Killing Your Project with Use Cases !

The Top 5 Use Case Killers

Presented by:

William F. Nazzaro, Nazzaro & Associates / Gary K. Evans, Evanetics

The Top 5 Use Case Killers

1. CRUD-Based Use Case Partitioning
2. Design by Use Case
3. Actor Mis-Classification
4. Use Cases for All Requirements
5. Use Case Normalization

Our full presentation has all Five Use Case Killers – Please contact us to learn more about our Use Case Courses and Workshops and how we can transform your team into use case modeling and requirements experts.

bill@williamnazzaro.com / (610) 831-1151 or
gkevans@evanetics.com / (803) 781-1308

Presentation Outline

- Who Are We?
- Promise Of Use Cases
- The Pain of Use Cases
- Use Case Refresher
- Definitions
- Two of the Top 5 Use Case Killers
  - *CRUD-Based Use Case Partitioning*
  - *Design by Use Case*
  - *Actor Mis-Classification*
  - *Use Cases for All Requirements*
  - *Use Case Normalization*

Who Are We?

Gary K. Evans is an independent agile process consultant, and object technology mentor for Evanetics in Columbia, SC. He provides OO and process consulting, training services, and coaches teams in agile development techniques. He is a Contributing Editor with *Software Development* magazine, and is a judge for the acclaimed Jolt Awards.

Website:

www.evanetics.com

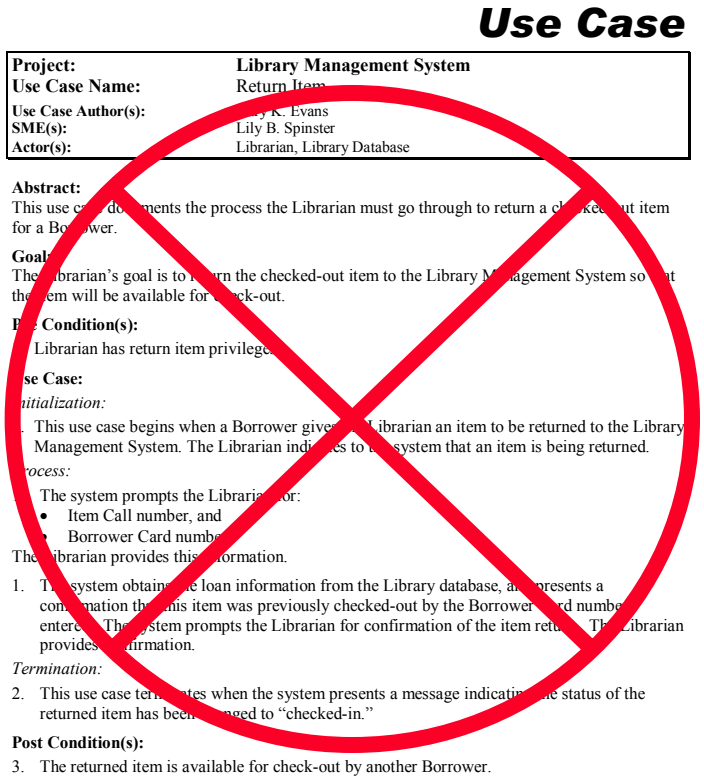
William F. Nazzaro is an independent RUP process and object technology consultant for Nazzaro & Associates located in Philadelphia, PA. He provides his clients with consulting, project management & oversight, off-shore development guidance & selection, process adoption & tailoring, mentoring, education delivery & curriculum development, and project execution for J2EE / .NET software development projects.

Website:

www.williamnazzaro.com

The Promise of Use Cases

- Use cases are supposed to be:
  - *A structured, but informal way of expressing functional requirements.*
  - *Easily accessible to all stakeholder groups.*
    - » *Natural language expression.*
    - » *No special training required to understand.*
  - *A vehicle for obtaining consensus on system vision and goals.*
- But use cases in practice have become:
  - *Difficult to develop.*
  - *Difficult to control.*
  - *Painful, and probably not worth the effort.*



It doesn't have to be this way.

The Pain of Use Cases

- First-time—and experienced—use case writers find:
  - *They don't know:*
    - *What to put in, or to leave out.*
    - *How much of the system's internal activity to include.*
    - *What to do with common, repeated chunks of functionality.*
  - *They put implementation details into the use case descriptions.*
  - *They try to put all requirements into the use case descriptions.*
  - *They use <<extend >> and <<include >> too much.*
  - *They either have:*
    - *Many use cases doing too little.*
    - *Few use cases doing too much.*
- Use case writers must clearly understand the use case's purpose.

