

Business Process and Requirements: Moving to Specification

Dr Keith Phalp

Recap – So far.

- Developers build systems for clients
- “Oh dear. The system doesn’t seem to meet the client’s needs”.
- One reason is that the developers didn’t understand the problem: or what they wanted or needed.
- We have reviewed what we mean by analysis and requirements (and found issues with current and past methods).
- We have suggested business process models to understand better and to inform requirements and specification.
- We have examined 'what we want' from such models, considered alternatives and introduced role models.
- We have tried out Role Activity Diagrams – at the same time discussing other aspects of the analysis – noting issues and finding improvements.
- STILL TO COME...
- Moving from process model to specification – some different views.
- A related topic: ensuring alignment
- Quality of models – how we gauge, review and revise models.

Moving to Specification

- The general view is that by undertaking the process modelling we:
- Increase our understanding of the client domain and write better requirements.
- Find issues / ambiguities - clarify these and thus write systems which meets their needs
- Spot things where we can improve the process - and added bonus for the analyst.
- However, ideally we want more than this...
- Want to ensure that the things we have learned are 'mapped' to the specification (don't lose our knowledge gained).
- Want to be able, via such mappings to show where requirements are met, or considered by specification.
- Also need, as with classical specification, to understand where we put the system boundary.

Different Views

- Having agreed what we want, views differ both to how straightforward our transition will be, and of course the best ways to do it.
- In order to be able to discuss this in a learned fashion (*such as in an assignment*), I will present some different views of these issues. (There are far more of course).
- Note also that successful mapping clearly has a bearing on alignment.
- However, in practice, I have tended to use (and teach) for practical purposes, the method of system roles – though even here, this can be less clear depending on the nature of the process and it's description (contrast insurance with tendering example).

What Views

- In brief, these views include:
- Role Models and Use cases are orthogonal, mapping will therefore be difficult, and we need to use an intermediate notation (POSD), though can later achieve similar results by annotations / groupings.
- The issue is that use cases do not have the expressive power of the RAD, and, therefore, we need to augment them, so that we can still represent dependencies etc., which we have learned are important to the process.
- The important issue is that of identifying system boundaries (to move to specification) and, therefore, we introduce system roles on the RAD (not very purist) to enable this.
- We need to embed within other approaches, e.g., within the CIM phase of model driven development.

The Orthogonal Argument

- This suggests that Roles, Use Cases and Objects are orthogonal views, and that this makes mapping 'difficult'.
- We will introduce different ways to cope with this.
- A rigorous way – via an intermediate notation (POSD): See POSD lecture.
- An ad-hoc way, annotating the RAD (grouping activities) and mapping to use case manually.



