Model-Driven Software Development

McGill

Overview

- Motivation
- Course goals
- Course info
- Background on me :)
- Textbooks
- Grading
- Questionnaire

Motivation (1)

- Building software is a challenging task
  - Modern software assumes more and more responsibility, and offers more and more features
    - Distribution
    - Concurrency
    - Interaction with other software and systems
- Developing reliable and dependable software by starting to write code is a thing from the past
- We need to look at different aspects of the software under development at different levels of abstraction in order to:
  - Understand what we need to develop
  - Understand consequences of decisions we make as developers
  - Prove properties of our solution
  - Plan for testing

McGill

Motivation (2)

- UML (Unified Modeling Language) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system
  - Combines ideas from Booch, OMT and OOSE methods
- UML defines lot of different diagrams, and there are lots of ways of using them
- UML is only a notation
- To develop software, we need a process
  - And hence have to choose a coherent subset of the UML diagrams in order to model the software under development at different stages of the development life cycle

McGill

Model-Driven Engineering

- Conceptual framework in which models are at the heart of software development
- Models represent different views of the system under construction, at different levels of abstraction, using different formalisms / notations, for different purpose
  - Use the most appropriate notation to express the relevant concerns
- Models are connected by model transformations
  - High-level specification models are refined / combined / transformed to include more solution details until the produced models can be executed

McGill

Course Goals

- Understand and apply the ideas of model-driven engineering
- Learn how to develop software following an object-oriented, model-driven development process
  - Requirements Elicitation
  - Requirements Specification and Analysis
  - Design
  - (Implementation)
- Master UML and OCL (Object Constraint Language)
  - Learn about other modelling notations, e.g. AoURN, RAM
- Learn about Aspect-Orientation
  - Learn about Aspect-Oriented Modelling

McGill

Course Outline (1)

- Overview
  - Introduction
  - UML and Fondue Method Overview
  - Object-Oriented Technology, Aspect-Orientation
- Requirements Elicitation
  - Use Cases
  - URN, AoURN
  - Domain Model
- Requirements Specification and Analysis
  - Concept Model
  - Environment Model
  - Protocol Model
  - Operation Model + OCL
  - Consistency checks
  - Extension for client-server systems

McGill

Course Outline (2)

- Design
  - Interaction Model
  - Dependency Model
  - Design Class Model
  - Inheritance Model
  - Principles of Good Design / Design Patterns
- Aspect-Orientation
  - Aspect-Oriented Design
- Implementation
  - Implementation Class Model
  - Mapping to Java
- CORE Process (if time permits)

McGill

Course Info

- Pre-requisites:
  COMP-335 or COMP-361 or ECSE-321 or my consent
- Course hours:
  - Monday, Wednesday: 2:35 - 3:55
- Course webpage:
  - http://www.cs.mcgill.ca/~joerg/SEL/COMP-533_Home.html
  - Lecture Schedule, Slide and Assignment Handouts

McGill

About Me

Jörg Kienzle

McConnell Engineering, room 327

Email: Joerg.Kienzle@mcgill.ca

Phone: (514) 398-2049

Office hours:

Monday: 9:30 - 11:00

+ any other time

(send email)

McGill

My Background

- Born in Princeton, NJ, USA
- German parents
- Grown up in Basel, Switzerland (German speaking part)
- Studied at the Swiss Federal Institute of Technology, Lausanne (French speaking part)
- Married to a Canadian Girl

McGill

My Interests in a Nutshell (1)

- Concern-Oriented Software Development (COSD)
  - Concerns are the main focus during software development
- COSD builds on
  - Model-driven Development
  - Reuse
  - Separation of Concerns
- Model Transformation Technology
  - Metamodelling
  - Model interfaces
  - Model customization
  - Model weaving
- Aspect-Oriented Modelling / Aspect-Oriented Programming

