Surrounding the Requirements Process

Keith Phalp

ESERG Workshop on Software Engineering, September 2002
Bournemouth University
Surrounding the Requirements Process

Using Process Technology to Inform Requirements

Keith Phalp
Overview: A game of two halves

• Assuming a use case approach

• Two themes
  – Using process models (directly) to inform the use case. *(I know what about PDOA?)*
  – Using process modelling technology (ideas really) to support the use case.

• Consider rationale for each and progress (very early days)
Use Case based requirements

Questions

• Where does the UC description come from?
  – Previous documents elicitation notes / invention / domain analysis / process models.

• How do we improve the description?
  – Initial writing, analysis, revision and validation.

• How do we (best) support these activities?

• Where does the UC description go?
• Difficult to preserve mapping when notations are orthogonal.
  – Sometimes utilise further (structuring and overview) notations, such as POSD.
POSD Approach

Front Office
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Print Room Staff
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Print Room
- Access Packs
- Print Packs
- Print Room Staff
- Collate Packs

Word
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Printer
- Access Packs
- Print Packs
- Print Room Staff
- Collate Packs

Collect
- Print Packs
- Print Room Staff
- Collate Packs

Print
- Print Packs
- Print Room Staff
- Collate Packs

Collate
- Print Packs
- Print Room Staff
- Collate Packs

Front Office
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Print Room Staff
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Print
- Print Packs
- Print Room Staff
- Collate Packs

Collate
- Print Packs
- Print Room Staff
- Collate Packs

Printer
- Access Packs
- Print Packs
- Print Room Staff
- Collate Packs

Collect
- Print Packs
- Print Room Staff
- Collate Packs

Print Room
- Access Packs
- Print Packs
- Print Room Staff
- Collate Packs

Word
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Front Office
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Print Room Staff
- Access Packs
- Prepare for Word
- Print Packs
- Print Room Staff
- Collate Packs

Print
- Print Packs
- Print Room Staff
- Collate Packs

Collate
- Print Packs
- Print Room Staff
- Collate Packs
• Examine connections among roles.
  – Much activity within connections.

• Describe (or group) the genuinely ‘shared behaviours’.
  – Roles sharing sets of interactions are candidates for grouping.

• Group activities with ‘related’ content.
  – (both interactions and actions).
• Must preserve connections ‘promises’.
• Processes (or roles and actors) become use cases.

Guidance / lessons

• Roles often become actors.
• Reduce system roles.
• Reduce process mechanism.
• Beware inconsistent levels of abstraction.
  – E.g., single interactions OR
  – Multiple actions and interactions as a Use Case.

• Moving from process to specification.
  – Hence, some process elements may not be described.
Observations: Process so far

- RAD phase helps ‘debug’ process.
  - Checklist for activities in the use case description.
  - Describes *dependencies* among activities.
  - POSD provides guidance for use cases diagram.
  - POSD allows further scope for viewpoints.
  - Mapping helps ensure that detail is not omitted.
- Presumes process models and requires effort.
  - Though these are models of the application domain (isn’t that requirements).
  - Doesn’t bring in other opportunities (e.g., frames).
  - Time a problem for industrial application.
Next Steps: Use Cases onwards

• Have moved from process description to use case diagram.
• Use cases help identify packages.
• Each Use case has associated description.
• From descriptions we discover objects.
  – By asking sets of questions.
  – By refining descriptions.
  – By considering dependencies
  – By running models{enaction}. 

What (else) is wrong with use case description anyway

- No details about the dependencies of actions.
- Consider a generic use case (CP rules)
  - SubjectA verb1 ObjectX
  - SubjectB verb2 ObjectY
  - SubjectC verb3 ObjectZ
- Under what circumstances does verb3 occur?
  - Dependent on verb2 or verb1 or neither?
  - Danger of assumptions?
- Importance of validation and domain knowledge.
Two sporting use cases

1. The match reached full-time
2. The ball crossed the goal-line
3. The referee blew his/her whistle
4. The goal was given

Alternatives
4. The goal was not given

Alternatives
4. The goal was not given

Validation & Context. Someone who ‘knows the the game’.
Main flow of events:
1. The Driver drives to the ticket machine.
2. The Driver presses the ticket button.
3. The ticket machine dispenses a ticket.
4. The Driver takes the ticket.
5. The entry barrier raises.
6. The Driver drives into the car park.
7. The entry barrier lowers.
8. The Driver parks the car.

Exceptional flow of events:
3. The ticket machine fails to dispense a ticket. The Driver calls for assistance.

Use Case 1: Enter Car Park

- **Actors:** Driver
- **Context:** The Driver wants to park in the local “Regional Car Park” so the Driver can go shopping.
- **Pre-condition:** There are parking spaces available inside the car park.
  - *(How do we know?)*
- **Post condition:** There is one less space available inside the car park.
Selection Driver.driveOverPad
   Me( initial -> DriverAtMachine )
   EntryPad( initial -> overPad )
End

Selection EntryPad.PadNotify
   Me( overPad -> initial )
   TicketMachine( initial -> CarAtMachine)
End

Selection Driver.PressForTicket
   Me( DriverAtMachine -> ticketRequested )
   TicketMachine( CarAtMachine -> ticketRequested )
End

Selection TicketMachineDispense
   Me( ticketRequest -> ticketDispensed )
   Ticket( initial -> date_stamped )
End

Interaction Driver.TakeTicket
   Me( ticketRequested -> ticketTaken)
   TicketMachine( ticketDispensed -> ticketTaken )

Object States:
  Formal

- Dependencies for 1 to 4.
- States act as pre / post conditions.
  E.g., for driver to take ticket it must have been dispensed.
- Ticket not from behaviour, but a data object.
  and so on...
• Write ‘sunny day’ scenario.
  – Add alternatives or exceptions.
  – Add actors when required.
• ‘Step through’ the use case.
  – Generated from description.
• Provide (some) guidance.
• Other opportunities
  – organise, link & synchronize multiple use cases, provide measures & estimates…
Where now?

• Developing (and extending) tool support for UC descriptions.
• Formalise mapping ideas.
• UC tool (and enaction) suggests need at process stage for better tool support:
  – Automatic enaction for validation
  – Bundling behaviours
    • (or moving straight to design). \textit{Process Oriented Systems Design}.
  – Support for use case generation
• Round in circles again.
Spare Slides
Opportunities

• Portions of business process models may map to subsequent documents?
  – RADs to use cases (larger scale) - using POSD.
  – Avoid use cases (RADs to interface / design).

• Business process technology (e.g., state-based, enaction) may be useful in supporting Use Case description.

• Use cases could be ‘interrogated’ to provide information to subsequent phases.
Enactable Use case for validation

• Process modelling experience:
  – State based approaches allow consideration of dependencies.
  – Annotate models with states.
  – Step through states with computer models.

• Hence, used this approach to produce enactable equivalents of use cases.
  – RolEnact (equivalent) to use case descriptions.

• Note original (RolEnact) + prototype tool.
Experience of Producing RolEnact equivalent to use cases

- Student subjects: SDM + Integrating Studies programmes.
  - Students coped relatively easily with language.
  - Aided validation. Increased understanding, clarified issues.
  - Also (bonus) teased out design issues (post-UCD).
- Projects (RolEnact): legal system, record & billing system.
- Significant overhead. Too time-consuming.
- Need for tool support. (May incorporate other ideas too).