How Do I Learn to Write Good Use Cases?

Go to the Theory?

- *But there is no single use case theory.*
- *Ivar Jacobson outlined a theory, but he didn't provide answers on how to implement this theory in a project setting.*

Go to the Literature?

- *Alistair Cockburn has identified at least 18 different definitions of what a use case is and how it should be written.*
- *Examples are usually short and terse (limited page space).*
- *The examples in the literature are usually hardware systems.*

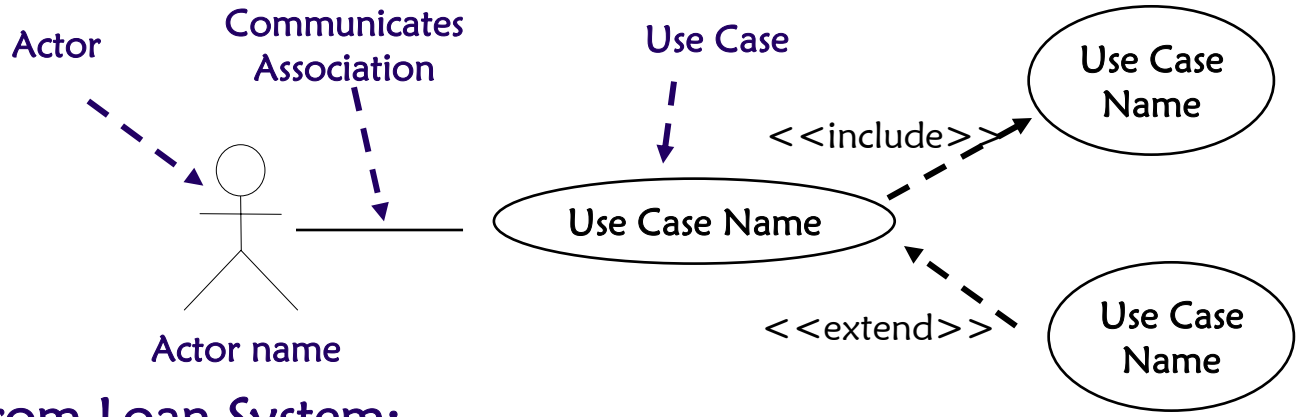
You need to learn from the experts and leverage our experiential education, workshops and project mentoring / coaching to fully embrace use case modeling – To learn more about how we can transform your team please contact us at:

