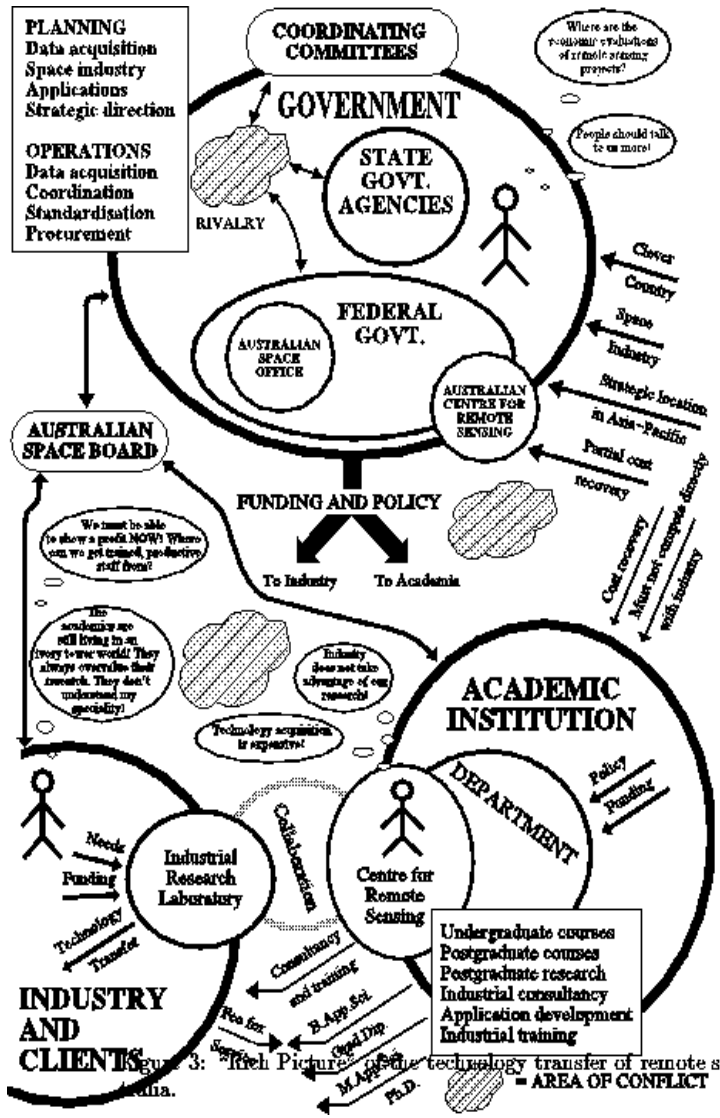


Figure 3: Rich Picture of the technology transfer of remote sensing in Australia.

