REQUIREMENTS ENGINEERING - USECASE MODELLING

DR. VIVEK NALLUR

VIVEK.NALLUR@SCSS.TCD.IE

https://www.scss.tcd.ie/vivek.nallur/teaching/slides/

OUTLINE OF THIS TALK

- Representing Requirements
- Usecase Modelling

REPRESENTING REQUIREMENTS

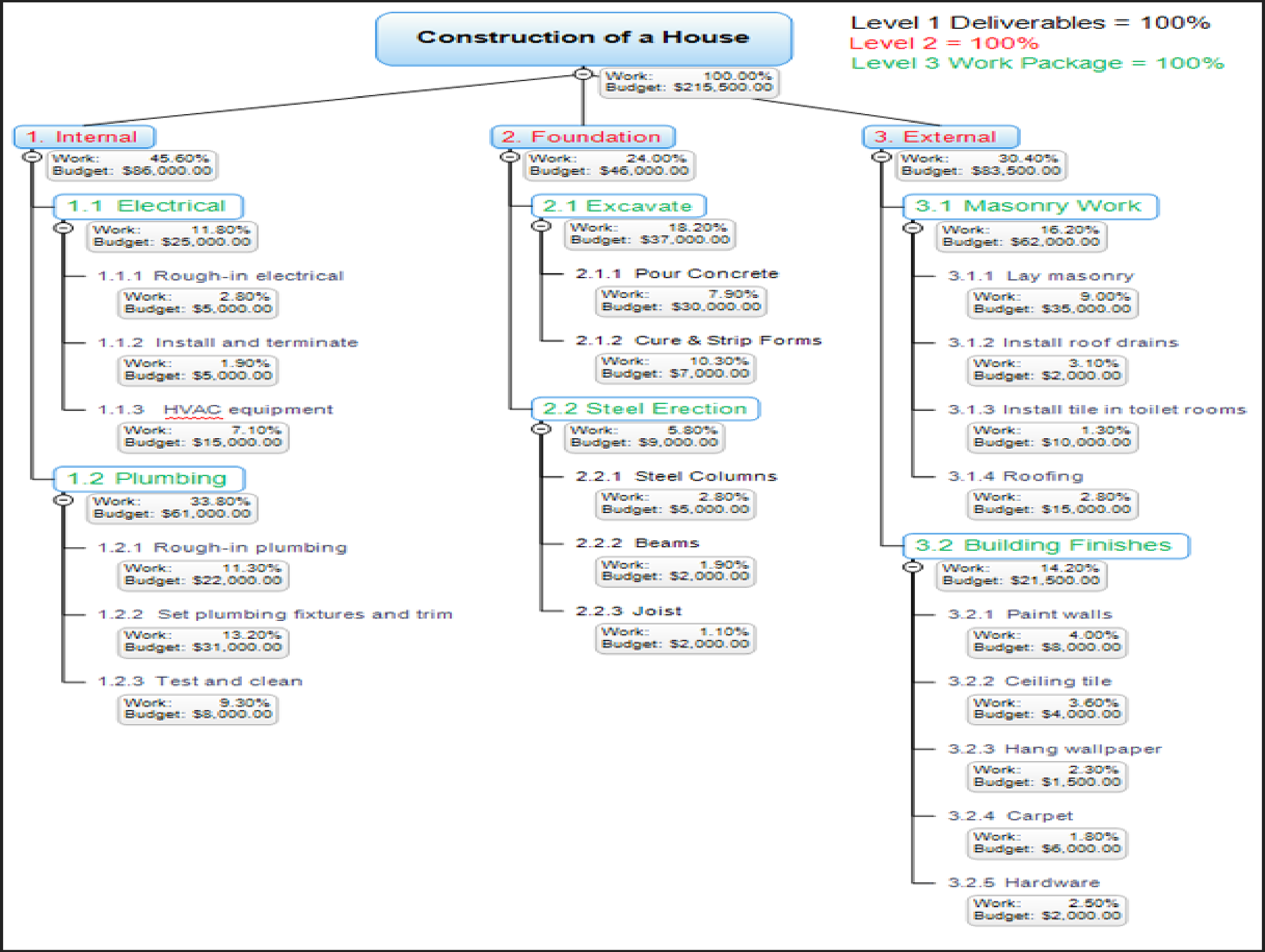
A *Software Requirements Specification* (SRS) is a description of a system to be developed. It contains:

- Functional requirements
- Non-functional requirements
- Constraints
- External Interface requirements (human, machine,...)