**Bournemouth
University**

The Lack of Power

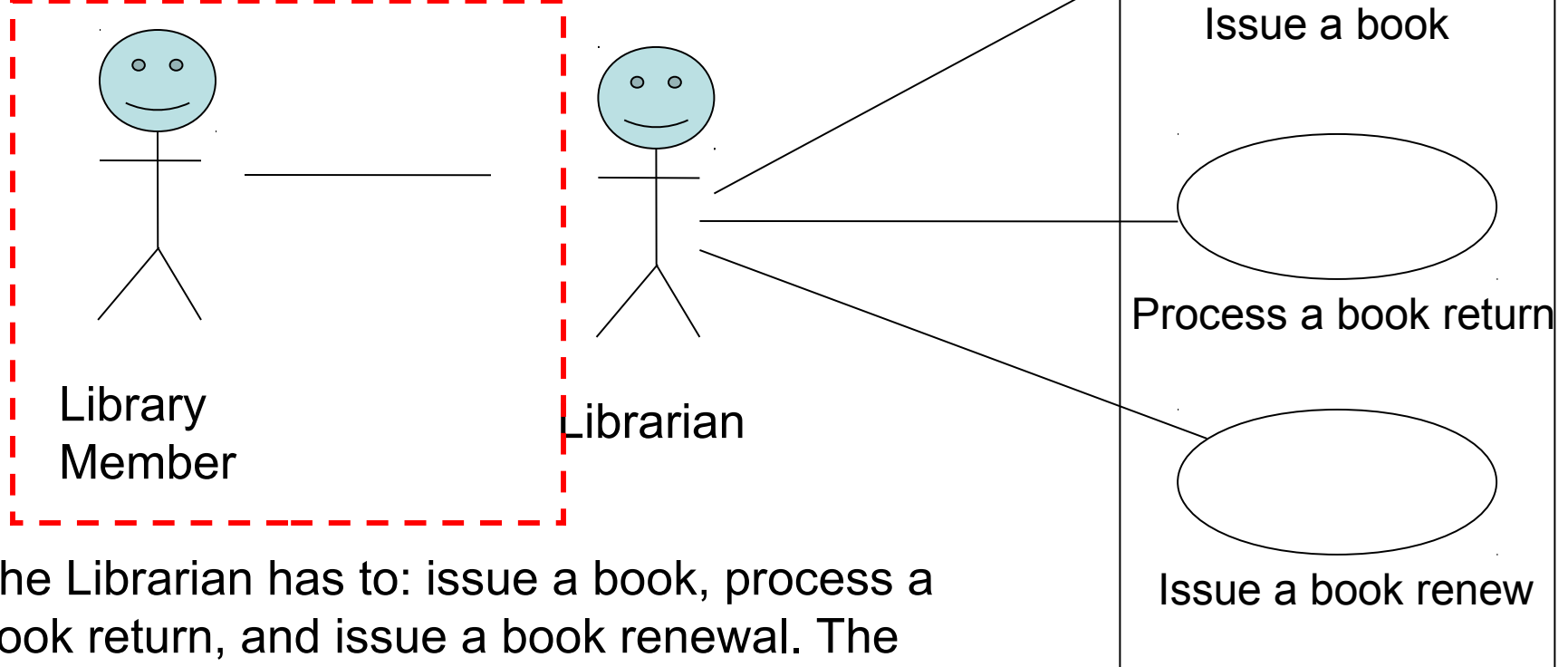
- Main observations:
- That the expressive power of the RAD is lost in moving to use case specification.
- That specification should be able to express dependencies (some astonished at this problem).
- That intermediate (or additional) specification approaches add time (and may introduce difficulty).
- E.g., state mechanisms in conjunction with use cases (state-charts etc, see Budgen), RolEnact, CSP etc (Phalp et. al., Abeysinghe et al.)
- That augmentation of the use case description will solve main (if not all) issues.
- That allowing this augmentation will improve alignment (REBNITA 2005).
- SEE *Educator* lecture.

The System Boundary

- It is clear that specification is about the interface between the problem domain and the machine (system).
- Use cases (and SASD before them) describe an explicit system boundary and the interactions (actor to use case) across this boundary.
- Indeed, the use case ignores domain information, such as interactions among actors outside this boundary.
- Hence, in moving to specification (use cases) we need to be clear about identifying the boundary (system scope), and what activities need to be dealt with by the system (including, of course, interactions between actors (roles) and the system).
- We may have systems, sub-systems, legacy, and so on, with which we will need to interact. (Different systems).
- Finding the boundary in the problem may not be clear, and may involve some decisions (perhaps in consultation with the client).

