The University of Iowa

CS:2820 (22C:22)

Object-Oriented Software Development

Spring 2015

Iteration 1: onto Elaboration

by
Mauricio Monsalve

Inception

- Suppose inception is done; then
- Use cases/requirements:
  - Short requirements workshop done
  - **Most** use cases, actors, goals defined/named
  - Most use cases written in brief format

    10-20% in detail, though
  - most influential and risky quality requirements identified

Inception

- Technical proof-of-concept (feasibility)
  - *Hello worlds to check if library X can be combined with Y*
  - *Can we do X on system Y? Show so*
- User interface-oriented prototypes
  - HTML, Slides, etc., to show how functionality would work

Inception

- Preparing the elaboration
  - Recommendations on what components to buy/build/reuse

    We don't need to do everything from scratch
  - High-level candidate architecture and components proposed

    This is just a brief speculation of how to organize the components of the system. It can change.
  - Plan for the first iteration

    This is what we talk about here

Iterative-incremental development

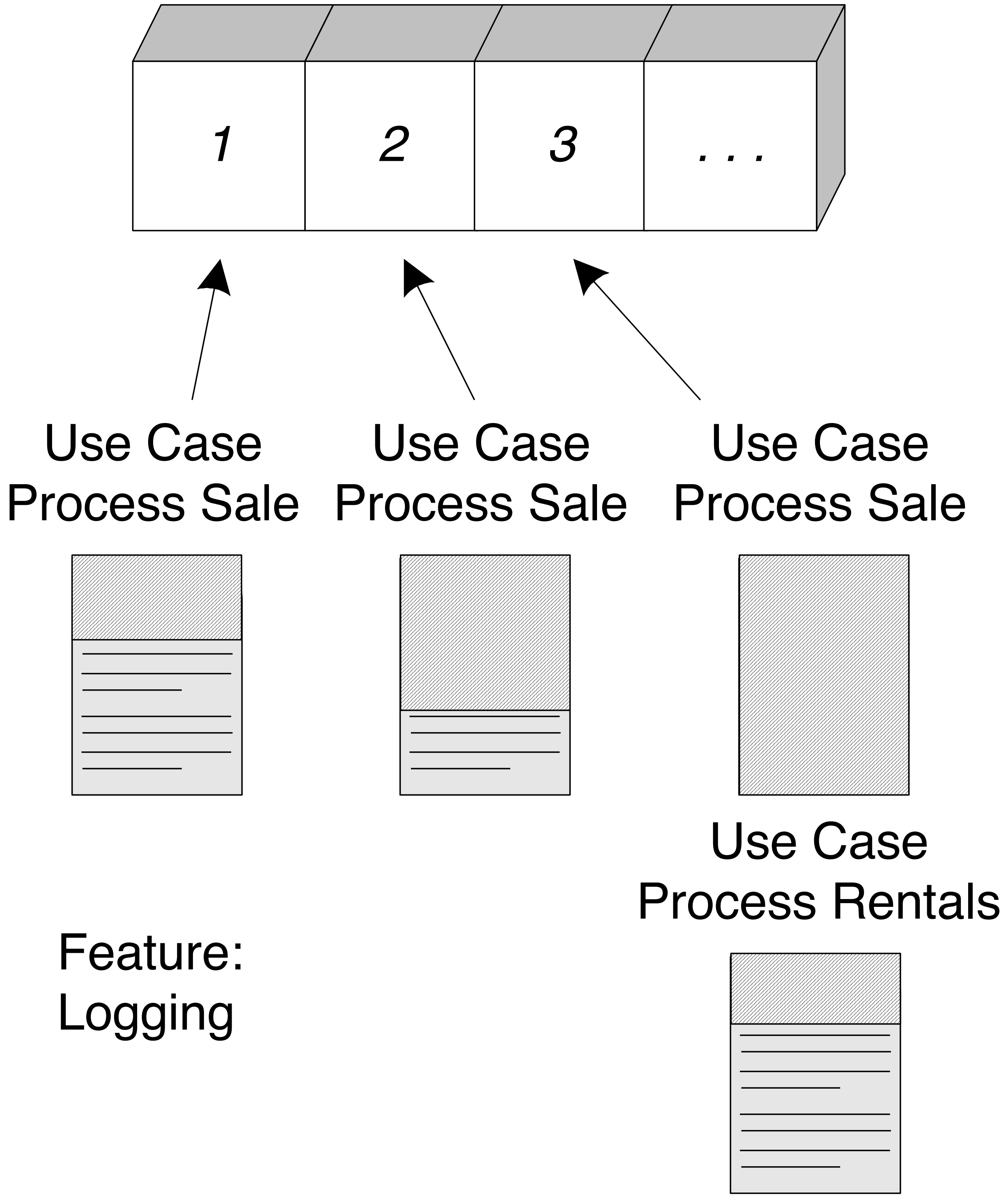
- The UP is both iterative and incremental
- Iterative development:

