Introduction to Software Design

Greg Butler

Computer Science and Software Engineering
Concordia University, Montreal, Canada

Email: gregb@cs.concordia.ca

Design is a Blueprint for the Software

Blueprint describes “mechanism” that does the computing

Design is a Phase in the Software Process

Requirements = WHAT

Design = HOW

Implementation and Test

Deployment

Maintenance and Evolution

Design is an Activity to Create the Design

Design is a Series of Decisions to Resolve Design Issues

Design = Trade-Offs = Setting Priorities

Design Cycle

Draft design and describe clearly
Review/Assess/Evaluate design — discover issues
Resolve design issue = make decision

Issue-Driven Design

Design issue has alternative solutions
Solutions have pros and cons
Weigh pros and cons to make decision
Incorporate selected solution to resolve issue

Design Includes a Construction Plan for the Software

Construction Plan

Must have plan to develop/implement/construct the software
Based on dependencies between design components

Construction Plan determines ...

schedule of implementation and test
skills required by developers
number of developers needed
when developers are needed
work tasks for developers

Design Turns a Black-Box into a White-Box of Black-Boxes

Architectural Design

Decompose system into parts (layers, subsystems, ...)
that collaborate to fulfil contracts
Describe services and interface of each part
Facade pattern to encapsulate part (make it a black-box)
Specify contract of each interface

Top-down Design is Recursive

ie Nested boxes

Design by Contract

Design is a Document to Describe the Design

Design in Seamless OO Software Development

OO Analysis — System is Black-Box

Static Info

Objects are entities in the problem domain
Associations between entities
Main attributes of entities
Organize entities into classes

Dynamic Info

Behaviour as use cases & scenarios
System operations specificed by contracts

OO Design

OOA entities become software entities
Interfaces to encapsulate objects
Collaborations of objects to execute work
Object internal details — data structures and algorithms

OO Programming

Implementation
[Unit Testing]

OO Software Evolution

Extension Points; Variation Points
Sublcassing; Polymorphism
Refactoring
Frameworks; Generative SE; Software product Lines

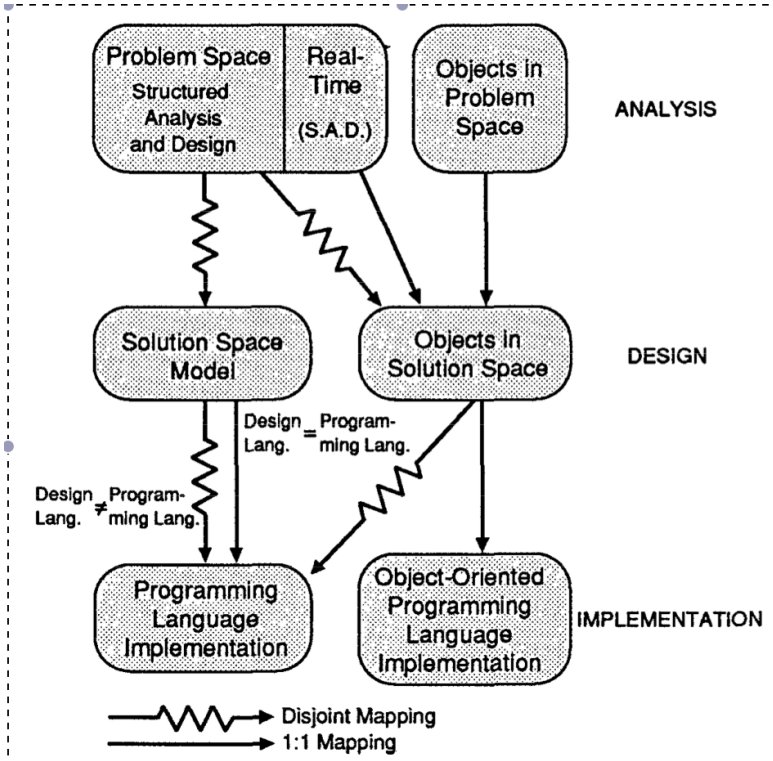
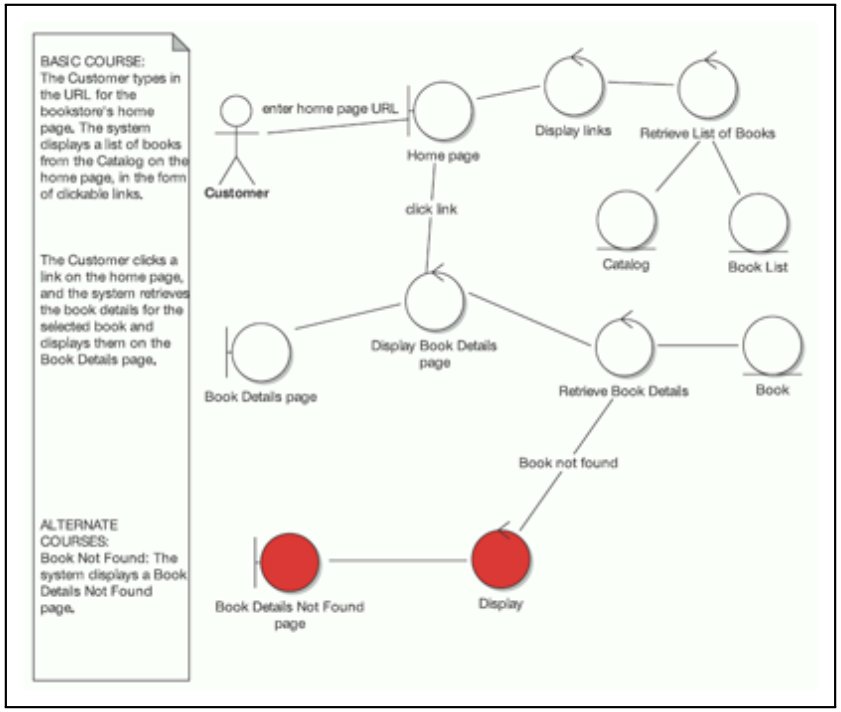