Software Development

Process, Models,
Methods,
Diagrams
Software Development
Life Cyles

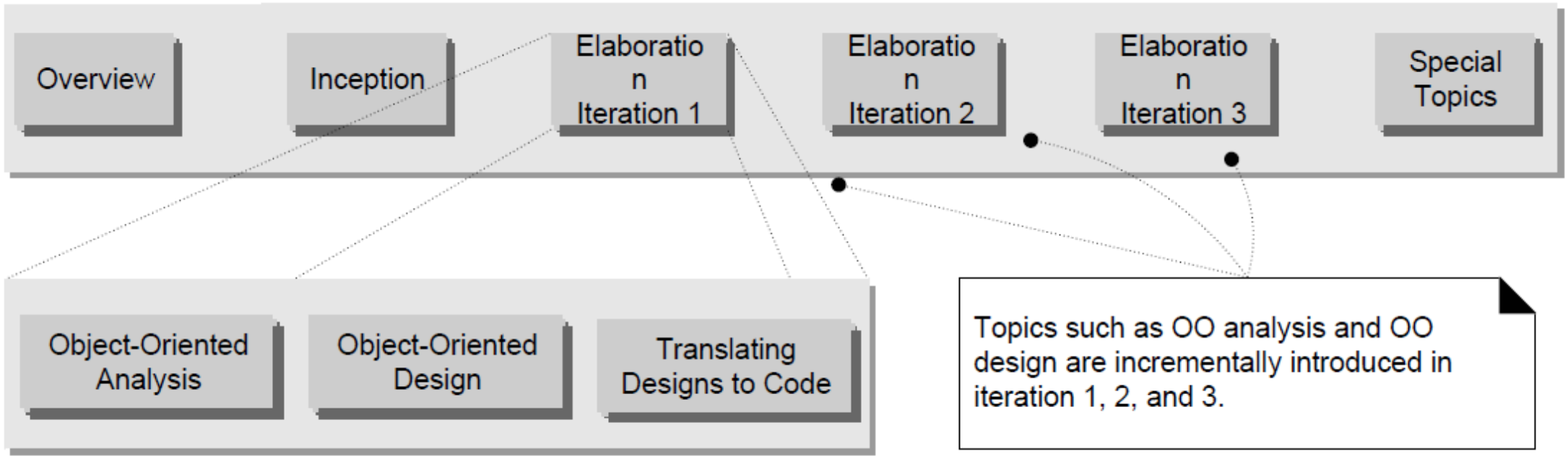
Part - III

As a Reminder

Discipline	Artifact Iteration-*	Incep. 11	Elab. El. .En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		
	Data Model		s	r	
Implementation	Implementation Model		s	r	r
Project Management	SW Development Plan	s	r	r	r
Testing	Test Model		s	r	
Environment	Development Case	s	r		

Sample Development Case of UP artifacts, s - start; r - refine

Sample Agile based SP



Re-Inception: Artifacts

Artifact[1]	Comment
Vision and Business Case	Describes the high-level goals and constraints, the business case, and provides an executive summary.
Use-Case Model	Describes the functional requirements, and related non-functional requirements.
Supplementary Specification	Describes other requirements.
Glossary	Key domain terminology.
Risk List & Risk Management Plan	Describes the business, technical, resource, schedule risks, and ideas for their mitigation or response.
Prototypes and proof-of-concepts	To clarify the vision, and validate technical ideas.
Iteration Plan	Describes what to do in the first elaboration iteration.
Phase Plan & Software Development Plan	Low-precision guess for elaboration phase duration and effort. Tools, people, education, and other resources.
Development Case	A description of the customized UP steps and artifacts for this project. In the UP, one always customizes it for the project.

Understanding Requirments

- Requirements are capabilities and conditions to which the system—and more broadly, the project—must conform
- Req. Managment: a systematic approach to finding, documenting, organizing, and tracking the changing requirements of a system

FURPS+ Requirement Model

- Functional - features, capabilities, security.
- Usability - human factors, help, documentation.
- Reliability - frequency of failure, recoverability, predictability.
- Performance - response times, throughput, accuracy, availability,
- resource usage.
- Supportability - adaptability, maintainability, internationalization,
- configurability.
- The “+” means
  - Implementation - resource limitations, languages and tools, hardware..
  - Interface - constraints imposed by interfacing with external systems.
  - Operations - system management in its operational setting.
  - Packaging - for example, a physical box.
  - Legal - licensing and so forth.

In common usage, requirements are categorized as functional (behavioral) or non-functional (everything else); some dislike this broad generalization, but it is very widely used.

Use-Case Model: Writing Requirements In Context

- The idea of use cases to describe functional requirements was introduced in 1986 by Ivar Jacobson
- Alistair Cockburn, in 1992, defined what use cases are (or should be)
- A scenario is a specific sequence of actions and interactions between actors and the system under discussion; it is also called a use case instance
- A use case is a collection of related success and failure scenarios that describe actors using a system to support a goal.

Use Case Analysis

- What is a Use Case?
  - A sequence of actions a system performs that yields a valuable result for a particular actor.
- What is an Actor?
  - A user or outside system that interacts with the system being designed in order to obtain some value from that interaction
- Use Cases describe scenarios that describe the interaction between users of the system and the system itself.
- Use Cases describe WHAT the system will do, but never HOW it will be done.

