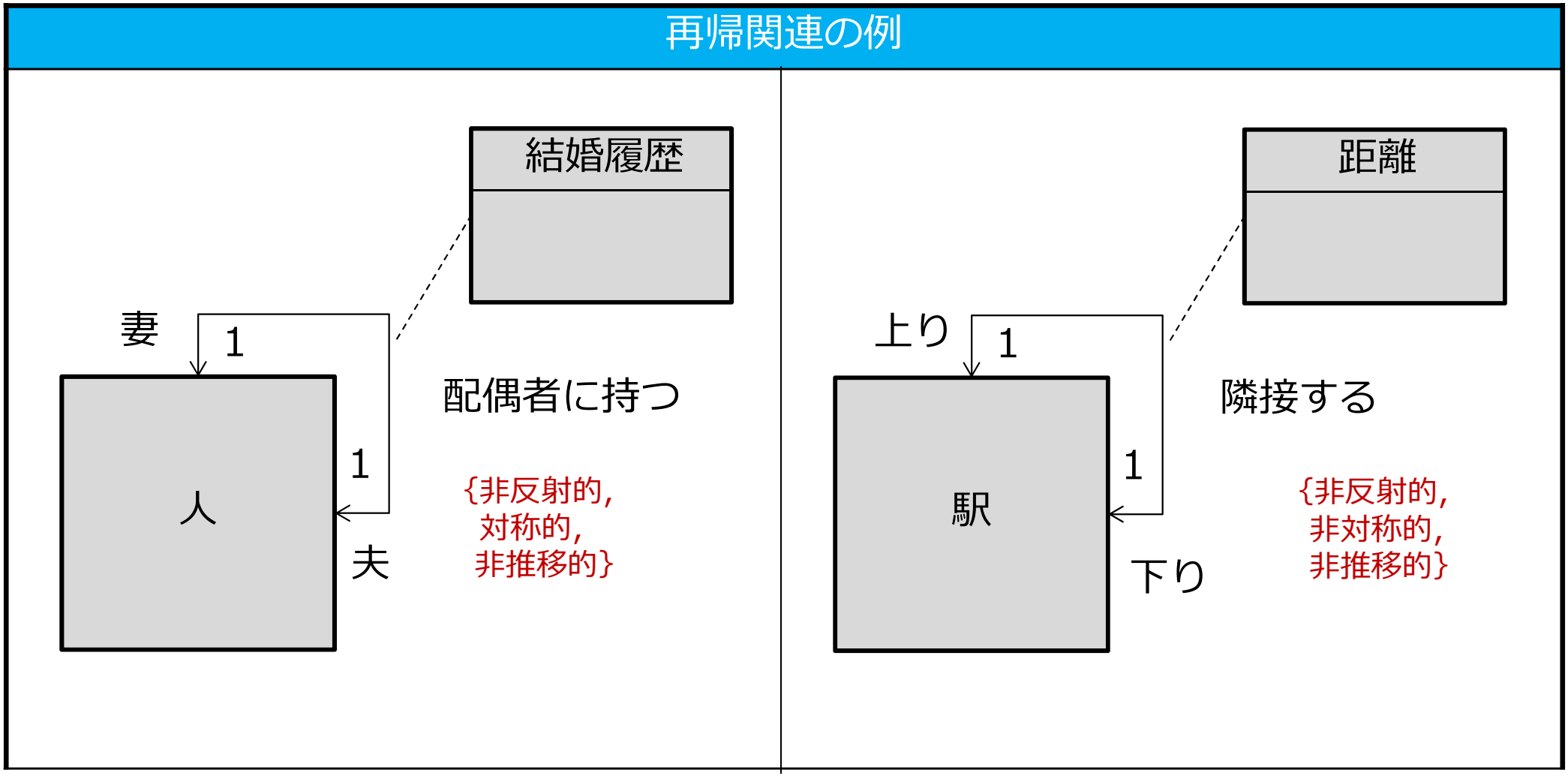
- 直属の部下は一人以上持てる
- 一番下の部下には部下がいない



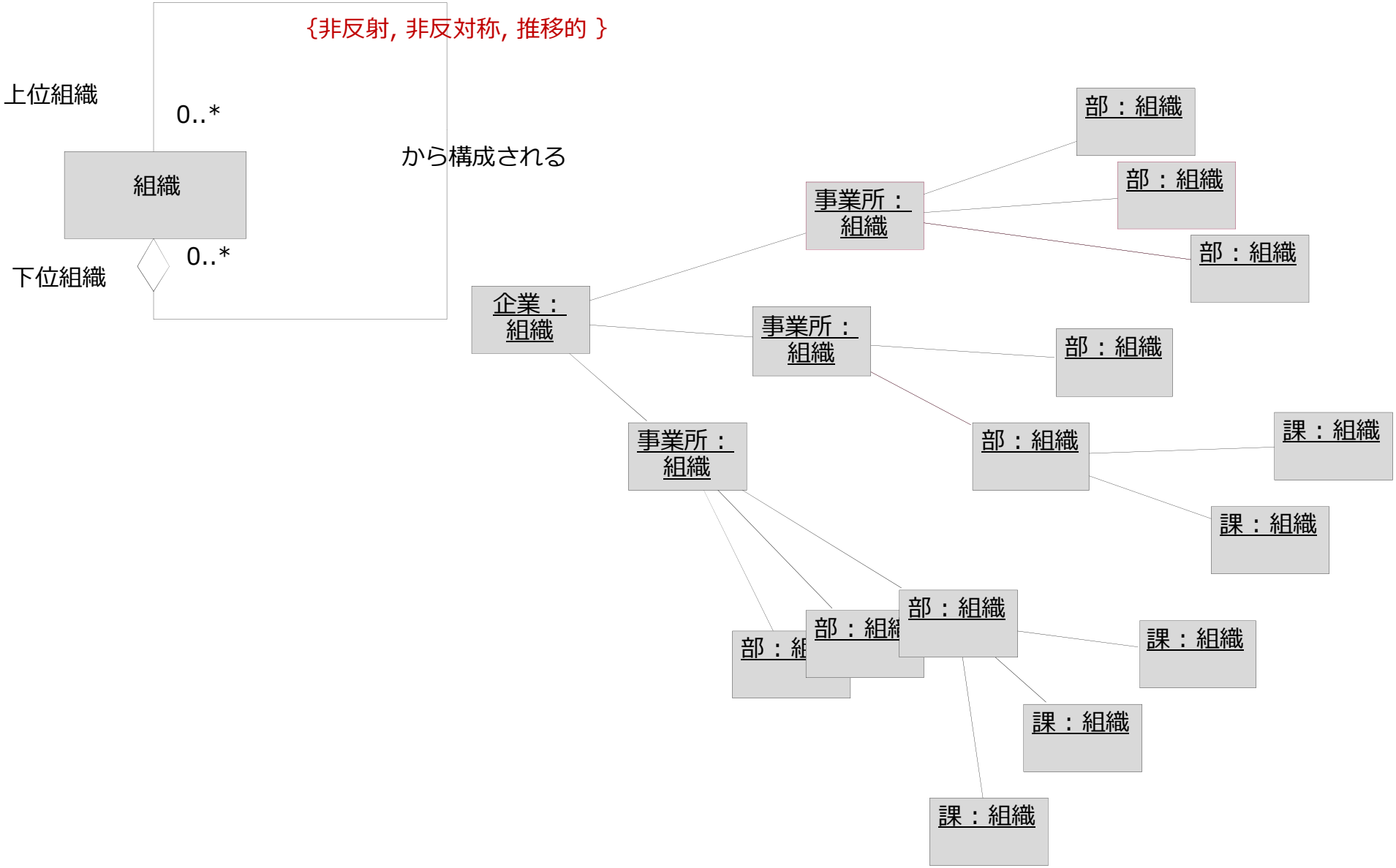
- 上司は一人のみ持てる
- 一番上の上司には上司がいない

再帰関連と制約②

再帰関連の例



再帰関連と制約の例

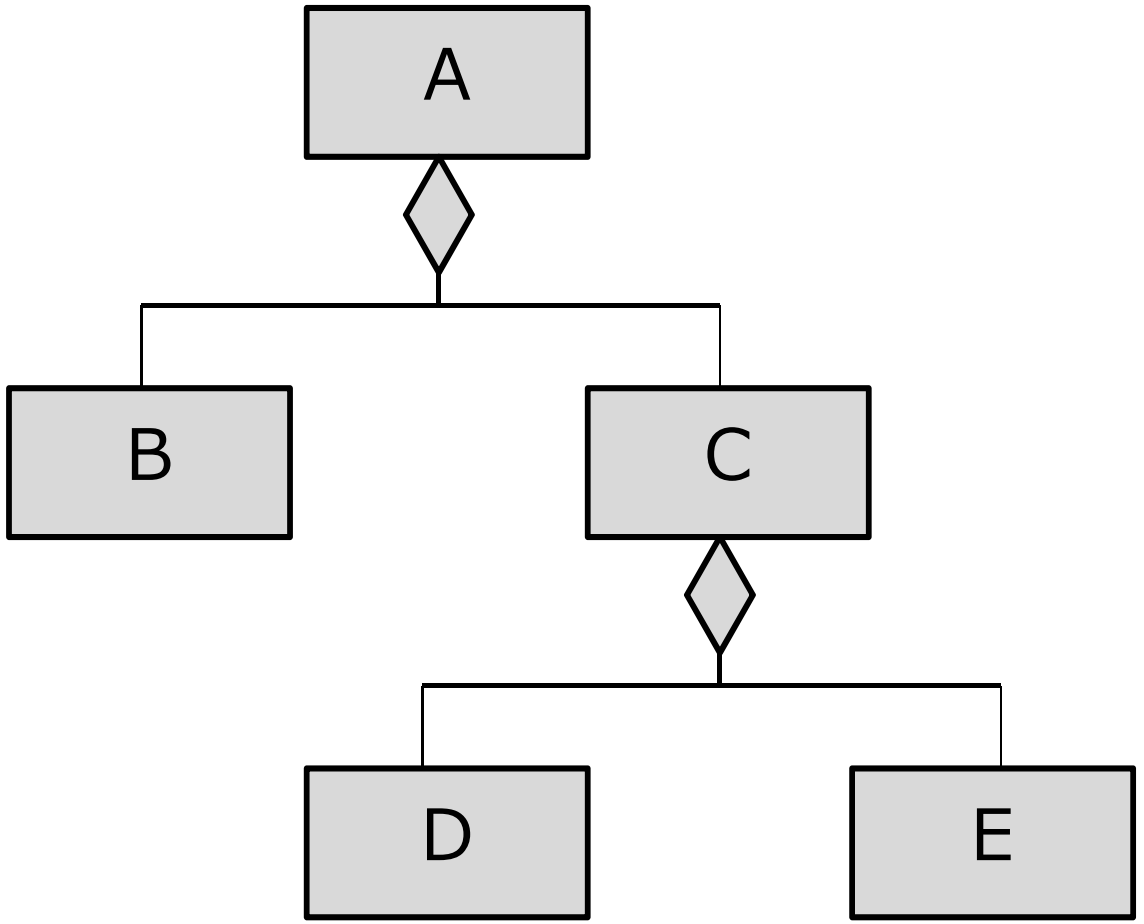


数学的関係

規則名	意味
反射律	• ∀a∈U のとき　aRaが成立
対称律	• ∀a,b∈U のとき　aRb⇒bRa が成立
推移律	• ∀a,b,c∈U のときaRb∧bRc ⇒ aRc が成立
反対称律	• ∀a,b∈U , aRb∧bRa ⇒ u = v が成立
