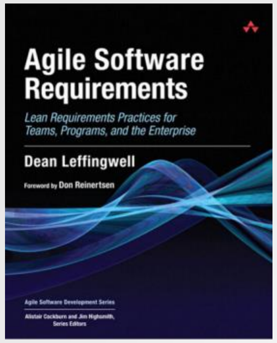

Agile Software Requirements
Lean Requirements Practices for Teams, Programs, and the Enterprise
Dean Leffingwell
Foreword by Don Reinertsen
Agile Software Development Series
Alistair Cockburn and Jim Highsmith, Series Editors

Agile Software Requirements

Software Requirements Engineering – 40688

Computer Engineering department

Sharif university of technology

Fall 402

Chapter 19:

Use cases

The Problems with user stories and Backlog items (1)

- Alistair Cockburn is one agile thought leader with his foot in both of these camps.
- He bemoans the apparent loss of use cases from agile development.
- User stories and backlog items don't give the designers a context to work from.
- When is the user doing this, what is the context of their operation, and what is their larger goal at this moment?
- User stories and backlog items don't give the project team any sense of scope or potential "completeness".

The Problems with user stories and Backlog items (2)

- A development team estimates a project at (e.g.) 270 story points, and then as soon as they start working, that number keeps increasing, seemingly without bound.
- The developers and sponsors are equally depressed. How big is this project, really?
- User stories and backlog items don't provide a mechanism for looking ahead at upcoming work.
- With user stories, the extension conditions are usually detected mid-sprint, when it is too late.

5 good reasons to still use use cases (1)

- The list of goal names provides executives with a short summary of what the system will contribute to the business and the users.
- The main success scenario of the use case provides everyone with an agreement as to what the system will and will not do.
- The extension conditions of the use case provide a framework for investigating all the little, niggling things that somehow take up 80% of the development time and budget.

5 good reasons to still use use cases (2)

- The use case extension scenarios provide answers to the many detailed, tricky business questions programmers ask: “What are we supposed to do in this case?”
- The full use case set (use case model) shows that the developers/analysts have thought through every user’s needs, every goal they have with respect to the system, and every business variant involved.

Use Case Basics (1)

A use case describes a sequence of actions between an actor and a system that produces a result of value for that actor.

- **Sequence of actions:**
  - The sequence of actions describes a set of interactions between the actor and the system.
- **System:**
  - The system works for the actor. It executes some function, algorithmic procedure, or other activity.

Use Case Basics (2)

- **A result of value**
  - Like a user story, the use case must deliver value to a user.
- **Actor**
  - The particular actor is the individual or device (Mark, the resident; a message from the utility) that initiates the action.

Use Case Actors

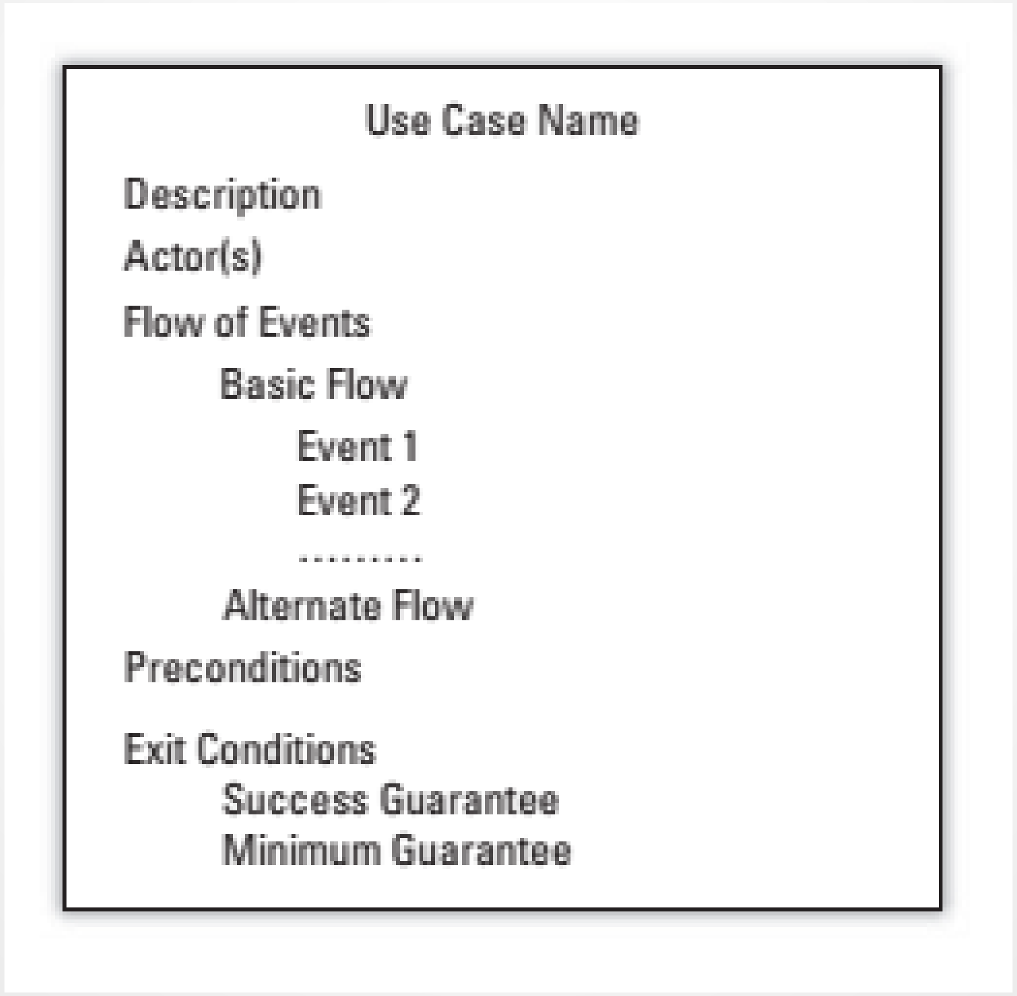
- **An actor** is someone or something that interacts with the system.
- **Users**: Users act on the system
- **Other systems or applications**: Most software interacts with other systems or other applications.
- **A device**: Many applications interface to a variety of input and output devices.