Usecase presentation formats

- brief—terse one-paragraph summary, usually of the main success scenario.
- casual—informal paragraph format. Multiple paragraphs that cover vari ous scenarios
- fully dressed—the most elaborate. All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees.

What's in a Use Case?

- Define the start state and any preconditions that accompany it
- Define when the Use Case starts
- Define the order of activity in the Main Flow of Events
- Define any Alternative Flows of Events
- Define any Exceptional Flows of Events
- Define any Post Conditions and the end state
- Mention any design issues as an appendix
- Accompanying diagrams: State, Activity, Sequence Diagrams
- View of Participating Objects (relevant Analysis Model Classes)
- Logical View: A View of the Actors involved with this Use Case, and any Use Cases used or extended by this Use Case

Lets see sample

Usecase Level

- Elementary Business Usecase: A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state
- A common use case mistake is defining many use cases at too low a level; that is, as a single step, subfunction, or subtask within an EBP
- Goals, namely usecases, are usually composite, from the level of an enterprise ("be profitable"), to many supporting intermediate goals while using applications ("sales are captured"), to supporting subfunction goals within applications ("input is valid"). Similarly, use cases can be written at different levels to satisfy these goals, and can be composed of lower level use cases

Finding Usecase, Goal, Actor

Use cases are defined to satisfy the user goals of the primary actors. Hence, the basic procedure is:

1. Choose the system boundary. Is it just a software application, the hardware and application as a unit, that plus a person using it, or an entire organization? Once the external actors are identified, the boundary becomes clearer.

2. Identify the primary actors those that have user goals fulfilled through using services of the system

3. For each, identify their user goals. Raise them to the highest user goal level that satisfies the EBP guideline

4. Define use cases that satisfy user goals; name them according to their goal. Usually, user goal-level use cases will be one-to-one with user goals, but there may be exceptional situation.