同値関係	• 反射的,対称的,かつ 推移的関係が成立
半順序関係	• 反射的,推移的,かつ 反対称的が成立
全順序関係	• 反射的,推移的,反対称的, かつ比較可能が成立

「コンポジション(全体-部分)」関係の例①

- 「全体-部分」関係も2項関連の１つだよ
- そしてこの全体-部分関係では，「**半順序関係**」が成りたつんだ



「コンポジション(全体-部分)」関係の形式化の例②

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

「コンポジション(全体-部分)」関係の形式化

- 推移閉包T＊の要素(A,B)を使い,反対称律を保証する：
  - ∀x∀y[A(x) is subpart of B(y)
    ⇒¬A(y) is subpart of B(x)
- 推移閉包T＊の各要素(A ,B)(B,C)(A,C)を使い,推移律を保証する：
  - ∀x∀y∀z{[A(x) is subpart of B(y) ∧ B(y) is subpart of C(z)]⇒¬A(y) is subpart of C(z)}

- 関連の形式化では「閉包」を考えるのが定跡の１つなんだ。
- 「閉包」には「反射閉包」「推移閉包」「反射的推移閉包」など色々あるよ
- 形式化は大変に感じるけど,開発環境の支援を受け自動化という大きなメリットがあるよ
- これまで困難だった上流の成果物の妥当性確認にも有効だ

「コンポジション(全体-部分)」関係の形式化の例③

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.