FIGURE 4. Functional and object-oriented decomposition and their possible interlinking. One-to-one mappings are indicated by arrows and disjoint mappings by wavy arrows.

OOSD — Jacobson Robustness Analysis

Robustness analysis checks consistency of OOA model



boundary objects — at edge of black-box system
control objects — coordinate one use case
entity objects — for OOA entities
Trace scenarios

OOSD — Responsibilities

CRC Card

CRC Card	
Class: StudentRecord	
Responsibilities:	Collaborators:
contains description of student	Description
has a collection of Assignments	Assignment
changes grades and Curve Status for assignments when asked by Gradebook	GradeBook, Assignment

C = Class, name of class being described
R = Responsibility, purpose of the class
C = collaboration, other classes that work with the class

"No object is an island"

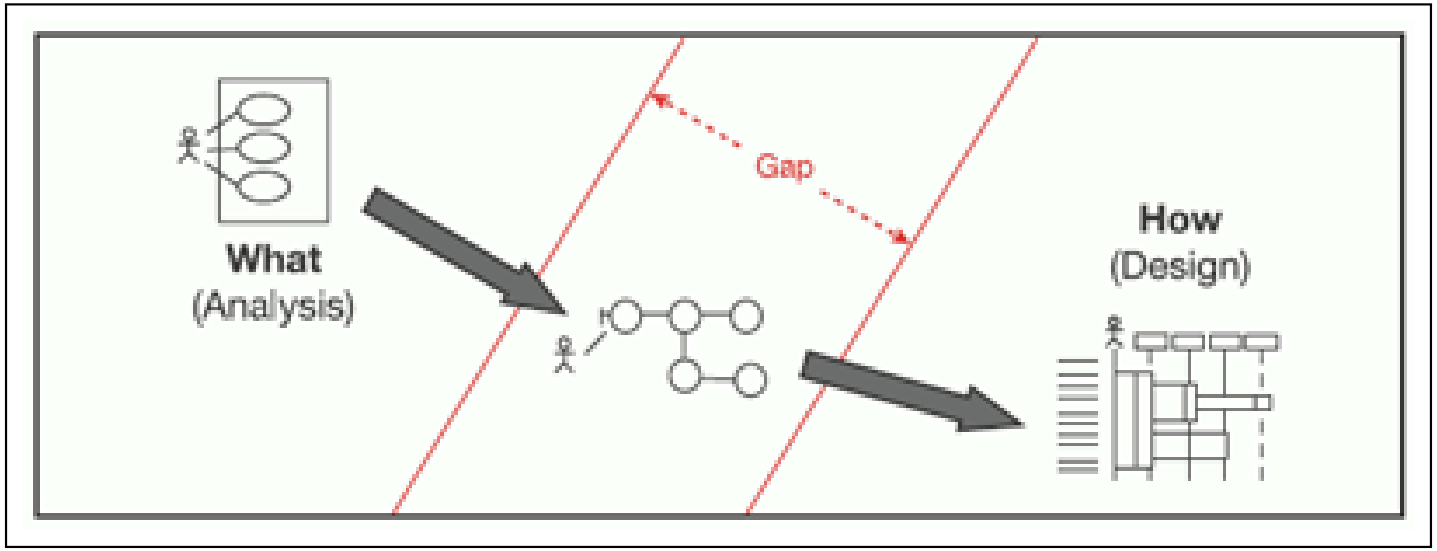
Basic Kinds of Responsibilities

To know
To do
To decide

OOSD — Is Not Seamless

Not every software object is an OOA entity

The Representation Gap

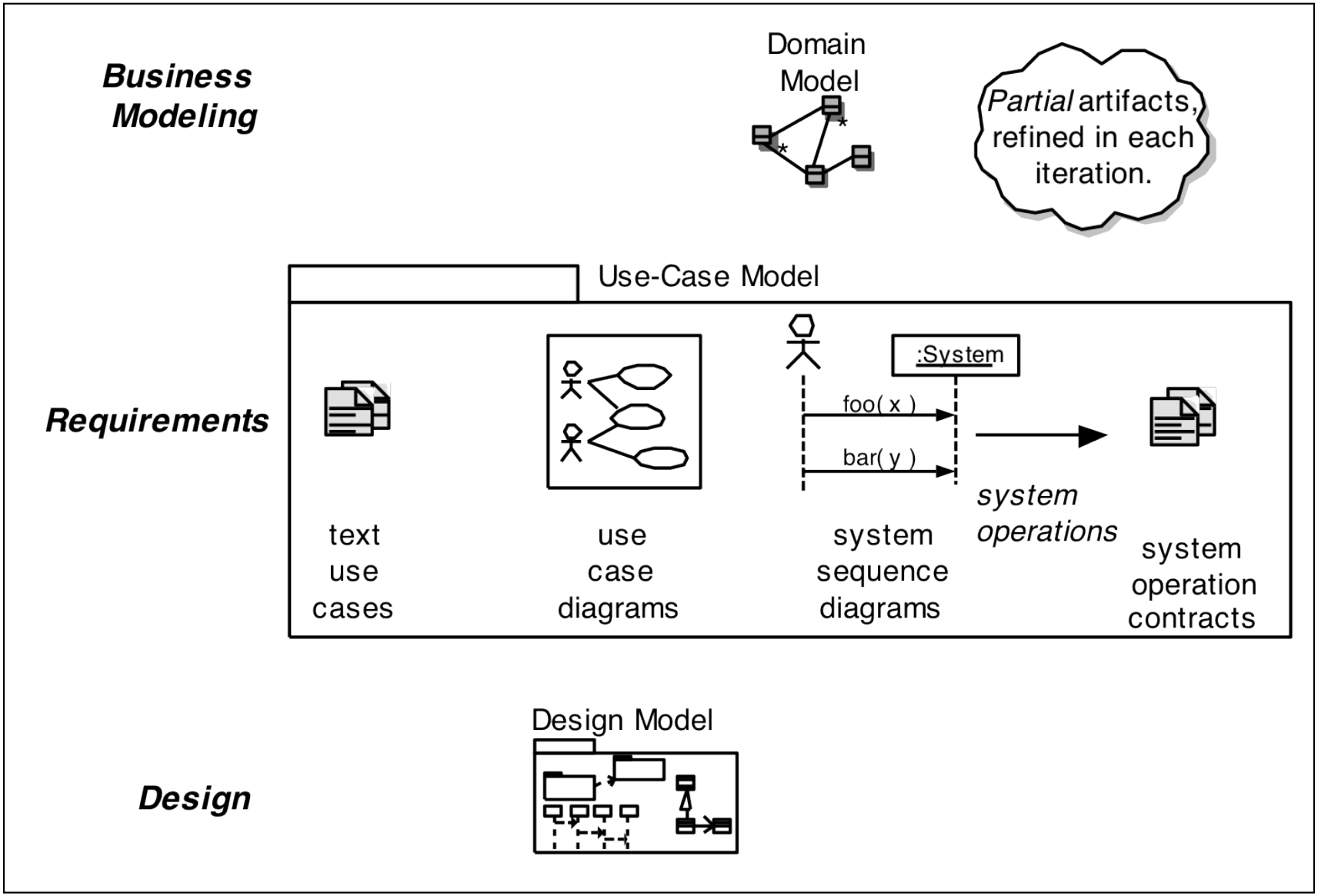


Larman GRASP Patterns

GRASP is General Responsibility Assignment Software Patterns

Guidelines for how to "jump" the representation gap

OOSD — Larman


Business Modeling
Domain Model
*
*
Partial artifacts, refined in each iteration.
Use-Case Model
Requirements
:System
foo(x)
bar(y)
text use cases
use case diagrams
system sequence diagrams
system operations
system operation contracts
Design Model
Design

History of Design

Functional Design

Issue-Driven Design; Scenario-Driven Development

Stepwise Refinement

Modularization

Design by Contract

OO Design

Responsibility-Driven Design

Aspects