Actor-Goal List

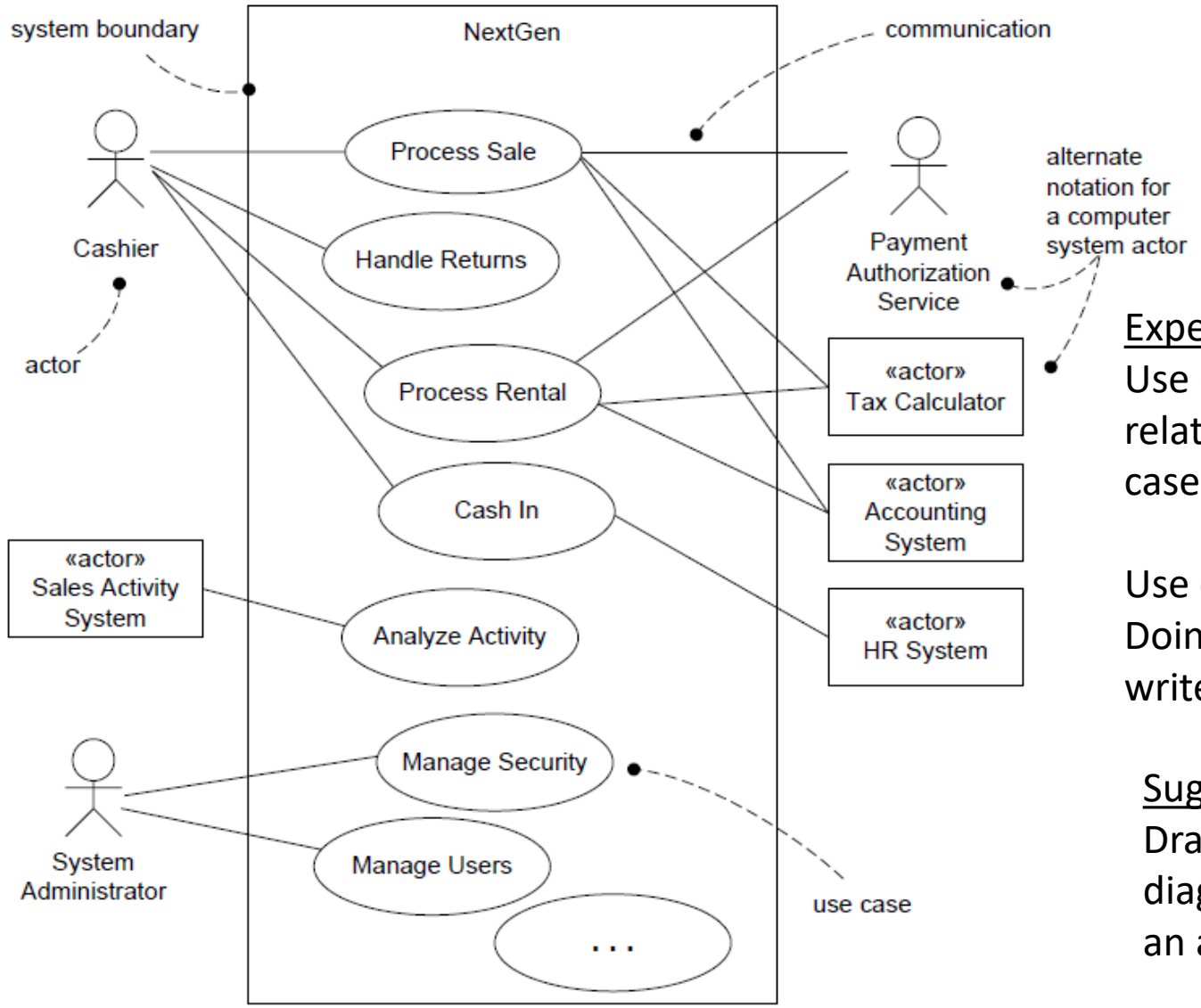
Actor	Goal	Actor	Goal
Cashier	process sales process rentals handle returns cash in cash out …	System Administrator	add users modify users delete users manage security manage system tables …
Manager	start up shut down …	Sales Activity System	analyze sales and performance data
…	…	…	…

Actor

An actor is anything with behavior, including the system under discussion (SuD) itself when it calls upon the services of other systems

- Primary Actor: has user goals fulfilled through using services of the SuD
- Supporting Actor: provides a service (for example, information) to the SuD
- Offstage Actor: has an interest in the behavior of the use case, but is not primary or supporting

Usecase Diagram



Expected

Use case diagrams and use case relationships are secondary in use case work.

Use cases are text documents. Doing use case work means to write text.

Suggestion

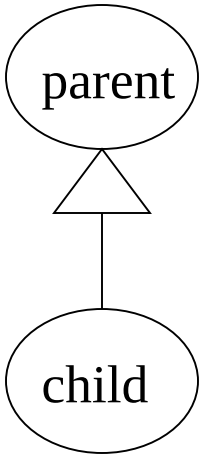
Draw a simple use case diagram in conjunction with an actor-goal list.

