

Reviewing UML as Part of the Project Team

Software Futures Ltd

2 Waterloo Way

Bredon

Tewkesbury

Glos.

GL20 7NA

GB

p +44 (0) 1684 772 691

f +44 (0) 1684 772 639

e richard@softwarefutures.ltd.uk

w www.softwarefutures.ltd.uk

Topics

- Why review UML?
- How to review UML?
- Experiences and lessons learnt.

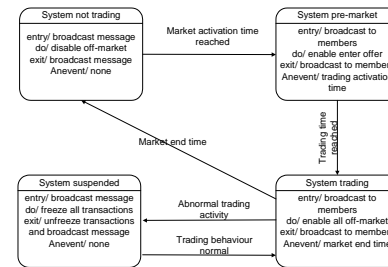
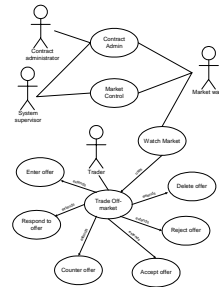
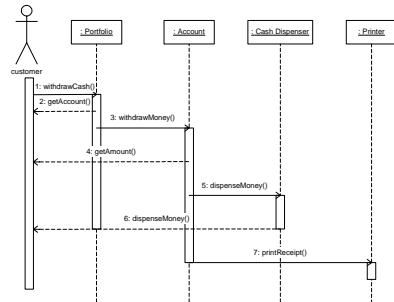
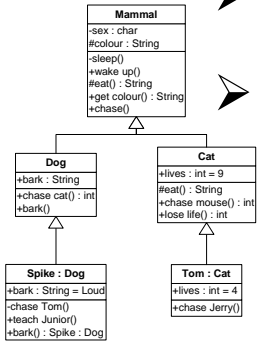
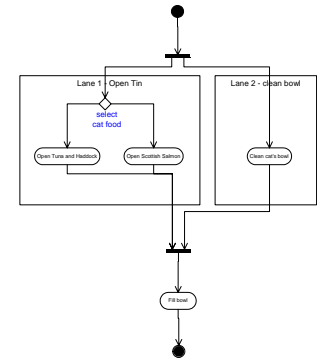
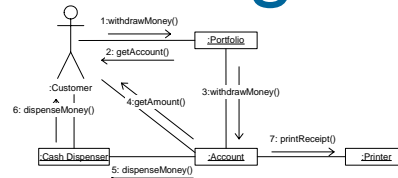
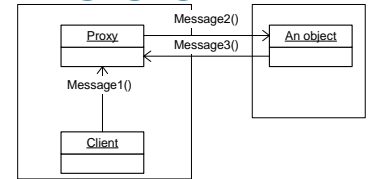
Why review UML?

- We know reviewing is good thing.
- UML is new and we have much to learn.
- Adopting UML creates new risks.
- Reviewing gets us working early with the analysts.

Unified Modelling Language

- The UML 1.4 specification is over 550 pages long.
- Four layer metamodelling architecture.
- Judgements:

- Which models do we use and where?
 - What rules and standards do we need?
- Where do we get the wisdom from?*



UML Testing

- Read and understand UML.
- Identify good from bad UML.
- Use it to design effective tests.
- Use it to document tests.
- Judgements are the real challenge.

Model-based development

- For functional behaviour.

- *Use Case model – user requirements and analysis.*
- *Class diagram – class specifications & relationships.*
- *Sequence diagram – time-based communication.*
- *Collaboration diagram – messaging between objects.*
- *Statechart diagram – states and events of objects.*
- *Activity diagram – actions, links and control flow.*
- *Component diagram – software components for building.*
- *Deployment/package diagrams – distribution architecture.*

Risk and UML

- Is UML a risk in itself?
 - yes if not managed properly.
 - like any technology it depends on the people.

Risk and UML

- How does UML reduce risks of error?
 - models many forms of behaviour.
 - can check between models.
 - consistency through the lifecycle.
 - tools supported processes.
 - supports incremental/iterative development.
- How does UML increase risks?
 - Will not make poor analysts good ones.
 - Will not cure project or test management problems.