「コンポジション(全体-部分)」関係の形式化

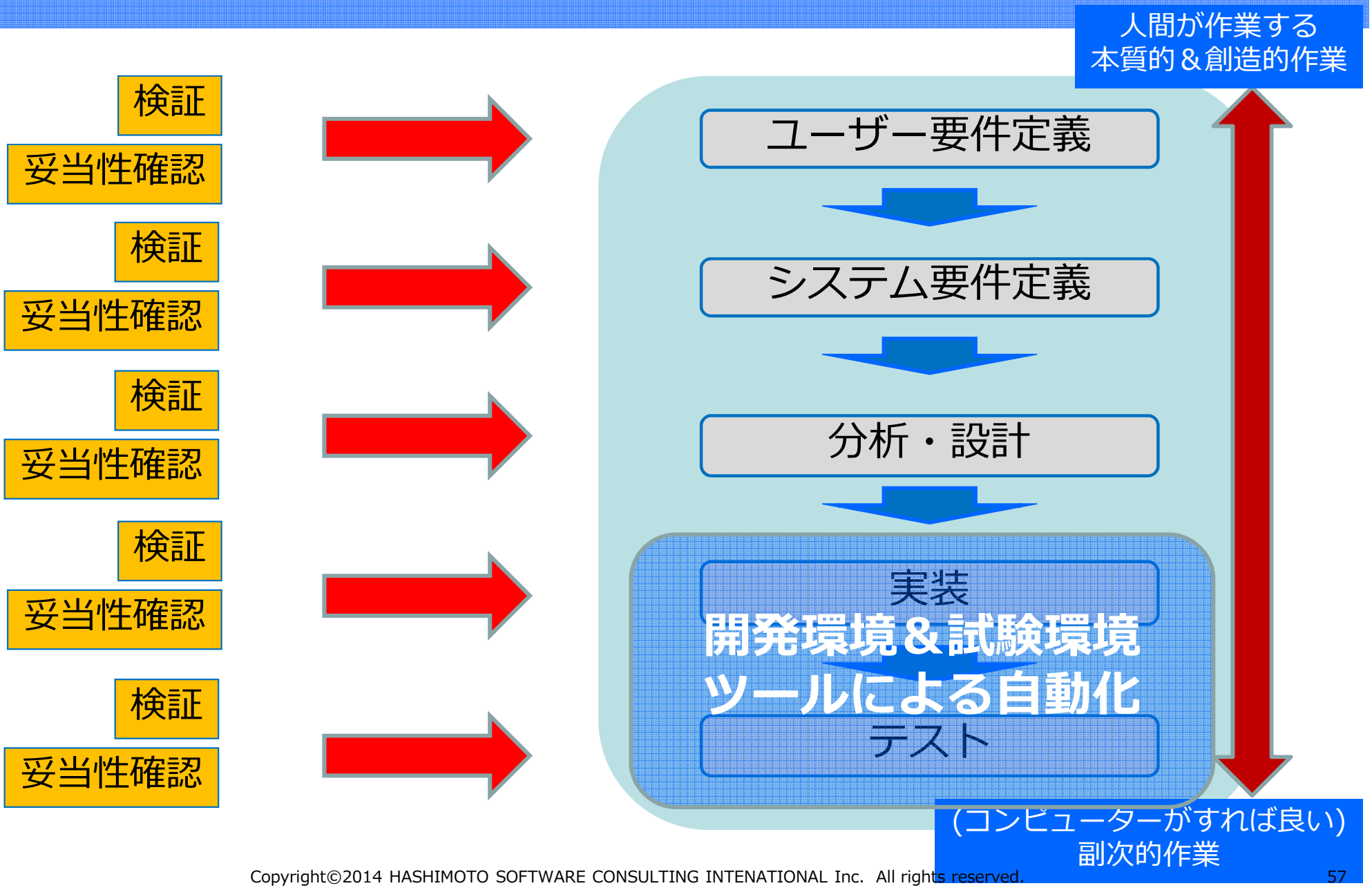
- 反対称律を保証：
  - ∀x∀y[B(x) is subpart of A(y) ⇒ ¬B(y) is subpart of A(x)]
  - ∀x∀y[C(x) is subpart of A(y) ⇒ ¬C(y) is subpart of A(x)]
  - ∀x∀y[D(x) is subpart of C(y) ⇒ ¬D(y) is subpart of C(x)]
  - ∀x∀y[E(x) is subpart of C(y) ⇒ ¬E(y) is subpart of C(x)]
  - ∀x∀y[D(x) is subpart of A(y) ⇒ ¬D(y) is subpart of A(x)]
  - ∀x∀y[E(x) is subpart of A(y) ⇒ ¬E(y) is subpart of A(x)]
- 推移律を保証：
  - ∀x∀y∀z{[D(x) is subpart of C(y) ∧ C(y) is subpart of A(z)] ⇒¬D(X) is subpart of A(z)}
  - ∀x∀y∀z{[E(x) is subpart of C(y) ∧ C(y) is subpart of A(z)] ⇒¬E(y) is subpart of A(z)}

ソフトウエア開発の自動化
～大きな飛躍の可能性～

SEIPartner
CMMI

分析作業と21世紀の開発

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.