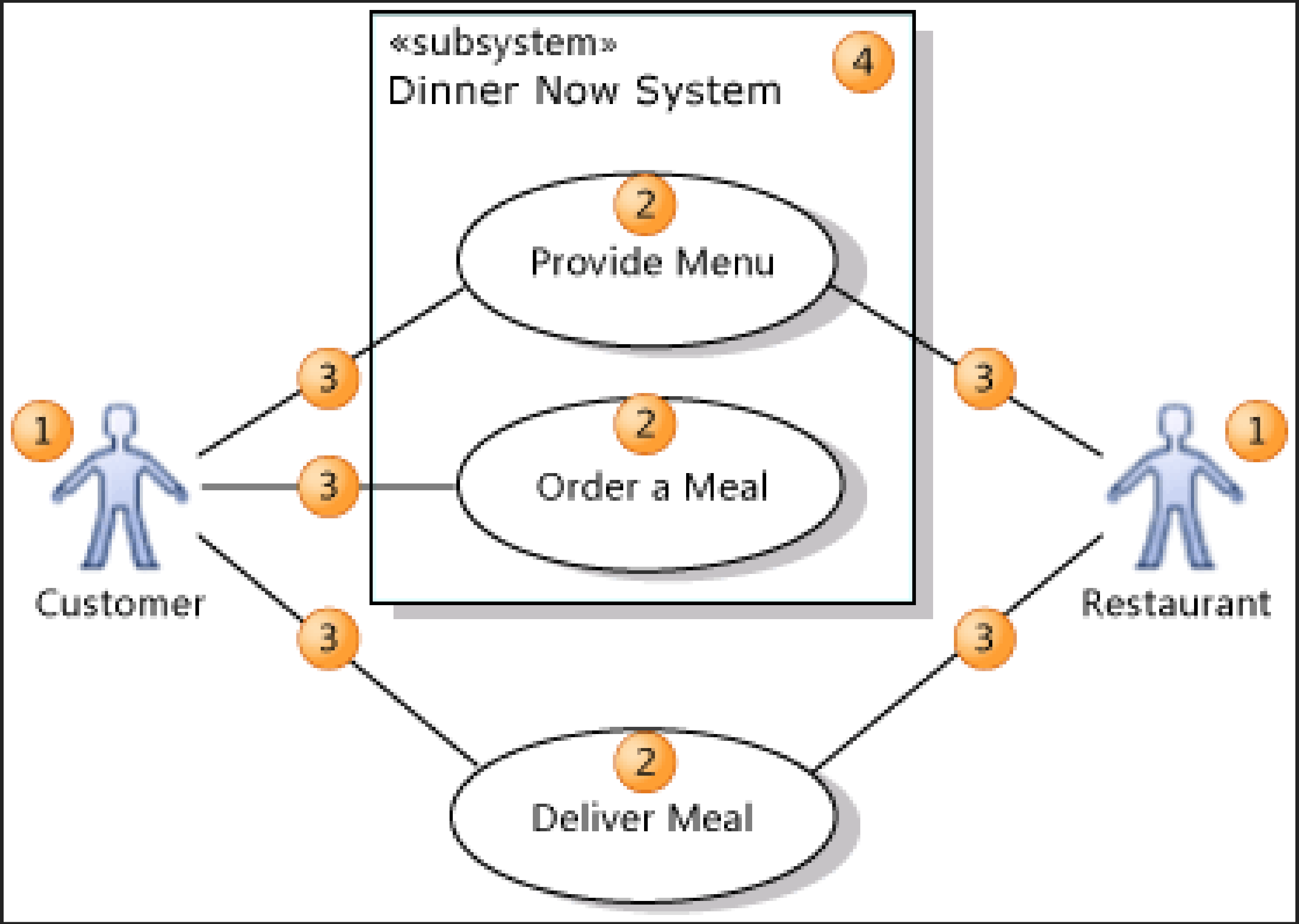
TECHNIQUES OF WRITING DOWN REQUIREMENTS

- Work-breakdown Structure
- Use cases
- Data Flow Diagrams, Activity Diagrams, Petri Nets, ...