Relationships between Use Cases

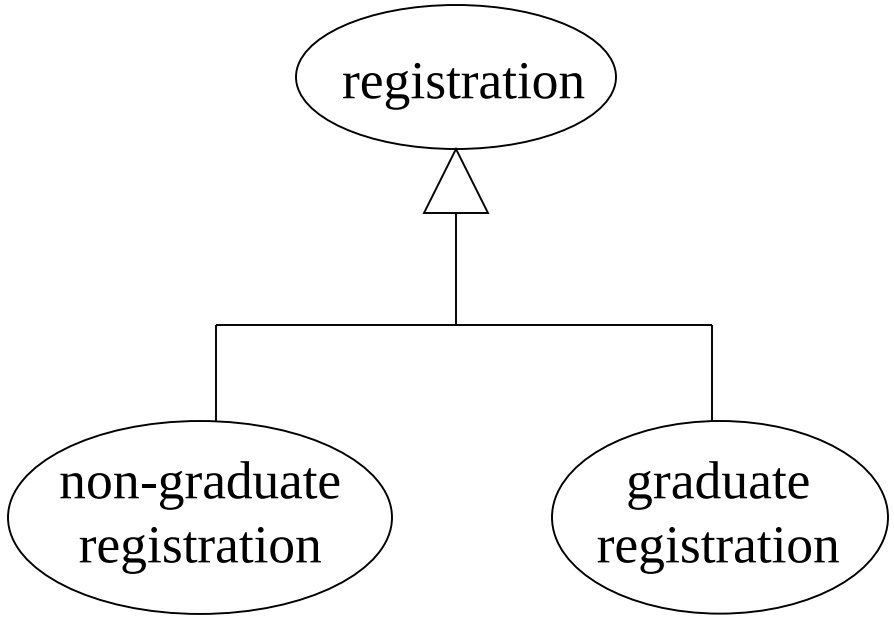
1. Generalization - use cases that are specialized versions of other use cases.
2. Include - use cases that are included as parts of other use cases. Enable to factor common behavior.
3. Extend - use cases that extend the behavior of other core use cases. Enable to factor variants.

1. Generalization

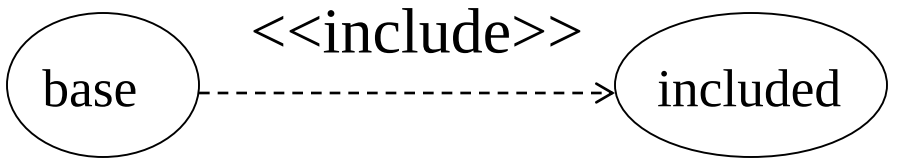
- The child use case inherits the behavior and meaning of the parent use case.
- The child may add to or override the behavior of its parent.



More about Generalization



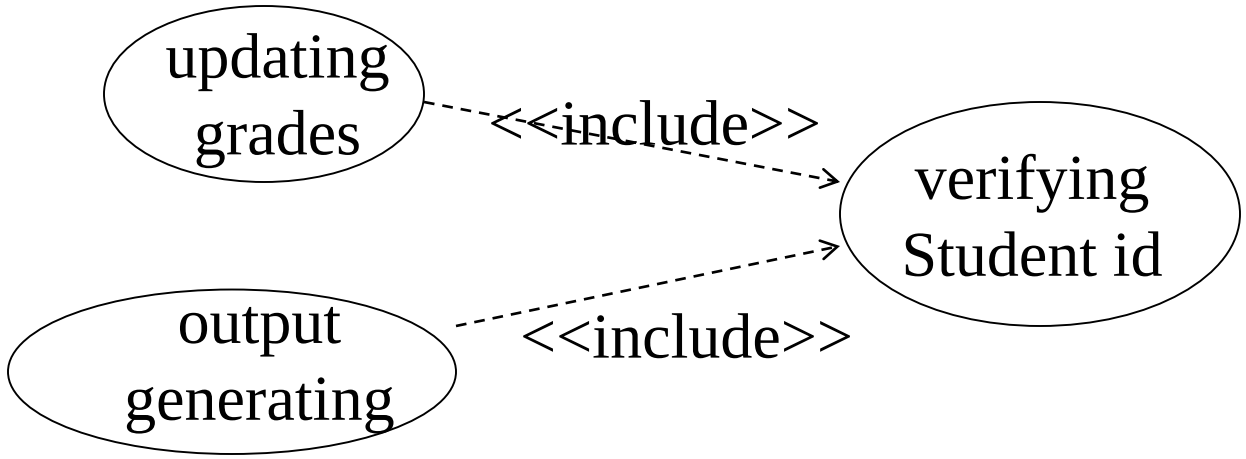
2. Include



- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never or sometimes stands alone. It only occurs as a part of some larger base that includes it.

More about Include

- Enables to avoid describing the same flow of events several times by putting the common behavior in a use case of its own.