bill@williamnazzaro.com / (610) 831-1151 or

gkevans@evanetics.com / (803) 781-1308

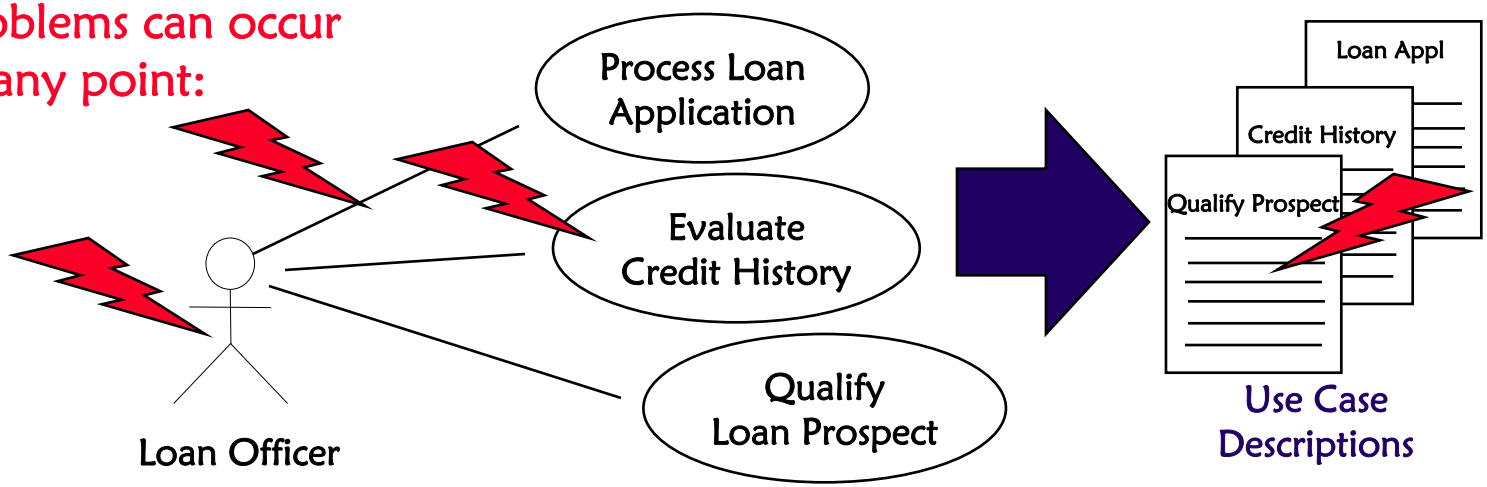
Use Case Refresher

Use Case Diagram Components:



Example from Loan System:

Problems can occur at any point:



Our Definitions

- Use Case
  - *Describes what happens in the system and how the system interacts with the actor when executed*
  - *Any behaviorally related sequence of transactions performed by a single actor in a dialogue with a system, the purpose of which is to provide some measurable value to the user**
- Actor
  - *An actor represents a source or destination of information that is outside the scope of the system*
  - *Any external class that models one of a set of roles played by a perspective user of a system**
- Use Case Diagram
  - *Documents a system's behavior as a set of actor, communications, and use cases*
  - *A use case diagram presents a collection of use cases and is typically used to specify or characterize the behavior of a whole system together with one or more actors that interact with that system**
- Communicates
  - *Communication lines are drawn to represent the interaction between actor and use case (only be concerned with who initiates the communication)*

1. CRUD-Based Use Case Partitioning

- Complaint:
  - *"When people refer to a discrete business transaction that is of meaningful value to the actor does that mean I should focus on CRUD (i.e. Create/Retrieve/Update/Delete)?"*
  - *"Our use cases are very clear about how we get and process data in the system, but they don't seem to really capture our business processes."*
  - *"Why do we have so many use cases?"*
- Symptoms:
  - *Use cases are partitioned and created based on CRUD considerations.*
  - *Use cases specify interaction with database rather than outlining the business process and interactions between actor and system.*

1. CRUD-Based Use Case Partitioning

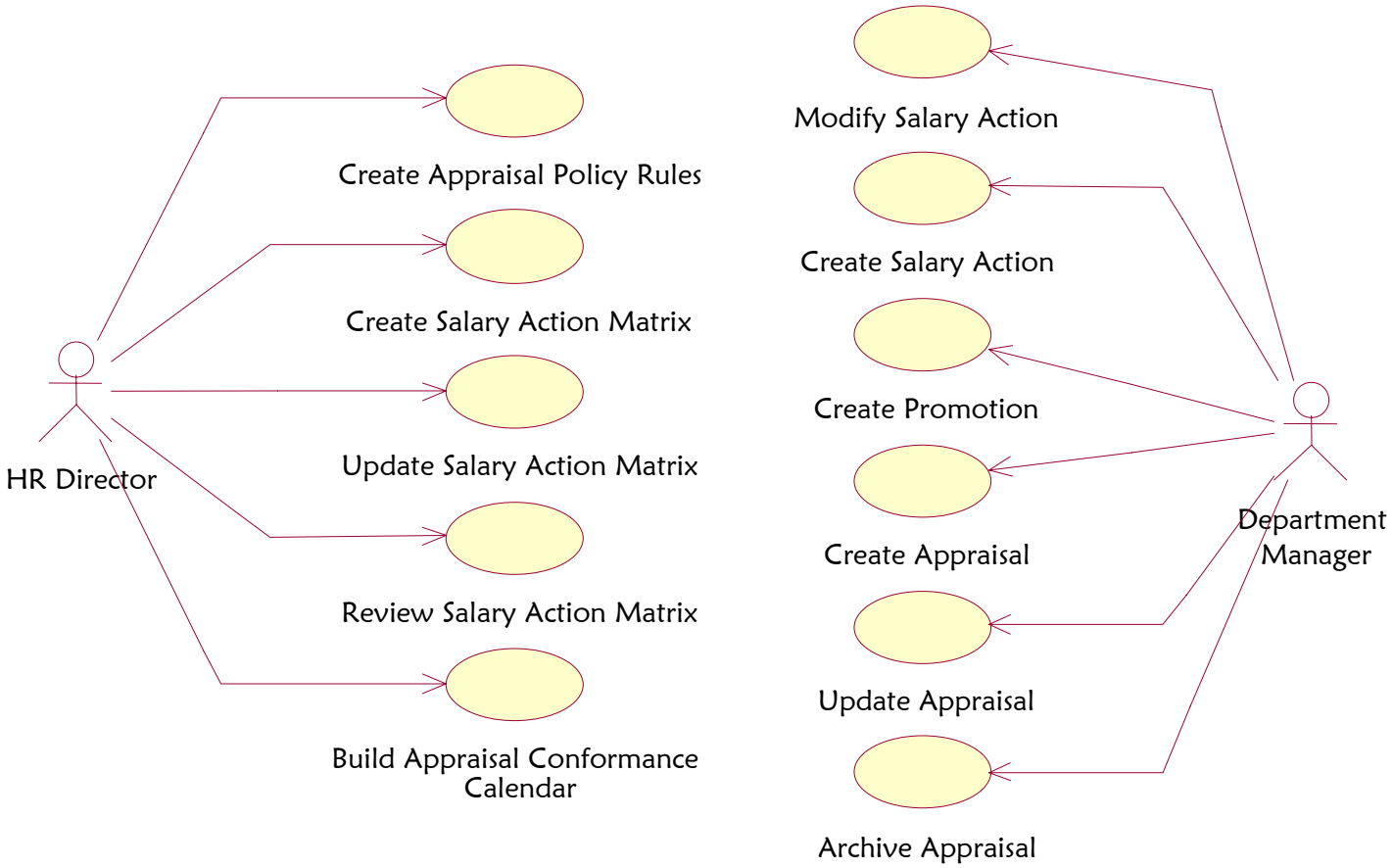
- Cause:
  - *Use case writer mistakenly substitutes a technology's interaction (e.g., database) as the business process.*