  Development is split in iterations
- Incremental development:

  Complexity (size) is increased in iterations

Not all requirements are tackled

in a single iteration



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

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

Elaboration

Elaboration is the initial series of iterations during which, on a normal project:

- the core, risky software architecture is programmed and tested
- the majority of requirements are discovered and stabilized
- the major risks are mitigated or retired

Elaboration

- Consists of 2 or more **iterations**
- Iterations: 2 to 6 weeks
  - prefer shorter iterations, unless large team
- During this phase, one is not creating throw-away prototypes
  - code and design are production-quality portions of the final system

Elaboration

In one sentence:

Build the core architecture,
resolve the high-risk elements,
define most requirements, and
estimate the overall schedule and resources

Elaboration

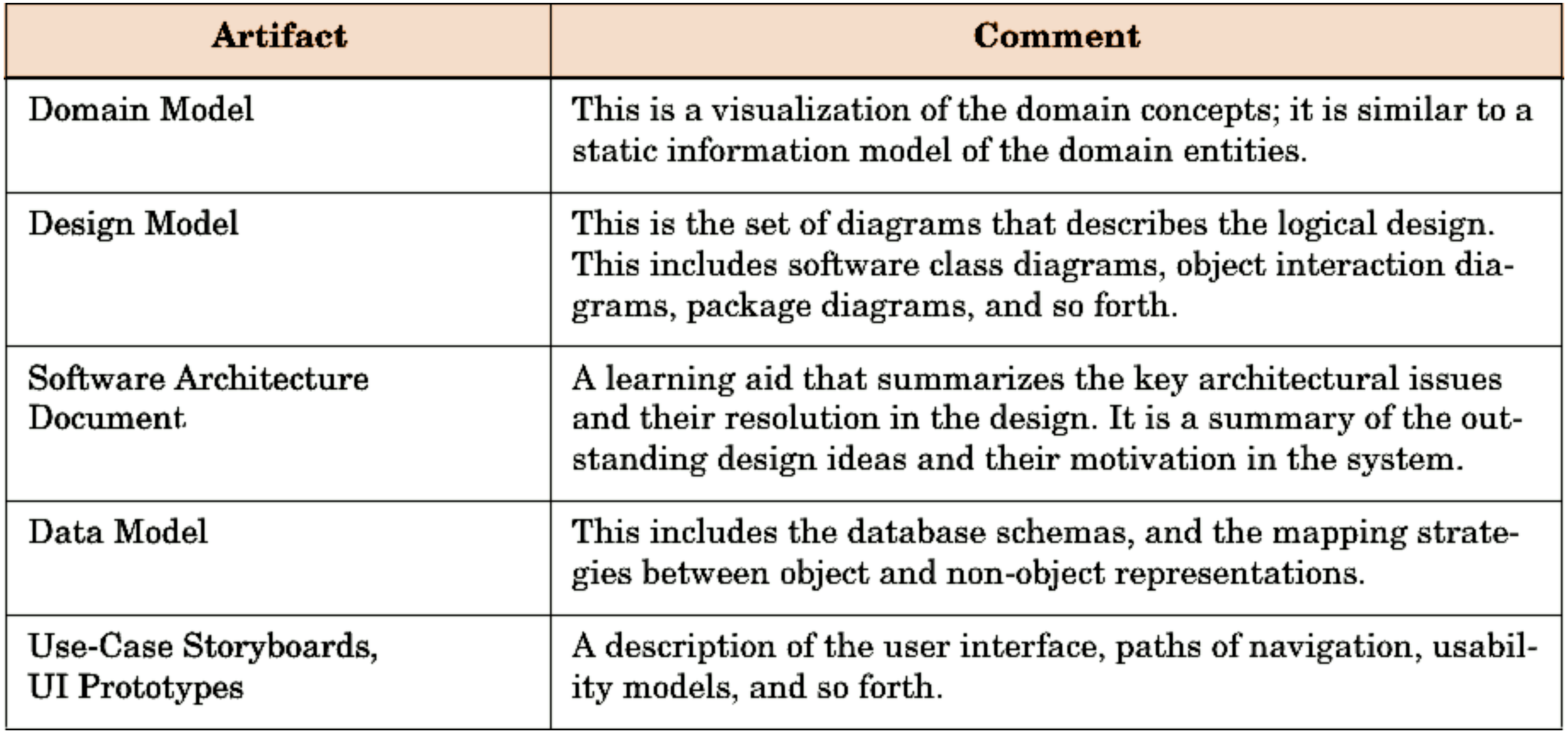
- Recommendations
  - Short timeboxed risk-driven iterations
  - Start programming early
  - Adaptively design, implement, and test the core and risky parts of the architecture
  - Test early, often, realistically
  - Adapt on feedback from tests, users, developers
  - Write most of the use cases and other requirements in detail, through a series of workshops, once per elaboration iteration