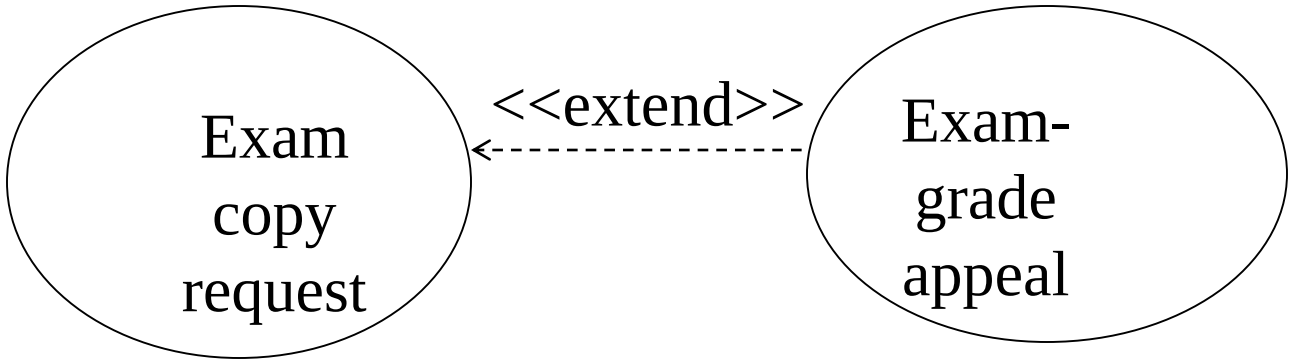


3. Extend

- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.

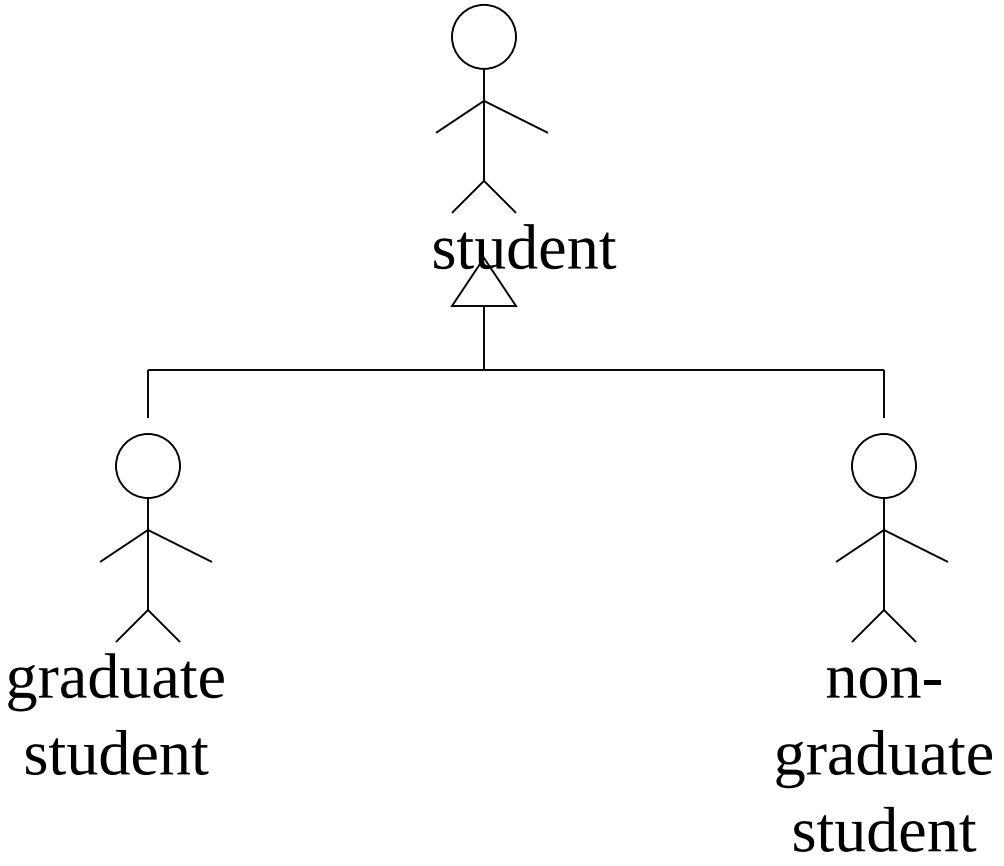
More about Extend

- Enables to model optional behavior or branching under conditions.