Use Case Structure

A use case has four mandatory elements.

1. **Name**: The name describes the goal, that is, what is achieved by the interaction with the actor.
2. **Brief description:** The purpose of the use case should be described in one or two sentences.
3. **Actor(s):** A use case has no meaning outside the context of its use by an actor.
4. **Flow of events:** The main body of the use case is the event flow, usually a textual description of the interactions between the actor and the system.

Use case template



Use Case Name

Description

Actor(s)

Flow of Events

    Basic Flow

        Event 1

        Event 2

        ....

    Alternate Flow

Preconditions

Exit Conditions

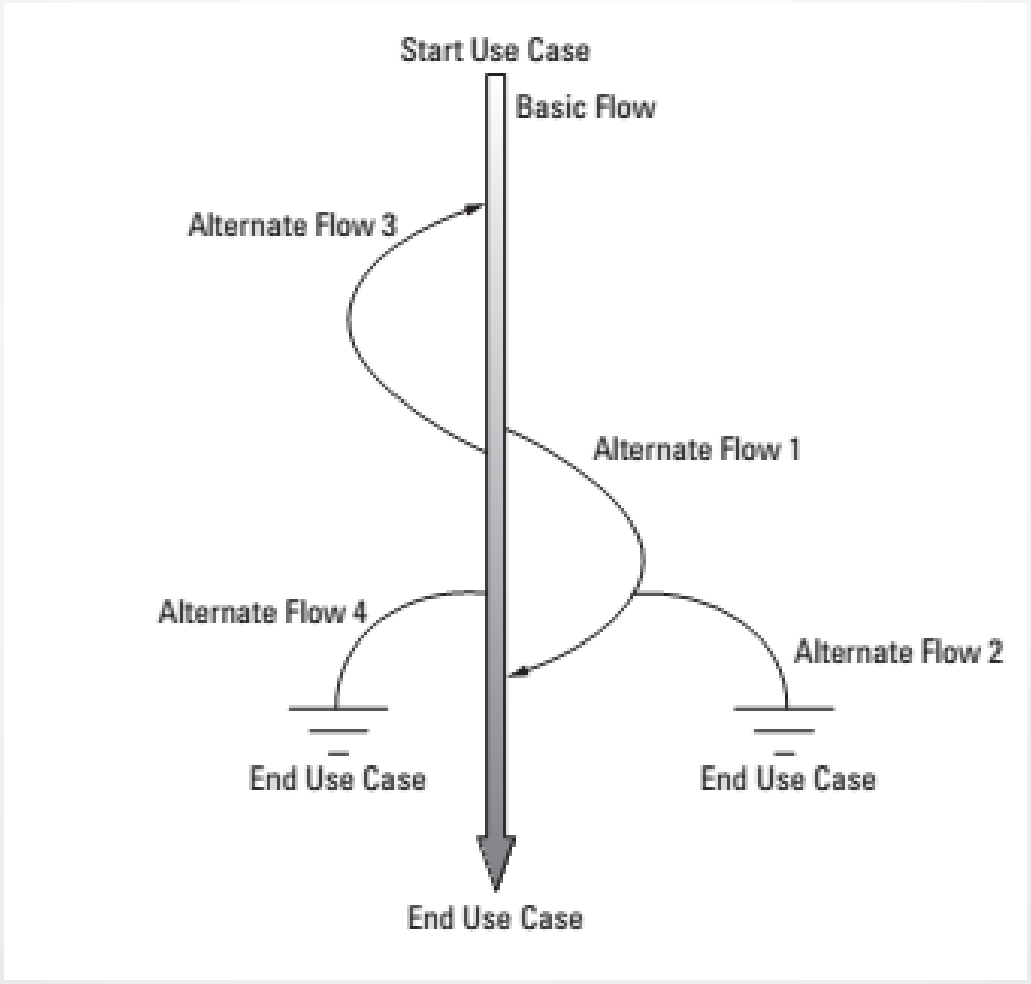
    Success Guarantee

    Minimum Guarantee

A Step-by-Step Guide to Building the Use Case Model

- Step **1**: Identify and Describe the Actors
- Step **2**: Identify the Use Cases.
- Step **3**: Identify the Actor and Use Case Relationships.