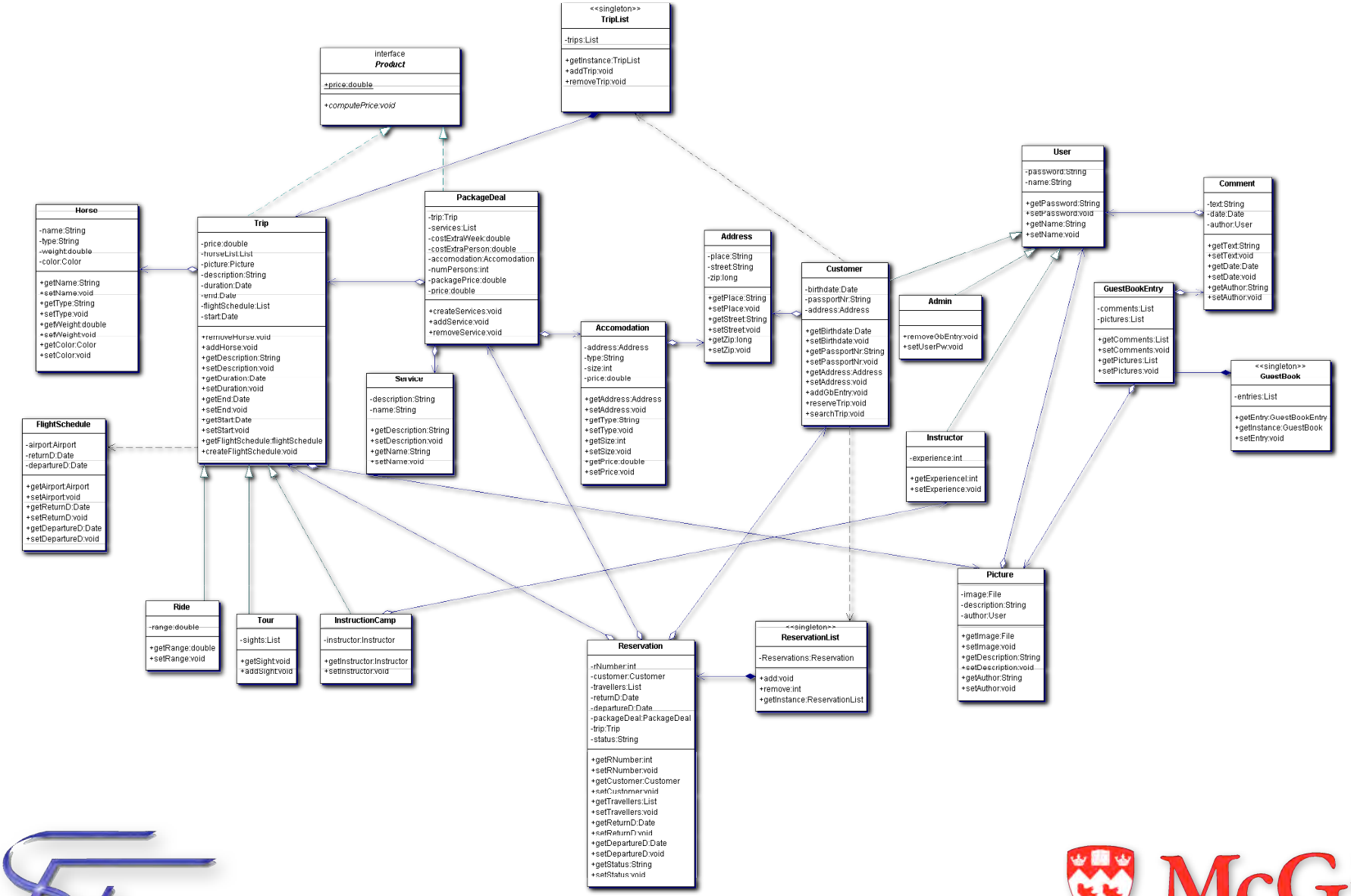
McGill

My Interests in a Nutshell (2)

- Fault tolerance
  - Integrating the concern of fault tolerance into the software development cycle
    - Determine the need for fault tolerance at the analysis level
    - Choose an appropriate architecture and fault tolerance model during design
  - Providing fault tolerance to the programmer (frameworks, aspect-orientation)
  - Implementing fault tolerance models on top of COTS middleware
  - Fault tolerance in massively multi-player games