Use Case System Boundaries

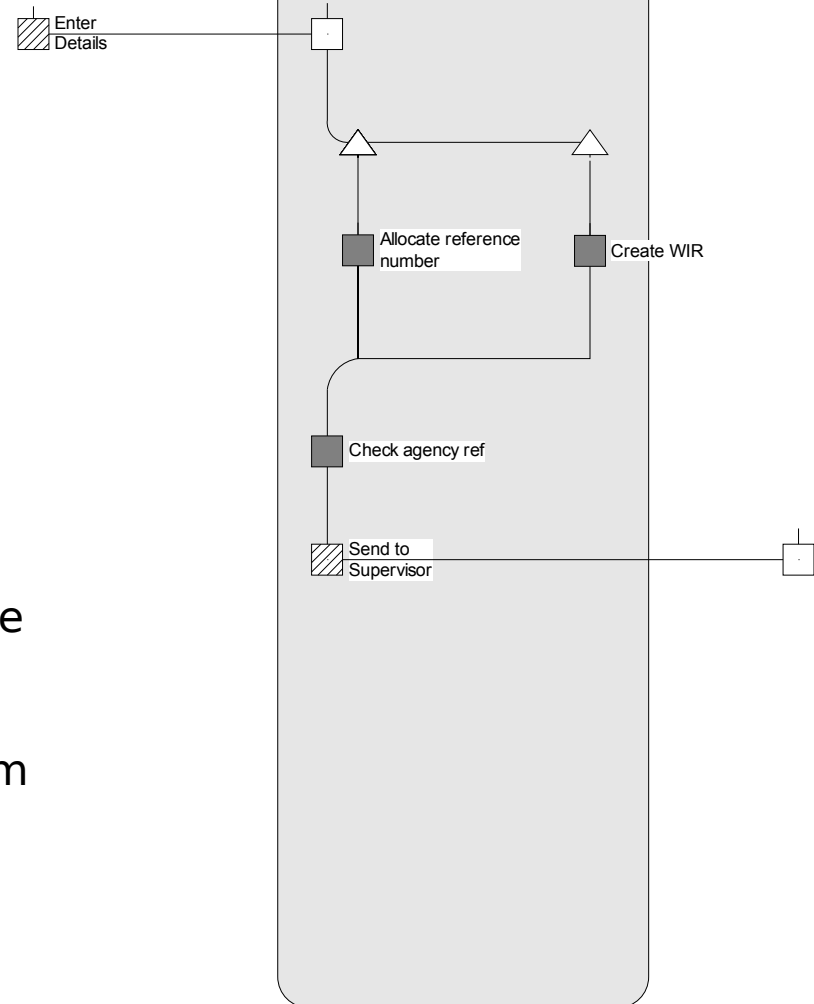
The Library Member's requirements: borrow a book, return a book, renew a loan



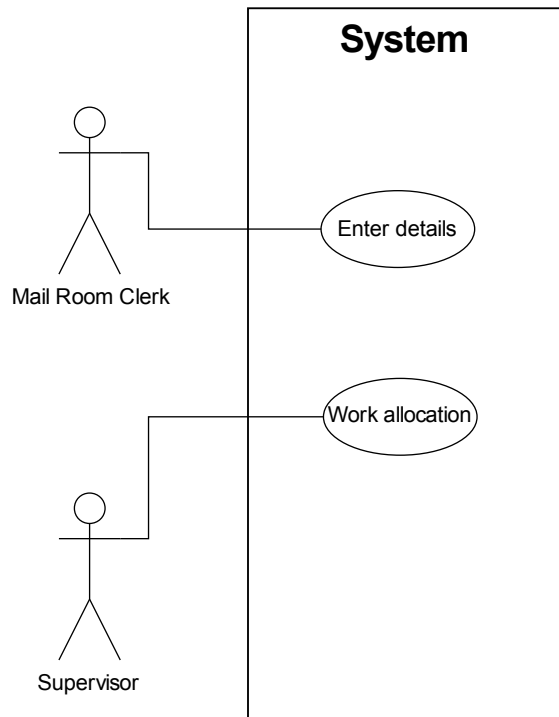
The Librarian has to: issue a book, process a book return, and issue a book renewal. The Library Member's requirements aren't quite the same as the Librarian's tasks.

Software View

- This is against the purist approach, and a rather simplified example.
- We (as software engineers) move towards specification.
- Need to ensure that we capture the system boundary (as with say a Yourdon Context Diagram).
- Need to ensure that, in moving to spec, we show cross boundary (problem to machine interactions).
- With a system RAD (usually will have different sub-system names) the interaction is between the roles (which will be actors) and the system role.
- This will correspond to use case communications, e.g., associations.



A simplified use case



- Hence, RAD acts as a way to consider the problem domain (inform requirements).
- RAD (with system roles) allows one to 'discover' or discuss the system boundary.
- Acts as a link between business view (intentions for system) and IT.