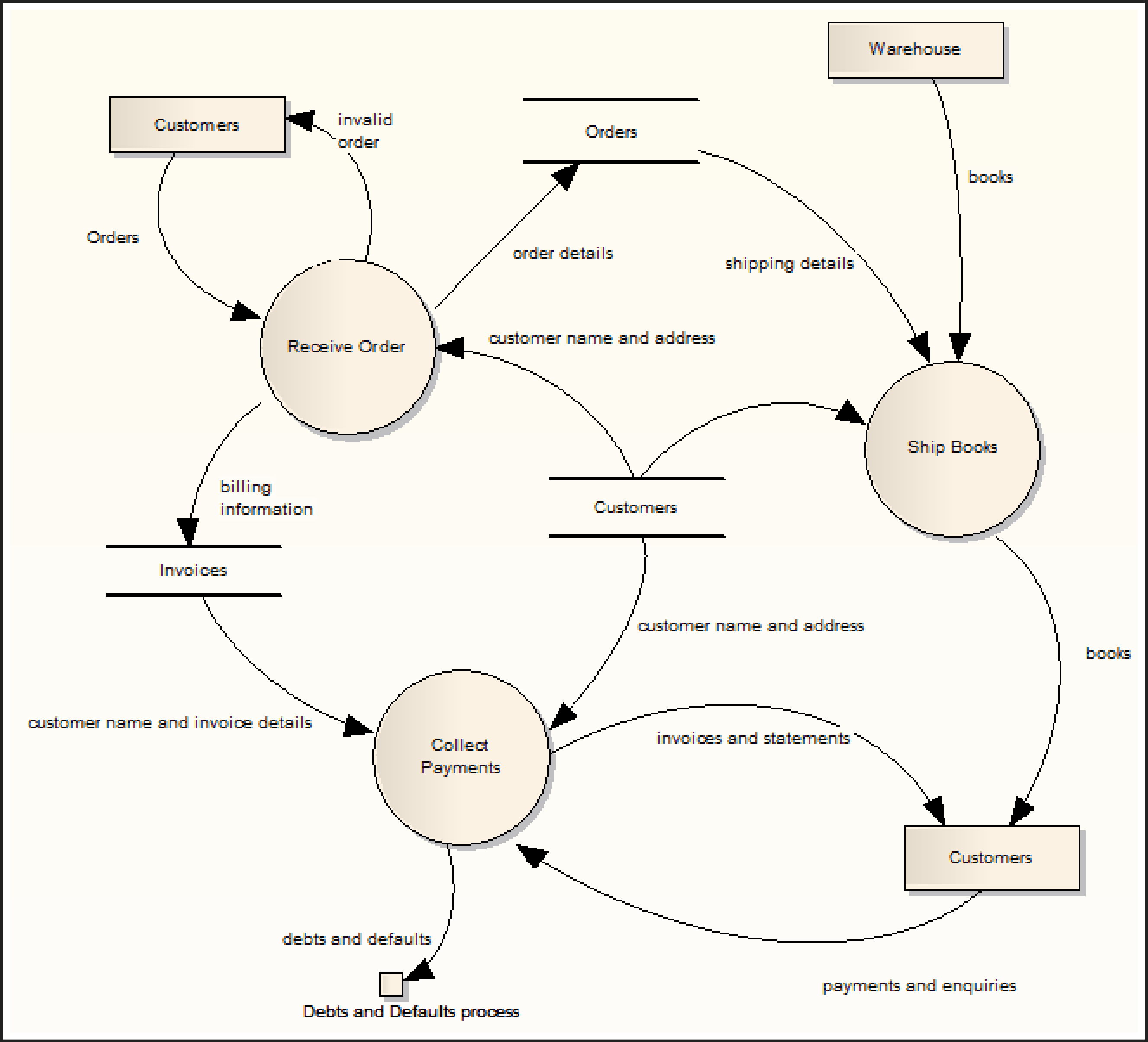
WORK-BREAKDOWN STRUCTURE



USE CASES


«subsystem»
Dinner Now System
4
2
Provide Menu
2
Order a Meal
2
Deliver Meal
1
Customer
1
Restaurant
3
3
3
3
3

DATA FLOW DIAGRAMS



USE CASES

- Invented by Ivar Jacobson in the 1960s
- Adopted by the object-oriented programming community in the 1990s
- Part of UML suite of diagrams
    - Still principally a text-based method

WHAT IS A USECASE?

> *A usecase is a description of the possible sequences of interactions between the system under discussion and its external actors, related to a particular goal*

- A thought-capturing format of writing
- Documents the *behaviour* of a system

WHAT IS A USECASE - 2

- Associated with the goal of **one particular actor**, called primary actor
- Describes various sets of interactions between various actors, while the primary actor is in pursuit of that goal
- Each possible sequence of interactions is called a *scenario*

WHAT IS A USECASE - 3

- A usecase collects together:
  - All scenarios related to *that* goal of *that* primary actor
  - Including those where goal is achieved
  - Including those where goal is not achieved
- An actor may be: a person, a group of people, a computer system, a component ...

STAGES OF USECASE WRITING

- Actors & Goals
- Definition of success
- Failure conditions
- Failure handling

TOLERANCE AND HARDNESS

- No two people have *exactly* the same style
  - Bird's-eye view or P.o.V
- High-ceremony vs. Low-ceremony
  - Medical devices, power plants, vehicles, ...
  - Small websites, point-of-sale software, ...
- Use the appropriate template

BASIC CONCEPTS: ACTORS

- Anything that has behaviour (also called *role*)
- System Under Discussion (SuD)
- Subsystems of SuD (internal actors)