  NOTE: ***A "good" use case must describe a business transaction (defined beginning and end-point) that provides some meaningful value to the actor.***

  A use case could last days or months, but IT people focus on the word "transaction" and then slip into the database definition of the word transaction.

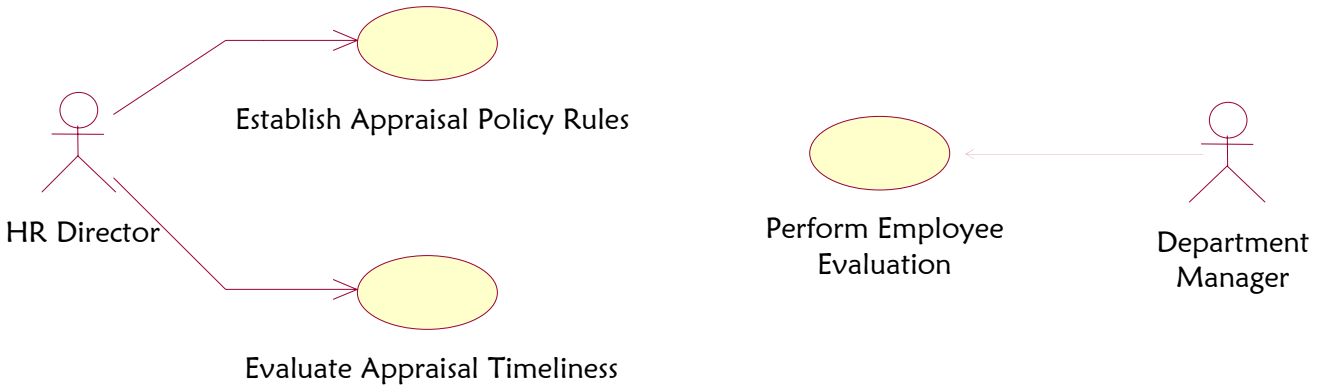
1. CRUD-Based Use Case Partitioning: Example

- An Employee Performance Appraisal System:



1. CRUD-Based Use Case Partitioning: Example

- A better Employee Performance Appraisal System:



1. CRUD-Based Use Case Partitioning: Why This is a Killer

- A) Will lead to too many use cases, use case proliferation.
- B) Use cases are too fine-grained.
- C) Will lose control of managing use case development.
- D) Will be difficult for the subject matter expert (SME) to understand the use case.
- E) Will be difficult for the SME to validate the use case because SME will not be able to recognize the business processes.