Practical

- Acts as a checklist for the specification.
- Gives a first cut list for the use case diagram communications.
- *Of course the meat of the use case is in the description, which brings further considerations (back to dependencies).*

Embedding in MDA

- Need to move from process models (within CIM) to PIM (typically UML models such as class diagrams, activity diagrams).
- Model driven development implies transformation, so want rules to transform from RAD.
- As with the systems role approach, need to have a way of representing the system boundary (problem domain and machine).
- Novel approach is to have three variants of the RAD:
 - _ Environment RAD - without system interactions
 - _ Shared RAD - each role split into Environment and Machine (really just those bits within the system boundary - that we will have to build).
 - _ Machine RAD - Only shows those within system scope.
 - _ SHOW presentation of these briefly.
- Machine RAD is then transformed to PIM
- A set of rules (we will show in words, but these are encoded), which perform the transformation.

Transformation - Initial Rules

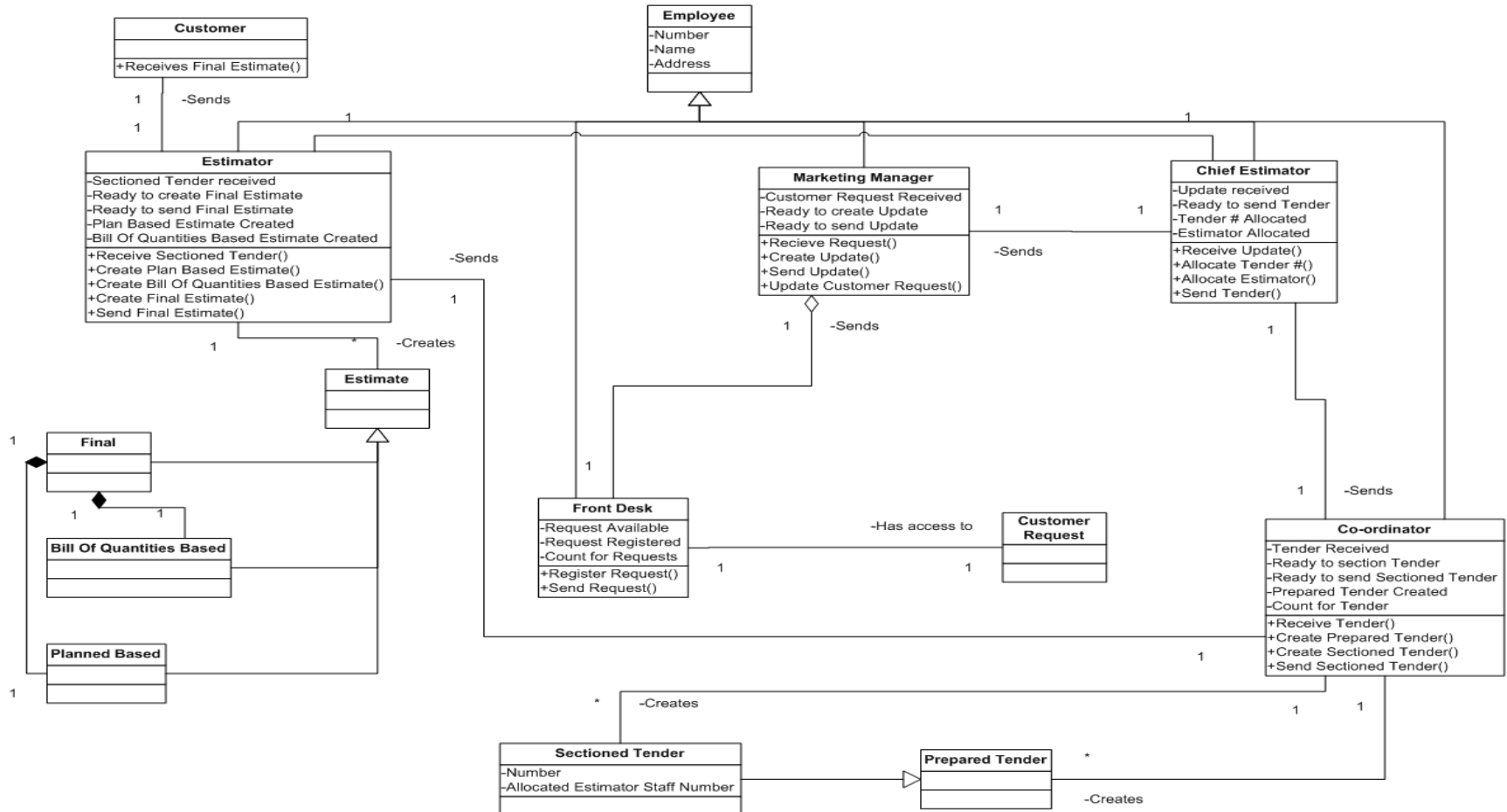
(1)