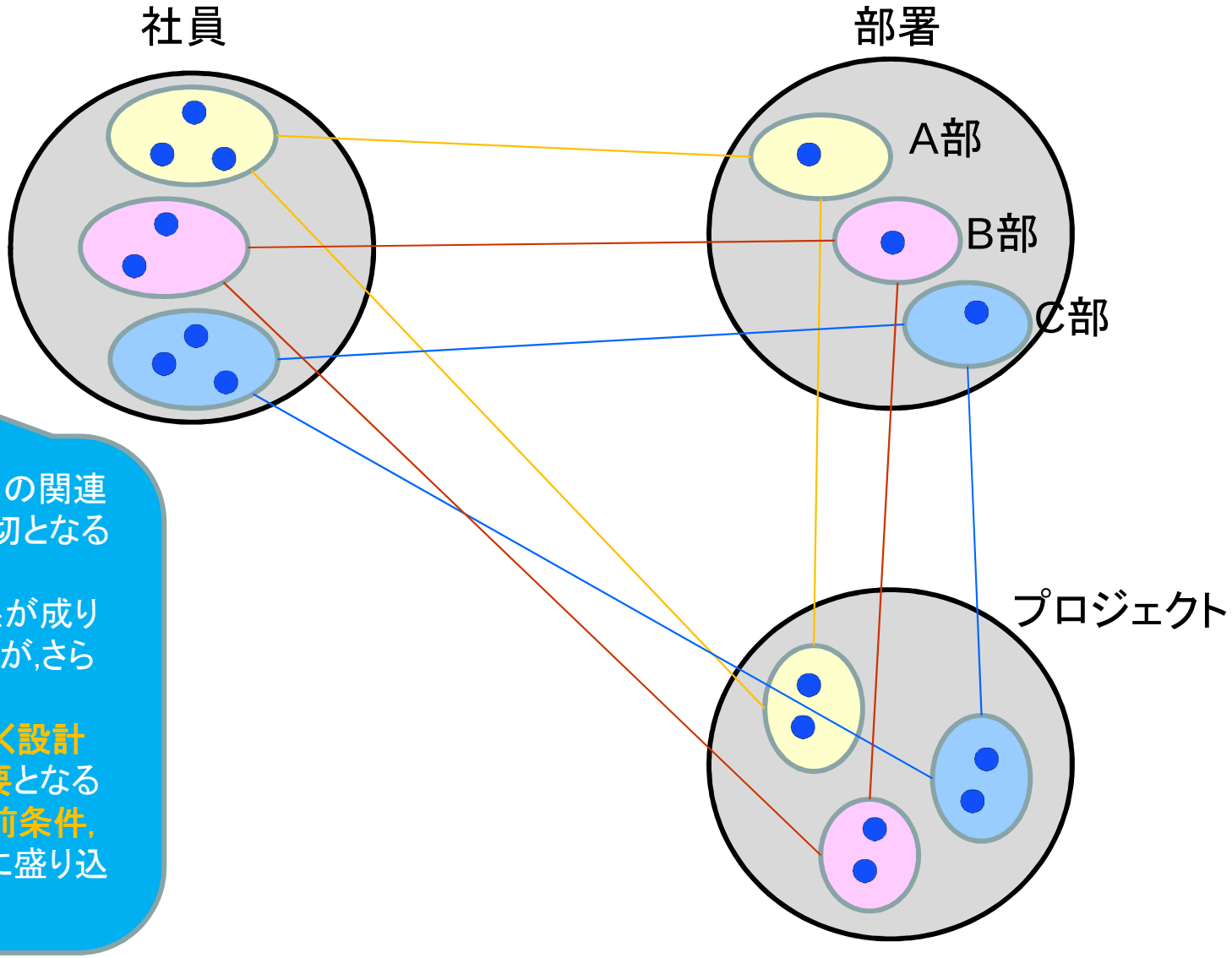
ケーススタディ

背景

- 各部署に管理しているプロジェクトが数個ある
- 各部署にはその部署で働く社員がいる
- プロジェクトに参画できる社員は,プロジェクトを管理している部署で働く社員でなければならない
- 部署が管理しているプロジェクトの予算の合計と社員の給料の合計が,部署の予算を超えてはならない
- 社員はプロジェクトは兼務できないので,「部署で働く社員」≧「部署で管理するプロジェクト数」であること

- このケースも,複数のクラス間の関連で定義域と値域の考慮が大切となるケース
- R1はR2とR3は合成写像関係が成り立たないといけないケースだが,さらに注意が必要だ
- こういうときにOCLが効果を発揮する