Elaboration

- How to prioritize work?
- Prioritize requirements
- Rank requirements based on
  - Criticality: degree of value to the customer
  - Coverage: how much of the system is related to the requirement
  - Risk: technical complexity, uncertainty of effort, usability
- Start with top ranked requirements

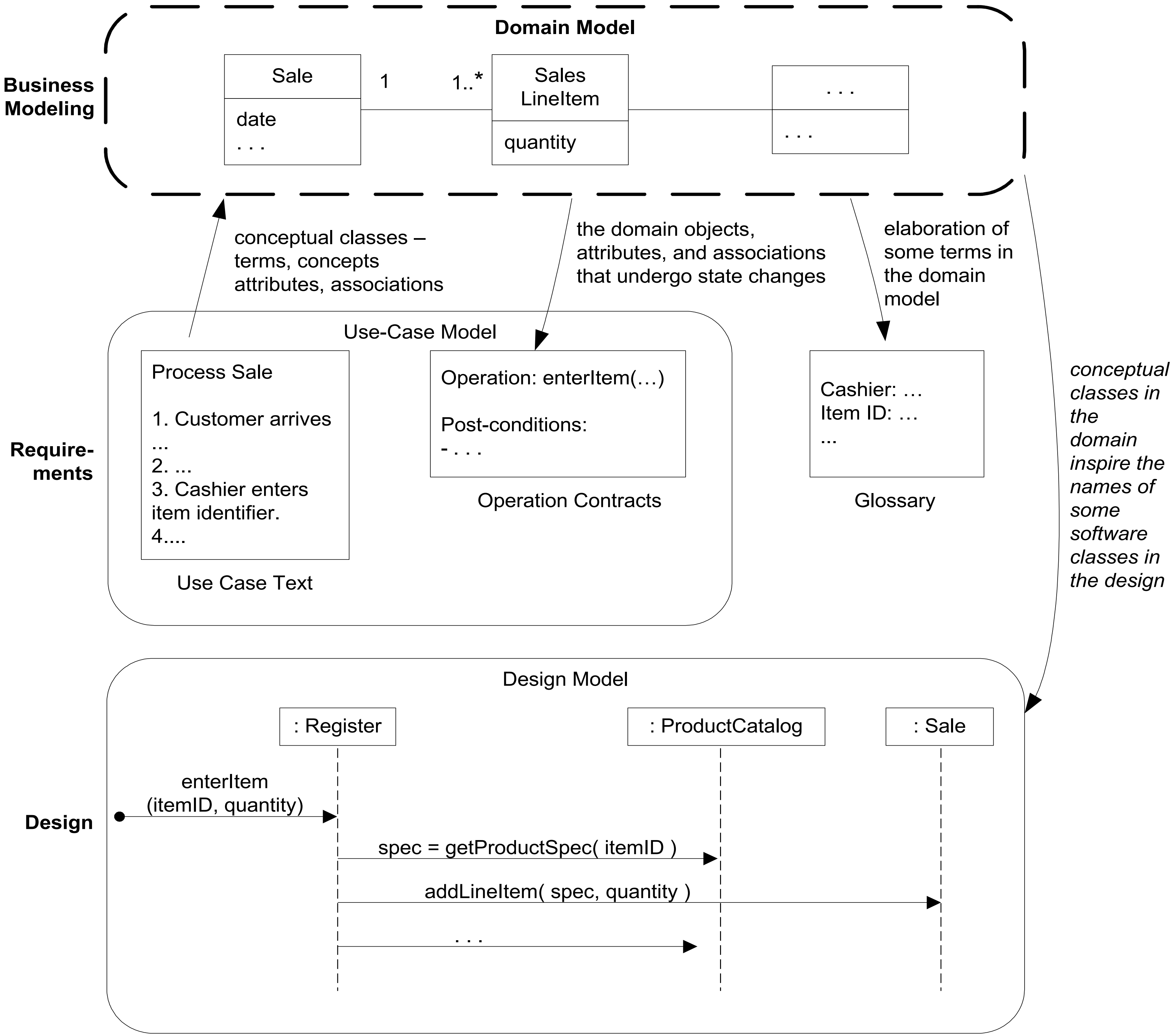
Elaboration

Some artifacts produced during the elaboration



Artifact	Comment
Domain Model	This is a visualization of the domain concepts; it is similar to a static information model of the domain entities.
Design Model	This is the set of diagrams that describes the logical design. This includes software class diagrams, object interaction diagrams, package diagrams, and so forth.
Software Architecture Document	A learning aid that summarizes the key architectural issues and their resolution in the design. It is a summary of the outstanding design ideas and their motivation in the system.
Data Model	This includes the database schemas, and the mapping strategies between object and non-object representations.
Use-Case Storyboards, UI Prototypes	A description of the user interface, paths of navigation, usability models, and so forth.

Sample UP Artifact Relationships


Domain Model
Business Modeling
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
conceptual classes – terms, concepts attributes, associations
the domain objects, attributes, and associations that undergo state changes
elaboration of some terms in the domain model
Use-Case Model
Requirements
Process Sale
1. Customer arrives
...
2. ...
3. Cashier enters item identifier.
4....
Use Case Text
Operation: enterItem(…)
Post-conditions:
- . . .
Operation Contracts
Cashier: …
Item ID: …
...
Glossary
conceptual classes in the domain inspire the names of some software classes in the design
Design Model
Design
: Register
: ProductCatalog
: Sale
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
addLineItem(spec, quantity)
. . .

Agile pedagogy!

- Re-read the slides—make associations
- Can you explain why we said what we said?

  Obviously, because we think it is useful and right, but why do we do so?
- Do you understand the implicitly defined concepts?
  - There are many concepts built from earlier work
  - We do not describe all possibilities:
    - *E.g., iterative and incremental development... what is [not]?*
- Can you apply some UP concepts to studying?