NATURAL ENGLISH	RAD	UML Class Diagram	UML Activity Diagram	UML Use Case	Example
Noun referring to human or system	Role	Class	Activity Partition	Actor; Relationship to Use Cases derived from that Role	Project Manager
Transitive verb with direct object (noun)	Independent Activity	Operation; Class & Association if Object results from Activity	Activity	Use Case; Note: Chunk of activity may define a single Use Case	Writes report
Clause where sequence is defined (before or after)	Looping, Line and Descriptor States	Attribute (only applicable for descriptor states)	Transition	Check context; May define "extend" Relationship	Ready to write report; Writes Report; Ready to send report; Sends Report;
Clause joined with "or" Conjunction	Case Refinement (Alternatives)	Attribute	Decision Diamond; Guard	Relationship	Write report or Delegate Task
Clause joined with "and" conjunction	Part Refinement (Concurrency)	Attribute; Composition if refinement objects are descendent from resulting object	Synchronisation Bar	"Include" relationship	Writes and Sends Report; Write report a) and report b) to be contained within Assessment Report
Sentence containing Role subject and object nouns with transitive verb; optionally modified by adverb	Interactions	Association; Operation in Role Classes. Aggregation if Role interactions are exclusive to only a single Role	Activity; Transition	Source and Destination Use Case; Relationship. Note: Chunk of activity may define a single Use Case	Project Manager sends Report to General Manager and Contractor Quickly

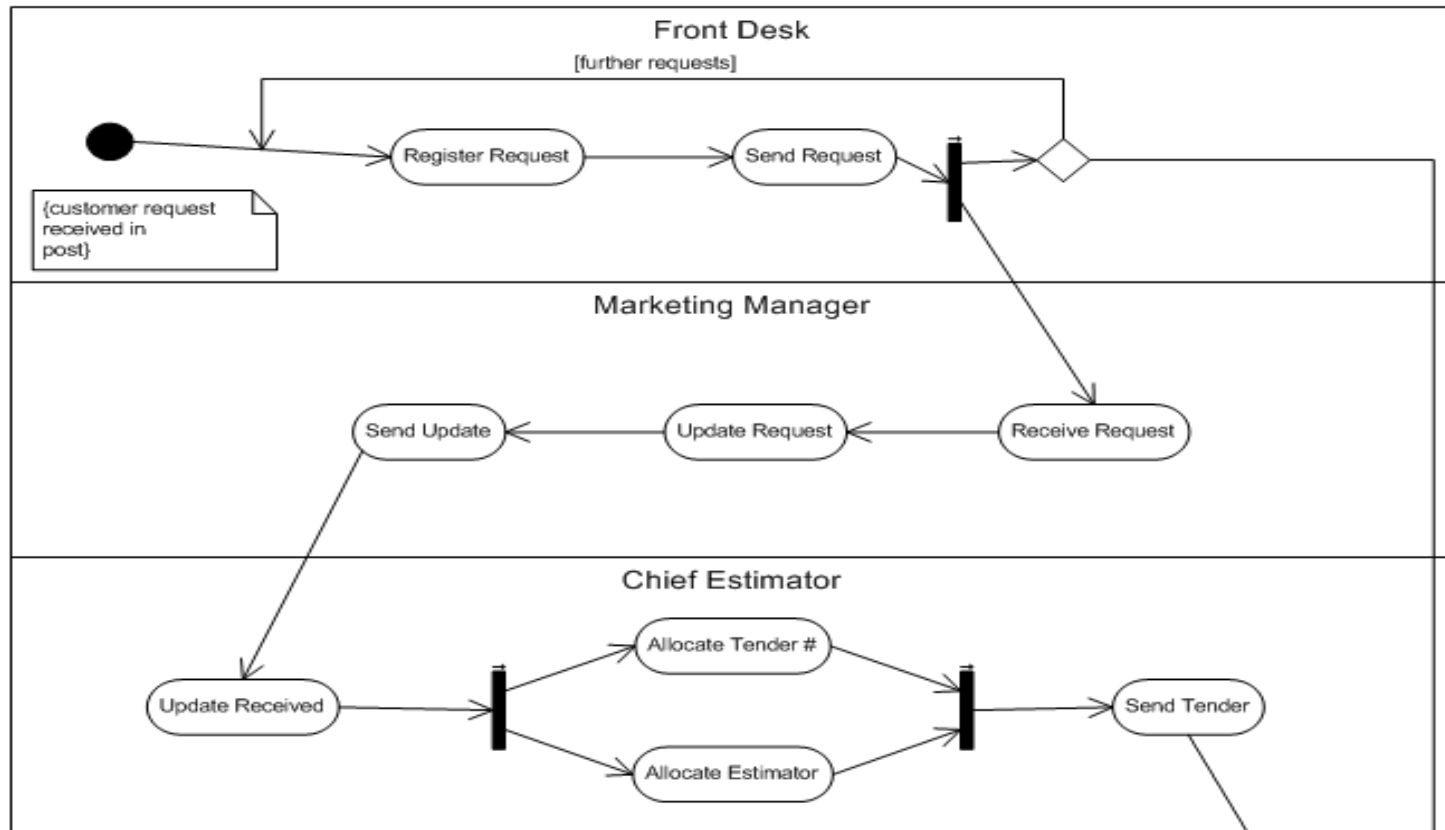
Transformation - Initial Rules (2)