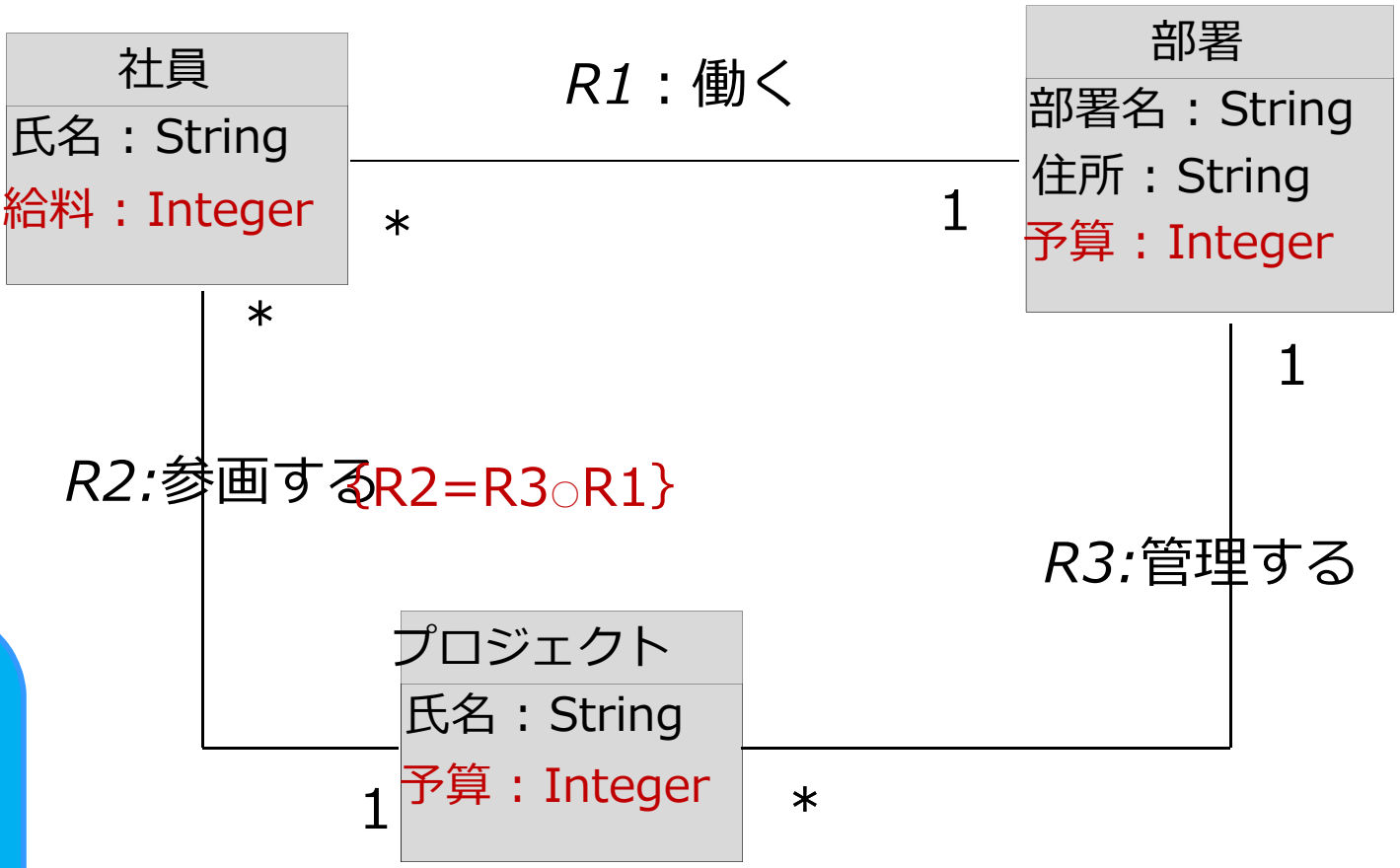
集合と写像



- このケースも,複数のクラス間の関連で定義域と値域の考慮が大切となるケース。
- R1はR2とR3は合成写像関係が成り立たないといけないケースだが,さらに注意が必要だ。
- このケースでは**「契約に基づく設計(Design By Contract)」が重要**となる
- つまり,**クラスの不変条件,事前条件,事後条件**を明確にして制約に盛り込むことになる

クラス図

- 単純にクラス図を描くとこうなる
- モデルを工夫すれば問題文の意図をUMLだけで表現できるがOCLで書いた方が自動化と再利用性から効果がある
- UMLのモデルとOCLから制約を含めてコード生成できるので常に整合性がとれたモデルになる