- Step **4**: Outline the Flow of the Use Cases.
- Step **5**: Refine the Use Cases

Outline the Flow of the Use Cases



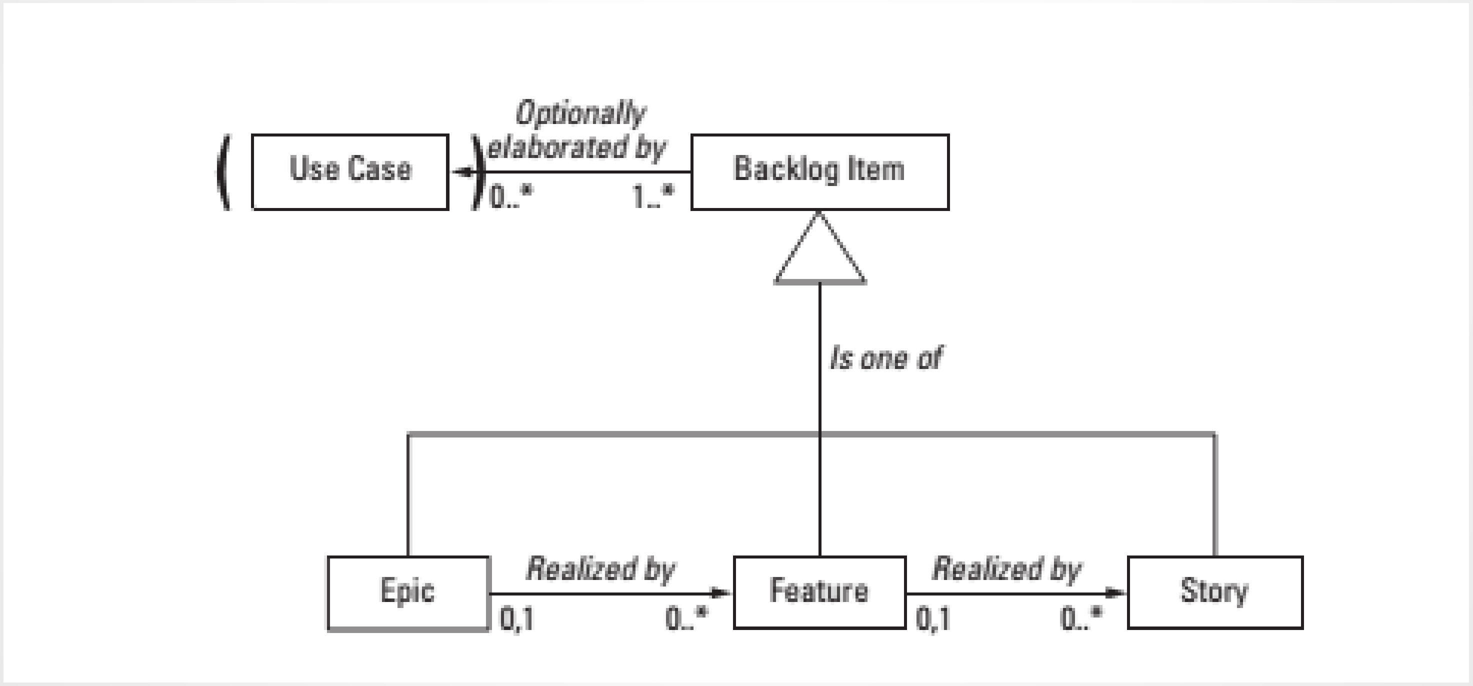
Refine the Use Cases

- Consider all alternate flows, including unusual exception conditions
- **Preconditions**: The refinement process will identify state information that controls the behavior of the system.
- **Exit conditions**: These describe the persistent states the use case leaves behind.

Tips for Applying Use Cases in Agile

- Keep them lightweight—no design details, GUI specs, and so on.
- Don't treat them like fixed requirements. Like user stories, they are merely statements of intended system behavior.
- Don't worry about maintaining them; they are primarily thinking tools.
- Model them informally—use whiteboards, lightweight tools, and so on.

Use Cases in the agile requirements information model


Use Case
Optionally elaborated by
0..*
1..*
Backlog Item
Is one of
Epic
Realized by
0,1
0..*
Feature
Realized by
0,1
0..*
Story