Transitive verb with Role noun initiating new Role noun	Start Role	Role Class; Operation in Source Role Class; and Association	Activity Partition; Activity and transition	Actor	Project Manager selects Contractor
Noun referring to an event that starts a process	Trigger	Attribute	Start; Guard	Notes	Complaint is received
Quantifier associated with activity	Replication	Count attribute	Decision Diamond; Guard; Transition (loop) encapsulating replicated activity	Notes	For every application received, assess it
Where sequence is undefined before or after)	Undefined	Check context; May define alternate CD; Association	Check context; May define alternate AD; Transition; Stop	Check context; May define alternate UC; Relationship	The project manager writes report; The Project Manager owns a car
Determiner associated with Role noun	✓	Multiplicity	Notes	Multiplicity	There are 500 employees; The Project Manager
Transitive verb and noun consumed by Role noun	Props	Class	Notes	Check context; May define alternate Actor; Use Case; Notes	Uses database
Sequence terminating Transitive verb	Stop	Attribute in originating Role Class	Stop	Notes	Project Ends
Adjective modifying noun or Pronoun	Notes	Attribute	Notes	Notes	Project Manager is logged in

Transformation - Class Diagram



Transformation - Activity Diagram (1)



Strengths & Weaknesses

- Of course this is for your discussion, but some observations.
- POSD allows a complete mapping, and is very flexible. It does add another model, however, and, with discipline we could produce informal mappings.
- Augmented use cases add power, allow us to show process issues properly, and (as a bonus) support enactment. However, they are another addition (albeit small), so perhaps might be best used sparingly (as we said of states anyway).
- System boundary ID is vital. Simple, pragmatic approach appears useful (and easy to learn / explain).
- Embedded approach currently using three layer RAD, but general idea (RAD - via transformation - PIM), can be achieved in other ways too. Concept fits well with MDA mindset, but again modelling overhead (even with tool support).

Recap – So far.

- Reviewed analysis and requirements and found issues with methods.
- Business process models to understand better and inform specification.
- Described issues, perspectives and alternatives to help move from process model to specification.
- These also support (though not specifically to address) business and IT alignment.
- Still need to consider how we ensure the quality of our models for different purposes (e.g., to use the models to drive process improvement).