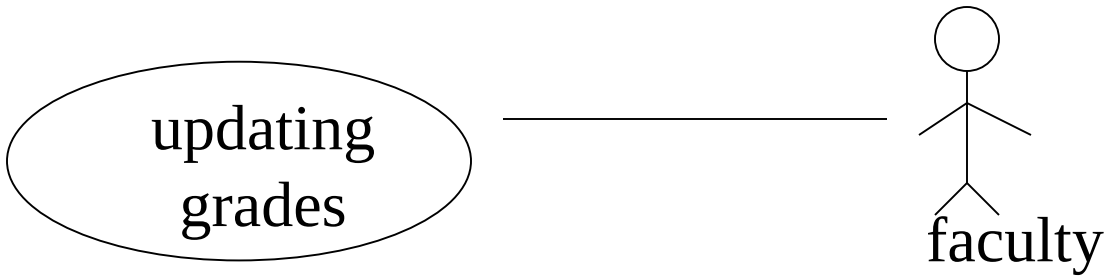
Relationships between Actors

- Generalization.



Relationships between Use Cases and Actors

- Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.



It is not a recipe, but good sample

Discipline	Artifact	Comments and Level of Requirements Effort				
		Incep 1 week	Elab 1 4 weeks	Elab 2 4 weeks	Elab 3 3 weeks	Elab 4 3 weeks
Requirements	Use-Case Model	2-day requirements workshop. Most use cases identified by name, and summarized in a short paragraph. Only *10%* written in detail.	Near the end of this iteration, host a 2-day requirements workshop. Obtain insight and feedback from the implementation work, then complete 30% of the use cases in detail.	Near the end of this iteration, host a 2-day requirements workshop. Obtain insight and feedback from the implementation work, then complete 50% of the use cases in detail.	Repeat, complete 707 of all use cases in detail.	Repeal with the goal of 80-90% of the use cases clarified and written in detail. Only a small portion of these have been built in elaboration; the remainder are done in construction.
Design	Design Model	none	Design for a small set of high-risk architecturally significant requirements.	repeat	repeat	Repeat. The high risk and architecturally significant aspects should now be stabilized.
Implementation	Implementation Model (code, etc.)	none	Implement these.	Repeat. 5% of the final system is built.	Repeat. 10% of the final system is built.	Repeat. 15% of the final system is built.
Project Management	SW Development Plan	Very vague estimate of total effort.	Estimate starts to take shape.	a little better...	a little bettor...	Overall project duration, major milestones, effort, and cost estimates can now be rationally committed to.

Usecases in Elaboration: Tag Req.

- Risk includes both technical complexity and other factors, such as uncertainty of effort or usability
- Coverage implies that all major parts of the system are at least touched on in early iteration, perhaps a "wide and shallow" implementation across many components
- Criticality refers to functions of high business value

Rank	Requirement (Use Case or Feature)	Comment
High	Process Sale Logging …	Scores high on all ranking criteria. Pervasive. Hard to add late. …
Medium	Maintain Users …	Affects security subdomain. …
Low	…	…