- Primary Actor
- Secondary Actor(s)

GOALS

- Functional scope
  - Create a goal-list
  - Decide whether it is "In Scope" or "Out of scope"
- Design scope
  - Boundary of the system (hardware, software, ...)
  - Interfaces to secondary actors

ACTORS HAVE GOALS

A goal not related to any primary actor will not get done!

- An actor achieves a goal, through interactions:
  - A simple interaction (*e.g.*, message, method-call, etc.)
  - A sequence of interactions
  - A set of sequences

BASIC CONCEPTS: SCENARIOS

- Conditions
  - Pre-conditions for the goal (*e.g.*, trigger, other goals)
  - Post-conditions for the goal
- Goal success (Basic Flow)
- Goal failure (Alternate Flow)

DEFINITION OF A SCENARIO

A scenario is a sequence of interactions that happens under certain conditions, with the intent of achieving the primary actor's goal. The interactions start from the triggering action and continue till the goal is achieved or abandoned.

USECASE IN DETAIL

- A scenario consists of steps (each step is an action)
- An action is of three sorts:
  - An interaction between two actors
  - A validation
  - An internal state change
- The SuD must satisfy and protect the interests of all agreed-upon actors
- Scenario ends when all interests of actors are satisfied or protected

LEVELS OF GOALS

- Very High-level (Organization-wide)
- High-level (business / strategic / system)
  - *E.g.*, Finish assignments for Software Engineering
  - *E.g.*, Buy concert tickets
- Low-level (user-goal)
  - *E.g.*, Compile code
  - *E.g.*, Make payment on website
- Sub-function: Usually exceedingly simple (*e.g.*, log on)

SAMPLE USECASE

- Goal: Operate an insurance policy
- Primary Actor: Customer
- Scope: Functional
- Level: High
- Steps:

1. Customer gets a quote for a policy
2. Customer buys a policy
3. Customer makes a claim against the policy
4. Customer closes the policy

PHRASING A GOAL

verb direct-object

- Examples:
  - Get a quote
  - Buy tickets
  - Finish assignment
  - Transfer money

WRITING A LOW-LEVEL USECASE

- Each step from a high-level usecase can be a goal of a low-level usecase
- Steps for low-level usecases are usually actions
- One low-level usecase can call/refer to another low-level usecase, if necessary [using `<includes>`]

PRE-CONDITION

- Pre-condition: Assertions about the state of the world before usecase can be triggered
  - Not checked during the usecase
  - *e.g.*, User is logged on, disk has sufficient space
- Bad pre-condition: conditions normally true, but cannot be guaranteed
  - *e.g.*, Before transferring file, user has saved latest copy

SUCCESS END CONDITION

- Goals of all actors (primary and/or secondary) have been achieved
- Assertion about state of the world at moment of successful completion
  - *e.g.*, File saved onto Dropbox

FAILURE END CONDITION

- Many ways to fail
- Usually described with a conditional
  - *e.g.*, If ATM did not dispense cash, user's account is not debited
  - *e.g.*, If `paste` location is not valid, `cut` does not delete file

MAIN SUCCESS SCENARIO (BASIC FLOW)

- Usually goal-achieving action by primary/secondary actor or SuD
- Most typical, easiest to understand scenario
- Definitely has no failures
- Sentence moves the process distinctly forward
  - *E.g.*, system verifies that order is placed

EXTENSIONS (ALTERNATE FLOW)

- Brainstorm all conceivable failures
- List alternate success paths
- Evaluate, eliminate, merge ideas

TYPICAL REASONS FOR FAILURES

- Primary actor sends bad data or requests
- Validation checks not passed
- Secondary actors do not respond or fail
- Inaction by primary actor (timeout)
- Bad data discovered inside SuD

DOCUMENTING ALTERNATE FLOWS

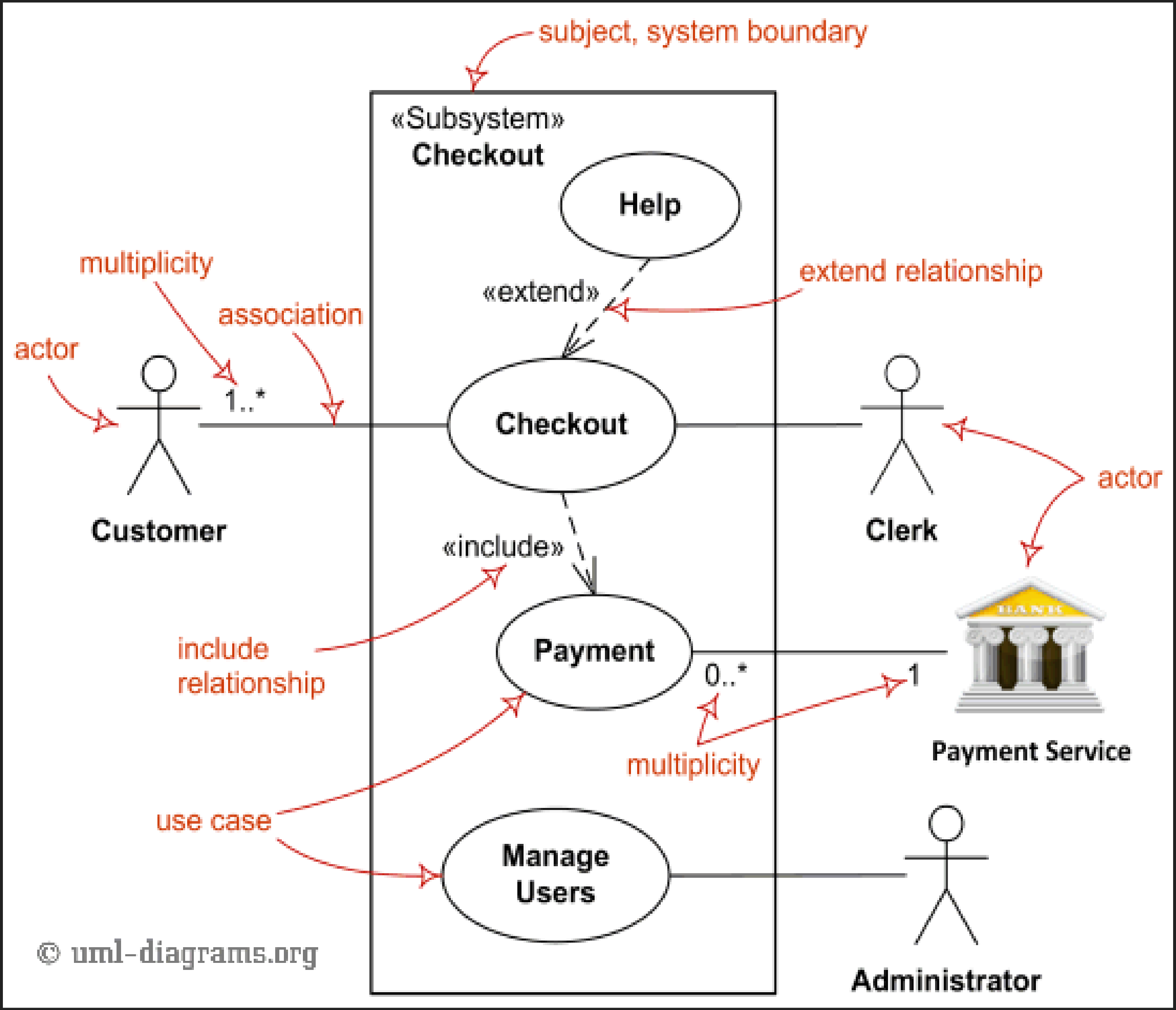
- Use step number of main success flow where alternate can occur
  E.g., 2: Withdraw cash
- Examples:
  - *2b*: Network down
  - *2c*: Insufficient funds