1. CRUD-Based Use Case Partitioning: Antidote

- Model the business process without regard to technology issues or constraints.
  - *Use cases should capture the process that is meaningful to the business, not the technical process implemented in the software application.*
  - *The technology will implement the business process.*
- CRUD-based partitioning is

- Remember:
  - *The system does not exist to read/write to a database.*
  - *It exists to produce something of measurable value for the actors..*

2. Design by Use Case

3. Actor Mis-Classification

4. Use Cases for All Requirements

- Complaint:
  - *"Where should I capture non-functional requirements (e.g., portability, reliability and scalability, etc.)?"*
  - *"Should I place business rules in my use cases?"*
- Symptoms:
  - *Use cases riddled with non-functional information such as:*
    - *Scalability*
    - *System performance specifications*
    - *Configurability*
    - *…the "-ilities"*

- Cause:
  - *Failure to partition requirements as functional and non-functional.*
  - *Failure to understand that the definition and enforcement of various requirements must be expressed in different contexts.*
  - *Placing requirements traceability within the use case description.*

4. Use Cases for All Requirements: Example

Use Case with non-functional requirements.

Project:	**Library Management System**
Use Case Name:	Return Item

Abstract:
This use case documents the process the Librarian must go through to return a checked-out item for a Borrower.

Goal:
The Librarian's goal is to return the checked-out item to the Library Management System so that the item will be available for check-out.

Pre Condition(s):

1. Librarian has return item privileges.

Use Case:

Initialization:

1. This use case begins when a Borrower gives the Librarian an item to be returned to the Library Management System. The Librarian indicates to the system that an item is being returned. [**Reqt: System must provide scalability to support 1000's of concurrent users.**]

Process:

2. The system prompts the Librarian for:
   - Item Call number, and
   - Borrower Card number.

   The Librarian provides this information.
3. The system obtains the loan information from the Library database, [**Reqt: system must segregate data services and business services using a 3-tier model.**] and presents a confirmation that this item was previously checked-out by the Borrower Card number entered. [**Reqt: 99.9% availability of repository must be assured.**] The system prompts the Librarian for confirmation of the item return. The Librarian provides confirmation.

Termination:

4. This use case terminates when the system presents a message indicating the status of the returned item has been changed to "checked-in." [**Reqt: System must serialize concurrent requests for check-in/check-out**]

Post Condition(s):
The returned item is available for check-out by another Borrower.

4. Use Cases for All Requirements: Example

Use Case with only functional requirements.

Project:	**Library Management System**
Use Case Name:	Return Item

Abstract:
This use case documents the process the Librarian must go through to return a checked-out item for a Borrower.

Goal:
The Librarian's goal is to return the checked-out item to the Library Management System so that the item will be available for check-out.

Pre Condition(s):

1. Librarian has return item privileges.

Use Case:

Initialization:

1. This use case begins when a Borrower gives the Librarian an item to be returned to the Library Management System. The Librarian indicates to the system that an item is being returned.

Process:

2. The system prompts the Librarian for:
   - Item Call number, and
   - Borrower Card number.

   The Librarian provides this information.
3. The system obtains the loan information from the Library database, and presents a confirmation that this item was previously checked-out by the Borrower Card number entered. The system prompts the Librarian for confirmation of the item return. The Librarian provides confirmation.