How to review UML?

- Do not second guess the analysts.
- Develop testability criteria with their assistance.

Defining testability

- Robert Binder (Testing Object Oriented Systems (2000)).
 - allows automatic generation of test cases.
 - models all features to be tested.
 - preserved detail essential for revealing faults and demonstrating conformance.
 - represents all actions.
 - defines state so that state checking can be automated.

Defining testability

- ISO 9126.

- Testability is a set of attributes for assessing the amount of design and implemented autonomous test aid functions present in the software product.

Testability review criteria

- Is the diagram understandable with syntactic and semantic conformance to UML?
- Does the diagram represent all the behaviours and properties in the user requirements? Are other diagrams needed?

Testability review criteria

- Do the models contain any properties that make testing unnecessarily difficult or impossible?
- The model should adhere to design heuristics and exceptions should be documented.
- Non-functional objectives should not be overlooked.

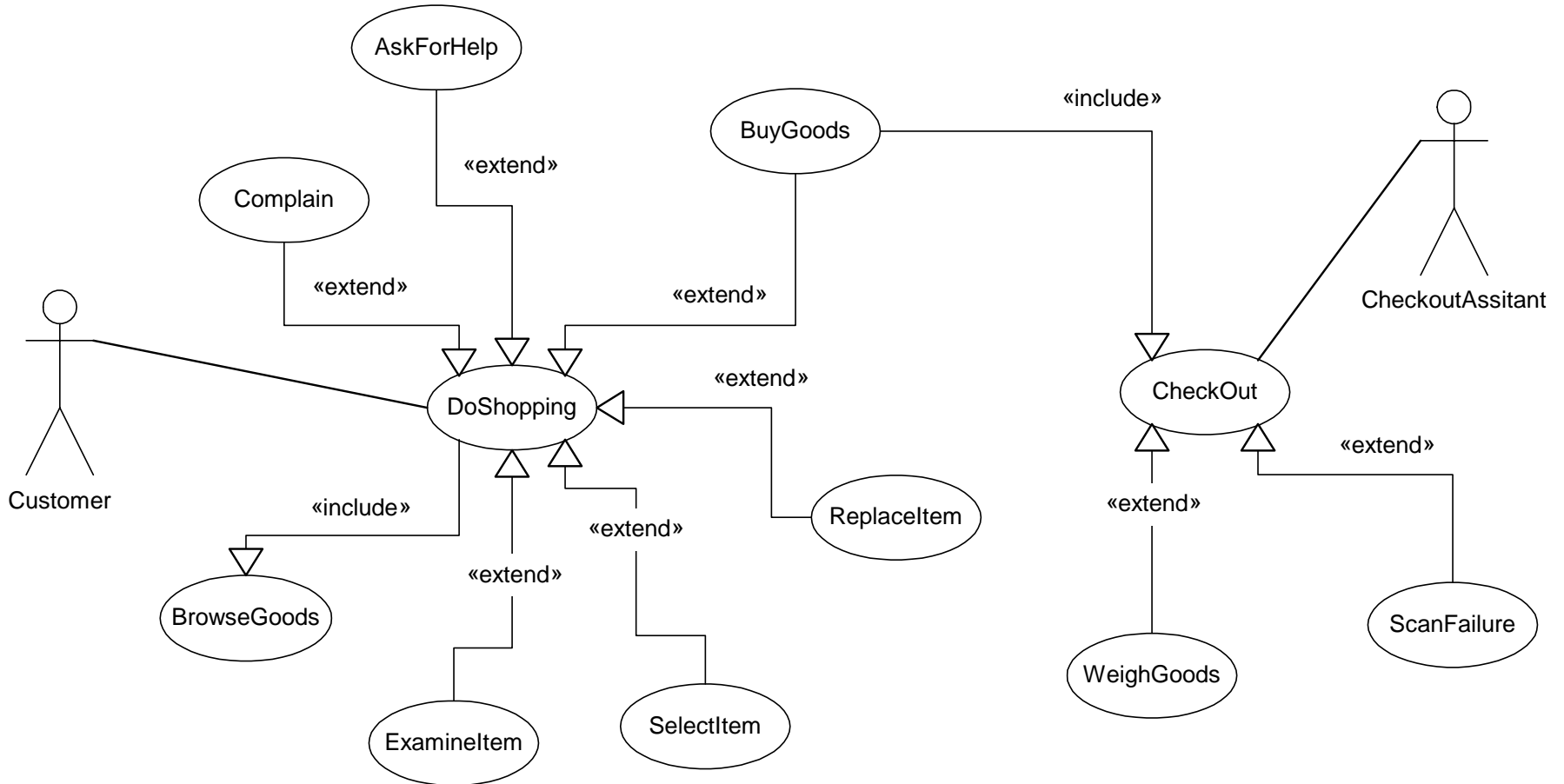
Testability review criteria

- Are the models logically consistent with each other?

What is a heuristic?

- It is a rule or guide we learn from experience or by deduction.
- A heuristic is a little bit of wisdom.
- Heuristics can be broken.
 - we identify when we do this.
 - we know why we do this.
- How to apply heuristics and testability criteria?

Simple Use Case model



But how well is it designed?

- Use Case model.
 - Decomposition into Use Cases, size, complexity and detail. .
 - Relationships between Use Cases, extend and include, control flow complexity.
- Poor design is a major threat.
 - Poor functional cohesion.
 - Overly complex relationships.
 - Contains design/implementation detail.

Applying cohesion rules

Problems to look for:

- Monolithic Use Cases
- Use Cases that are “small” verbs
- Too many Use Cases
- Controller Use Cases
- Imbalance of abstraction
- Attribute-based Use Cases
- Incorrect relationships
- Unstable, non-extensible
- Actors that are passive

Observations:

- *too much behaviour in one place*
- *single piece of behaviour*
- *lack of abstraction*