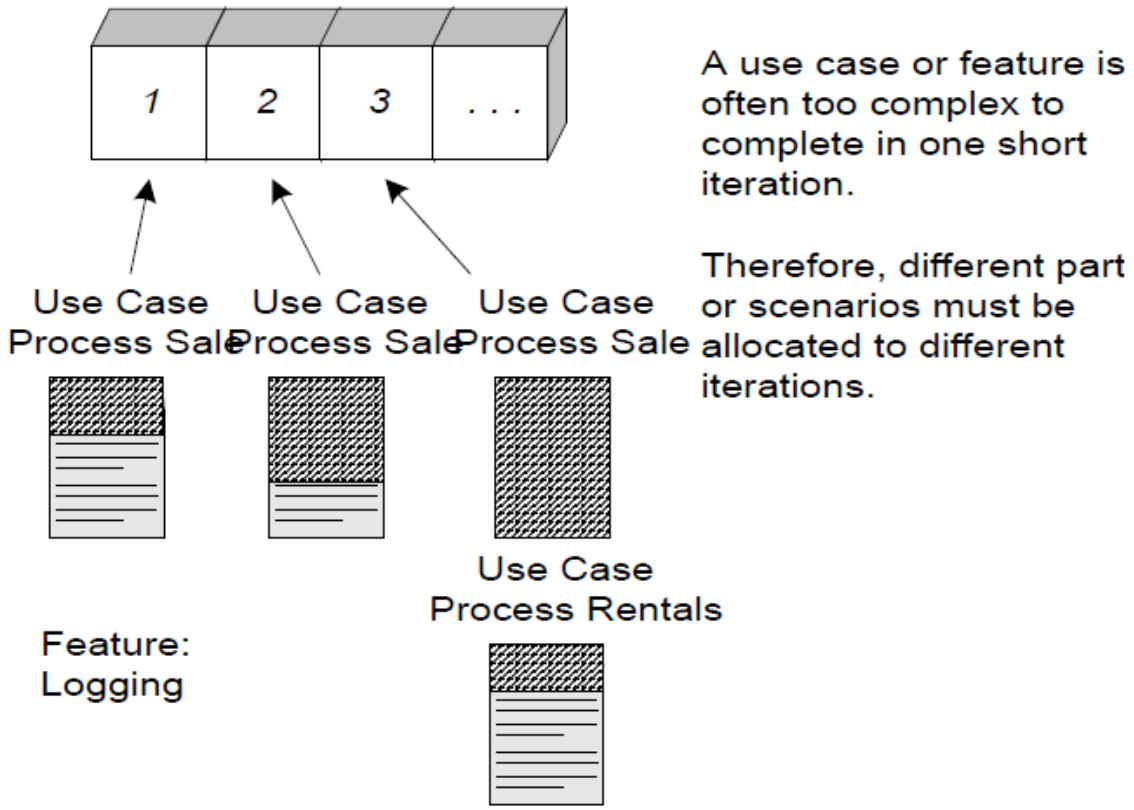
Sample Start-Up Iteration Planning

Iteration 1 Requirements : The requirements for the first iteration of the NextGen POS application follow:

- Implement a basic, key scenario of the *Process Sale* use case: entering items and receiving a cash payment.
- Implement a *Start Up* use case as necessary to support the initialization needs of the iteration.
- Nothing fancy or complex is handled, just a simple happy path scenario, and the design and implementation to support it.
- There is no collaboration with external services, such as a tax calculator or product database.
- No complex pricing rules are applied.
- The design and implementation of the supporting UI would also be done, but is not covered.

Subsequent iterations will grow on this foundation.

Incremental Development for the Same Usecase Across the Iterations



A use case or feature is often too complex to complete in one short iteration.

Therefore, different part or scenarios must be allocated to different iterations.

Use case implementation may be spread across iterations.

You Know You Didn't Understand Elaboration When...

- It is more than "a few" months long for most projects.
- It only has one iteration (with rare exceptions for well-understood problems)
- Most requirements were defined before elaboration.
- The risky elements and core architecture are not being tackled.
- It does not result in an executable architecture; there is no production-code programming.
- It is considered primarily a requirements phase, preceding an implementation phase in construction.
- There is an attempt to do a full and careful design before programming.
- There is minimal feedback and adaptation; users are not continually engaged in evaluation and feedback
- There is no early and realistic testing.
- The architecture is speculatively finalized before programming.
- It is considered a step to do the proof-of-concept programming, rather than programming the production core executable architecture.
- There are not multiple short requirements workshops that adapt and refine the requirements based on feedback from the prior and current iterations