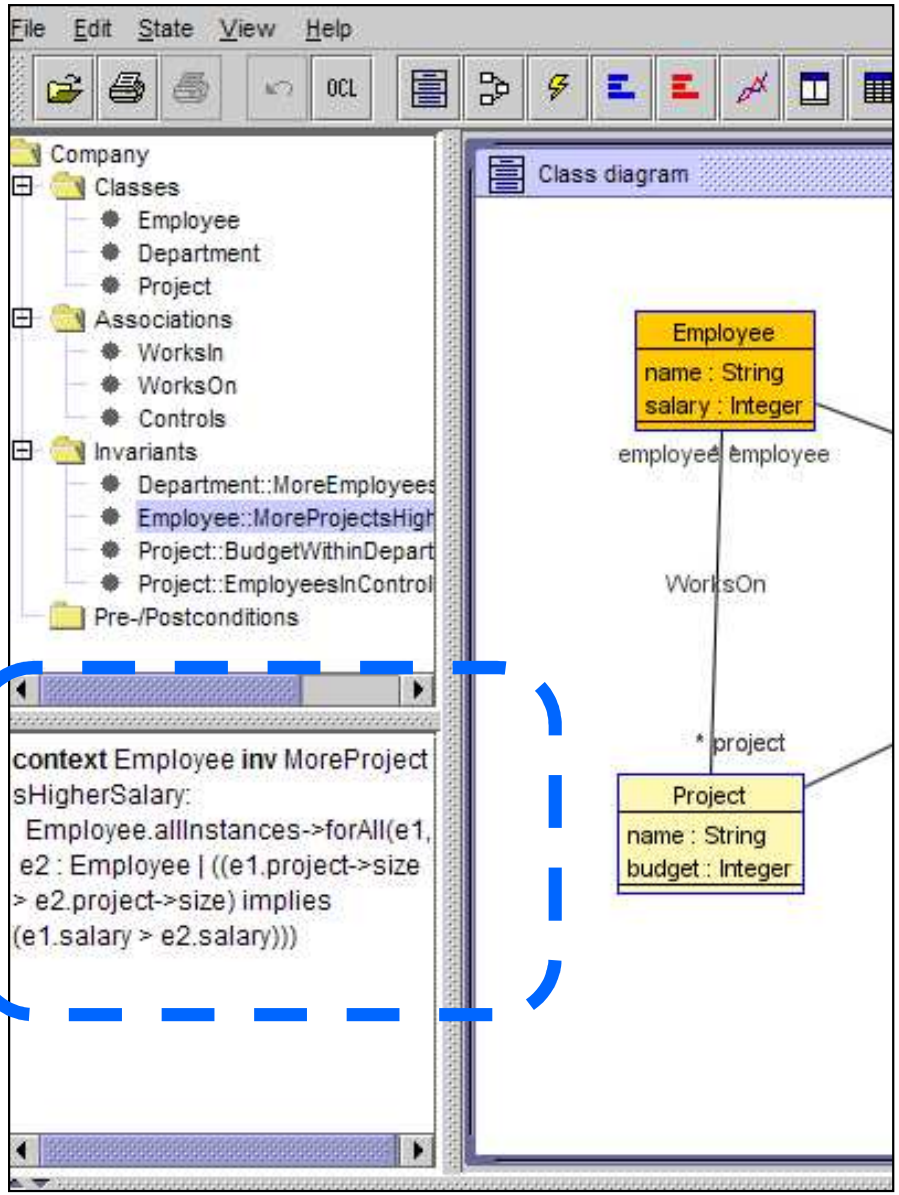


{部署の予算≧Σプロジェクトの予算}

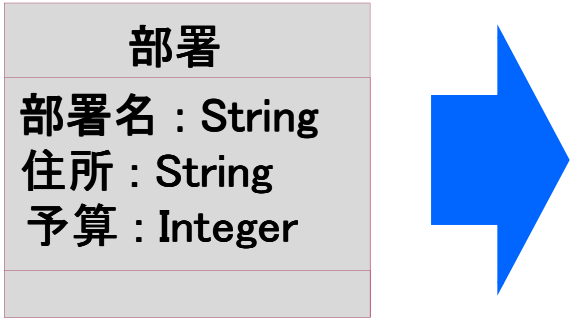
{部署で働く社員数≧部署で管理するプロジェクト数}

UMLとOCLでモデルを正確に記述する①

OCLによる記述

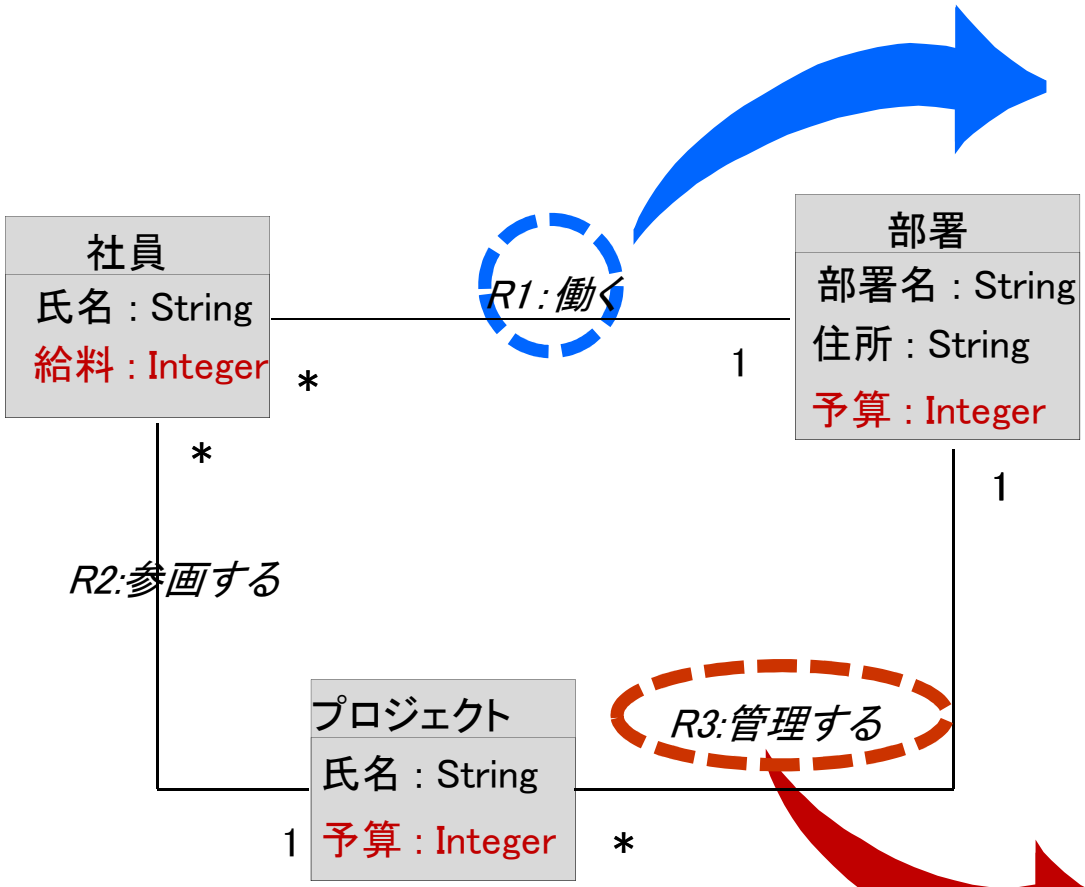


UMLとOCLでモデルを正確に記述する②




```
model Company

-- classes

class Employee
attributes
  name : String
  salary : Integer
end

class Department
attributes
  name : String
  location : String
  budget : Integer
end

class Project
attributes
  name : String
  budget : Integer
end
```

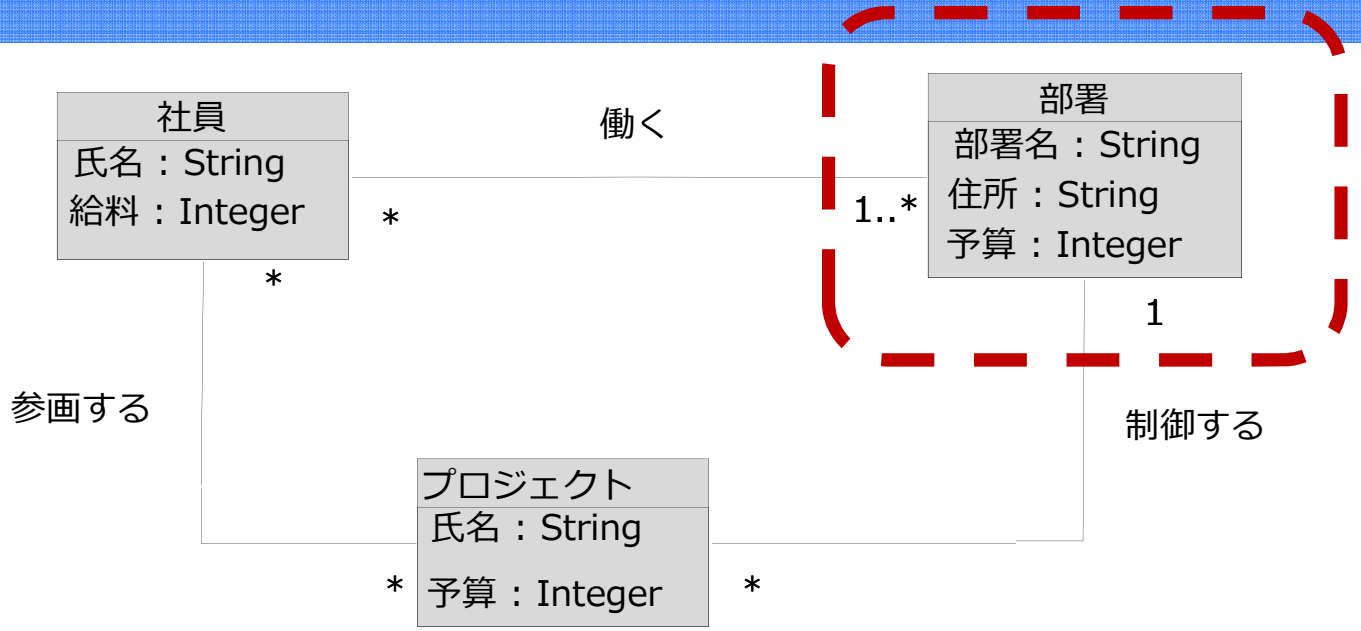
UMLとOCLでモデルを正確に記述する③




```
-- associations

association WorksIn
between
  Employee[*]
  Department[1..*]
end

association WorksOn
between
  Employee[*]
  Project[*]
end

association Controls
between
  Department[1]
  Project[*]
end
```

UMLとOCLでモデルを正確に記述する④

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.




```
-- OCL constraints

Constraints

context Department --部署
    -- 部署で働く社員は,部署が管理する
    -- プロジェクトと等しいか多く無ければならない
inv MoreEmployeesThanProjects:
    self.employee->size >= self.project->size
```

ETロボコンの参加と価値
～新しいことへの
挑戦のチャンス～

SEIPartner
CMMI

新しい開発アプローチへのチャレンジ

HASHIMOTO
SOFTWARE
CONSULTING
INTERNATIONAL Inc.

ETロボコンだからこそできるチャレンジ

- 斬新なアーキテクチャ設計とその評価
- 新しい手法の利用とその効果
- あたらしい開発環境の利用と評価

Thank You

HASHIMOTO SOFTWARE CONSULTING INTERNATIONAL Inc.