- *behaviour is “used”*
- *analysis/design cross-over*
- *not function-based*
- *uses and extends confused*
- *bad assumptions*
- *do not trigger behaviour*

Experiences of reviewing UML

- The outsourced Use Case modelling.
 - Analysts highly error-prone.
 - Very poor processing flows.
 - No use of Activity diagrams.
- In-house trading system.
 - Good Use Cases.
 - Much state behaviour implied but no Statecharts.
 - Complex data needed better modelling.

More experiences

- The curious processing flows.
 - Primary flow contained no decisions.
 - Every decision, however small, documented in its own table.
 - Extremely difficult to follow control flows.
- Other lessons.....

ISO 9126 Quality Characteristics

● Reliability

- Maturity
- Fault tolerance
- Recoverability

● Usability

- Understandability
- Learnability
- Operability
- Attractiveness

● Efficiency

- Time behaviour
- Resource utilisation

● Maintainability.

- Analysability.
- Changeability.
- Stability.
- Testability.

● Portability.

- Adaptability.
- Installability.
- Co-existence.
- Replaceability.

Implications for Testing

- UML is a very rich language and can lead to very rich, indigestible test situations. It can lead to complex class structures with complex dynamic behaviour.
- UML does not reduce the need for testing. It creates a wide range of new and at times subtle errors if the rules are broken, or UML is misapplied or incomplete.

Implications for Testing

- The first step is to make sure the rules are not broken; the purpose of reviews.
- We have to apply existing, traditional, methods as well develop new ones.
- Above all testers need to adopt a strategy dependent on the way the developers will design and build an application.

Implications for Testing - reviews

- Effective providing well done.
- Planned and managed.
- Use and develop UML heuristics and checklists.
- Involve developers, analysts and users as well as testers.
- Review as early as practical.

Implications for Testing - reviews

- Consider use of an Oracle.
- Review incrementally.
- Phase in starting with key documents.
- Select appropriate review technique and formality/ceremony.
- Avoid the A-Z trap; running out of time halfway through.

Software Futures

Assured solutions for tomorrow