DOCUMENTING ALTERNATE FLOWS -2

Scenario ends in one of three ways:

- Failure condition is fixed
  e.g., User finally enters the correct password
- Failure condition causes scenario to re-start
  e.g., User uses different credit card
- Failure condition causes scenario to abort
  e.g., User quits

PICTORIAL REPRESENTATION



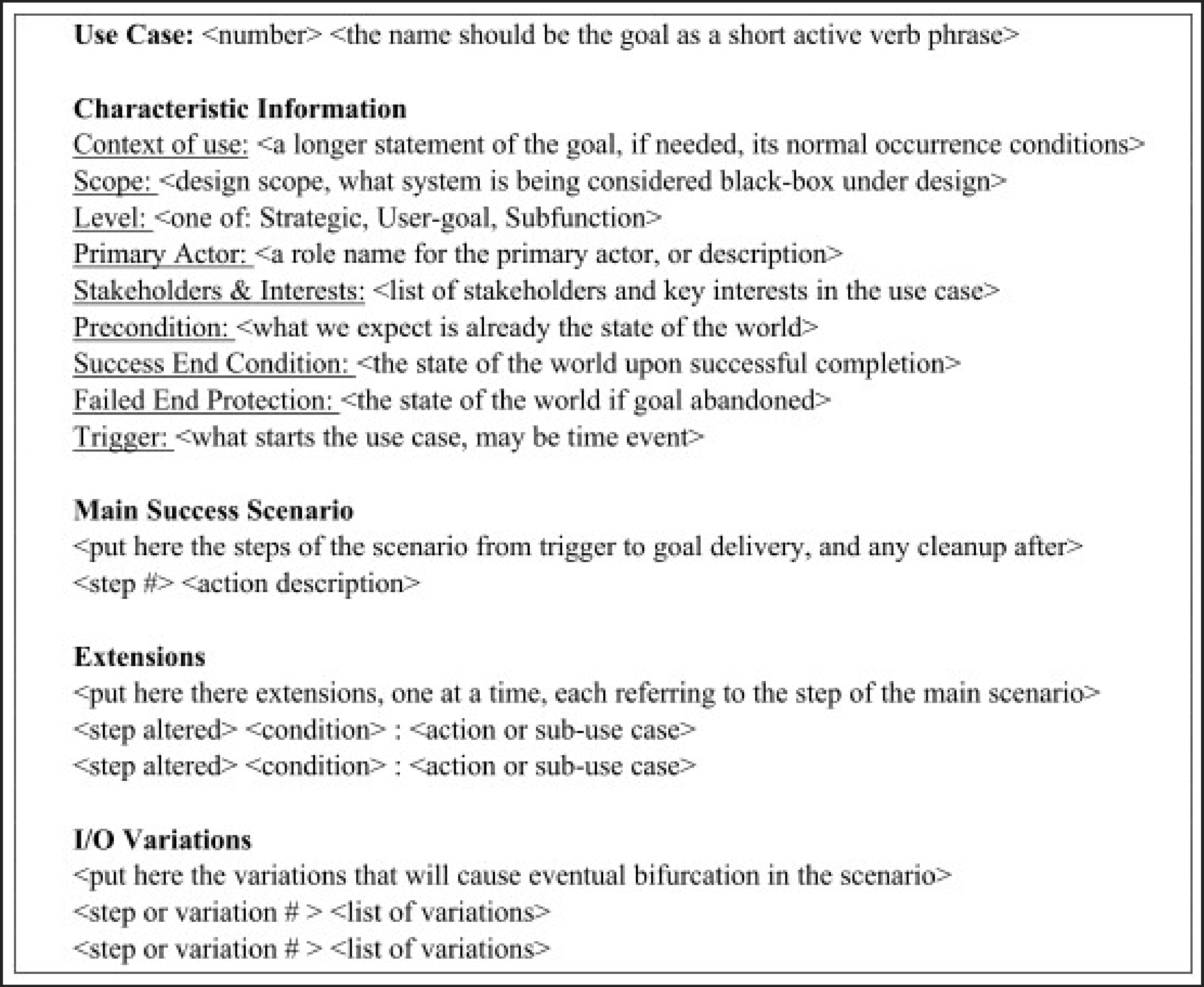
ADDITIONAL CONCEPTS: STAKEHOLDERS

- Someone with a vested interest in the behaviour of a usecase
- Not necessarily an actor (*e.g.*, customer, regulator, auditor)
- May not even appear in the description
- SuD must protect the interests of the stakeholders

FROM ACTORS & GOALS TO STAKEHOLDERS & INTERESTS

- S&I does not invalidate A&G (it's a superset)
- More precise about everybody involved
- Remember: primary actor is still a stakeholder

SAMPLE USECASE TEMPLATE



Use Case: <number> <the name should be the goal as a short active verb phrase>

Characteristic Information

Context of use: <a longer statement of the goal, if needed, its normal occurrence conditions>
Scope: <design scope, what system is being considered black-box under design>
Level: <one of: Strategic, User-goal, Subfunction>
Primary Actor: <a role name for the primary actor, or description>
Stakeholders & Interests: <list of stakeholders and key interests in the use case>
Precondition: <what we expect is already the state of the world>
Success End Condition: <the state of the world upon successful completion>
Failed End Protection: <the state of the world if goal abandoned>
Trigger: <what starts the use case, may be time event>

Main Success Scenario

<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>
<step #> <action description>

Extensions

<put here there extensions, one at a time, each referring to the step of the main scenario>
<step altered> <condition> : <action or sub-use case>
<step altered> <condition> : <action or sub-use case>

I/O Variations

<put here the variations that will cause eventual bifurcation in the scenario>
<step or variation # > <list of variations>
<step or variation # > <list of variations>

READING LIST

- Use Case Model - Sparx Systems
- Use case modelling - Open University
- Writing Effective Usecases by Alistair Cockburn
- UML Distilled by Martin Fowler

THAT'S ALL, FOLKS!

Questions? Comments?