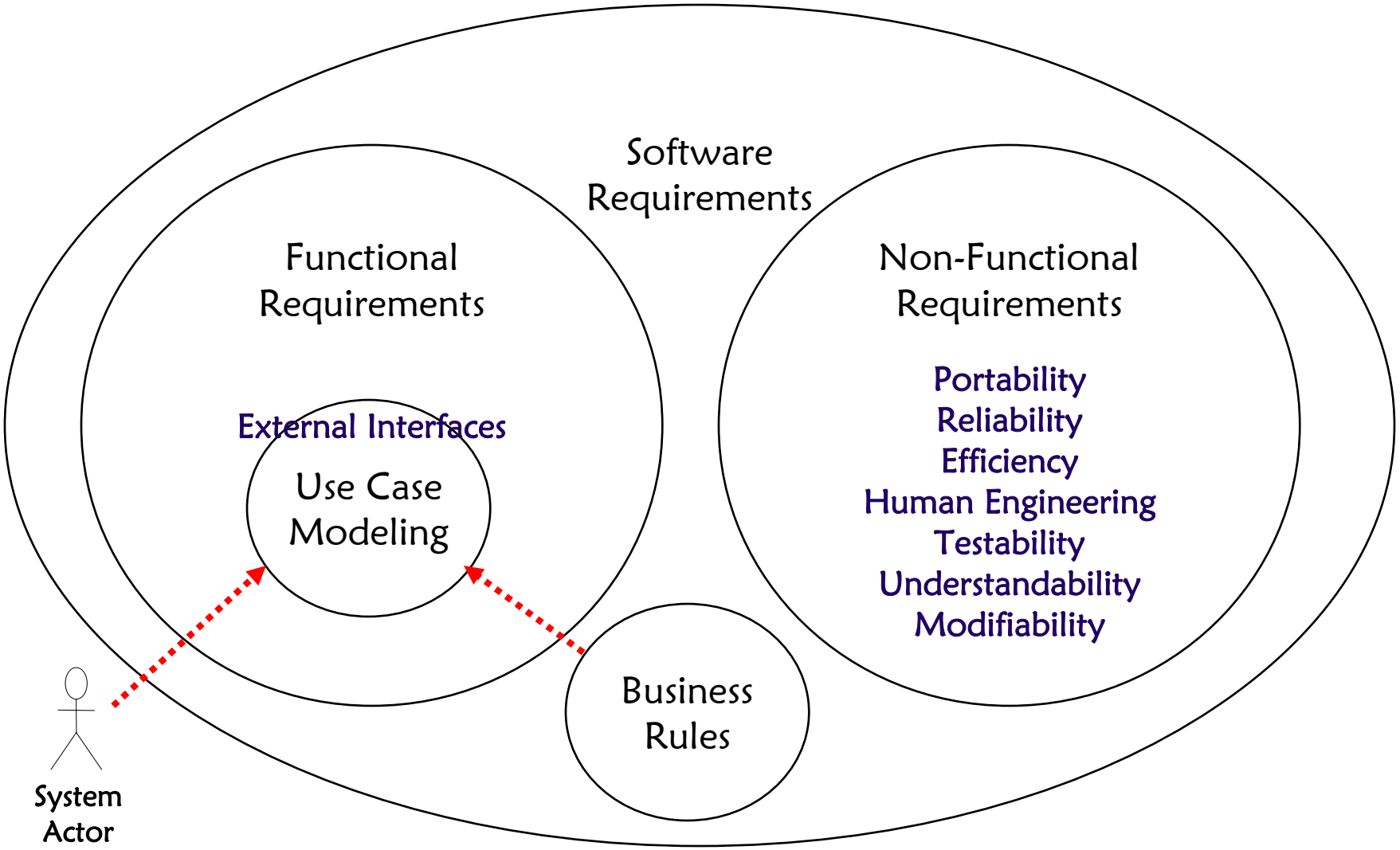
Termination:

4. This use case terminates when the system presents a message indicating the status of the returned item has been changed to "checked-in."

Post Condition(s):

The returned item is available for check-out by another Borrower.

4. Use Cases for All Requirements: Taxonomy



4. Use Cases for All Requirements: Why This is a Killer

A) The SME cannot validate the use case description because the business process is interrupted by the definition of business rules and the inclusion of non-functional requirements.

B) Including non-functional requirements within use case descriptions makes it difficult for the architect to extract and understand them.

Non-functional requirements are scattered.

Audience of the use case description are the SME and analyst, who use the use case descriptions to document the business processes.

Audience of the non-functional requirements is the architect, who uses them to define the system architecture. These are different audiences with different perspectives.

C) Use case descriptions should not define business rules.

Use case descriptions could demonstrate how certain business rules are enforced within the business process.

4. Use Cases for All Requirements: Antidote

- Define business rules and non-functional requirements in supplementary specifications.
- Use case descriptions do not contain non-functional requirements.
- Use case descriptions could demonstrate enforcement of business rules, but they do not define the business rules.
- Use a requirements traceability tool to couple defined business rules with their enforcement in the use case descriptions.
- Remember:
  - *The audience of the use case description is first the SME, second the analyst, and not the architect.*

5. Use Case Normalization

For More Information

Please contact us to learn more about our
Use Case Courses and Workshops and how
we can transform your team into
use case modeling and requirements experts.

bill@williamnazzaro.com / (610) 831-1151 or
gkevans@evanetics.com / (803) 781-1308