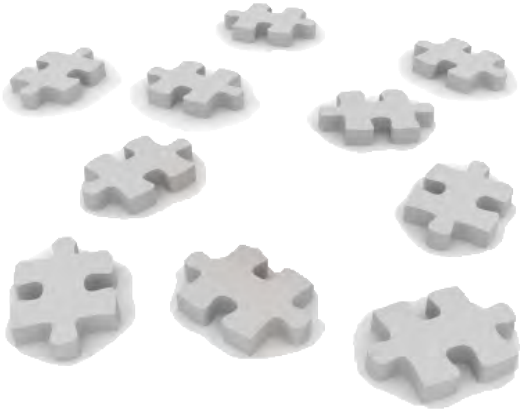
McGill

Modelling and Scalability



Aspect-Oriented Modelling

- Define *modelling notations that allow the modularization of (crosscutting) concerns at the modelling level*

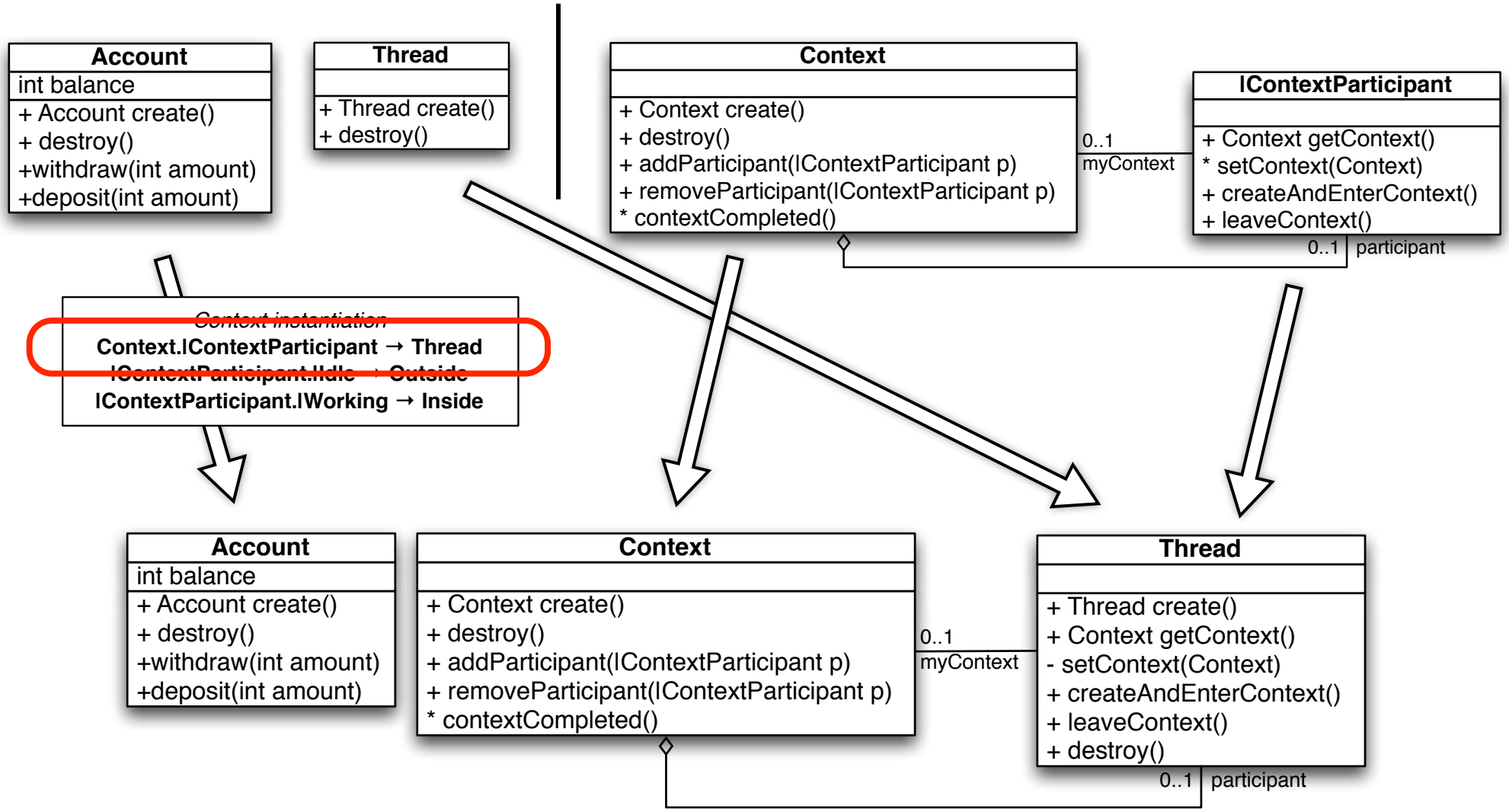


- Define a *model weaving algorithm* to create final application model

➡ For model checking, code generation, simulation / debugging purpose

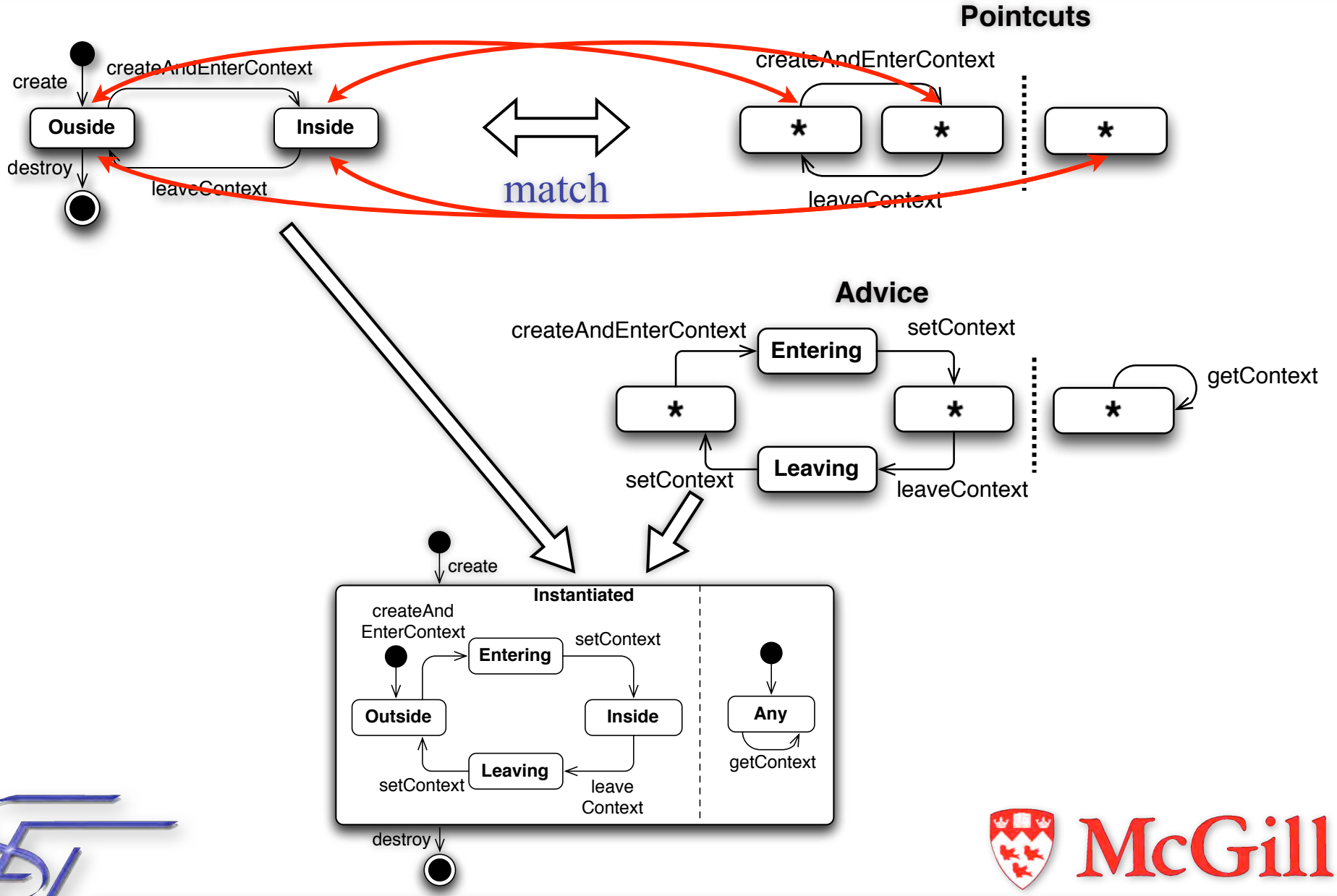
McGill

Class Diagram Weaving

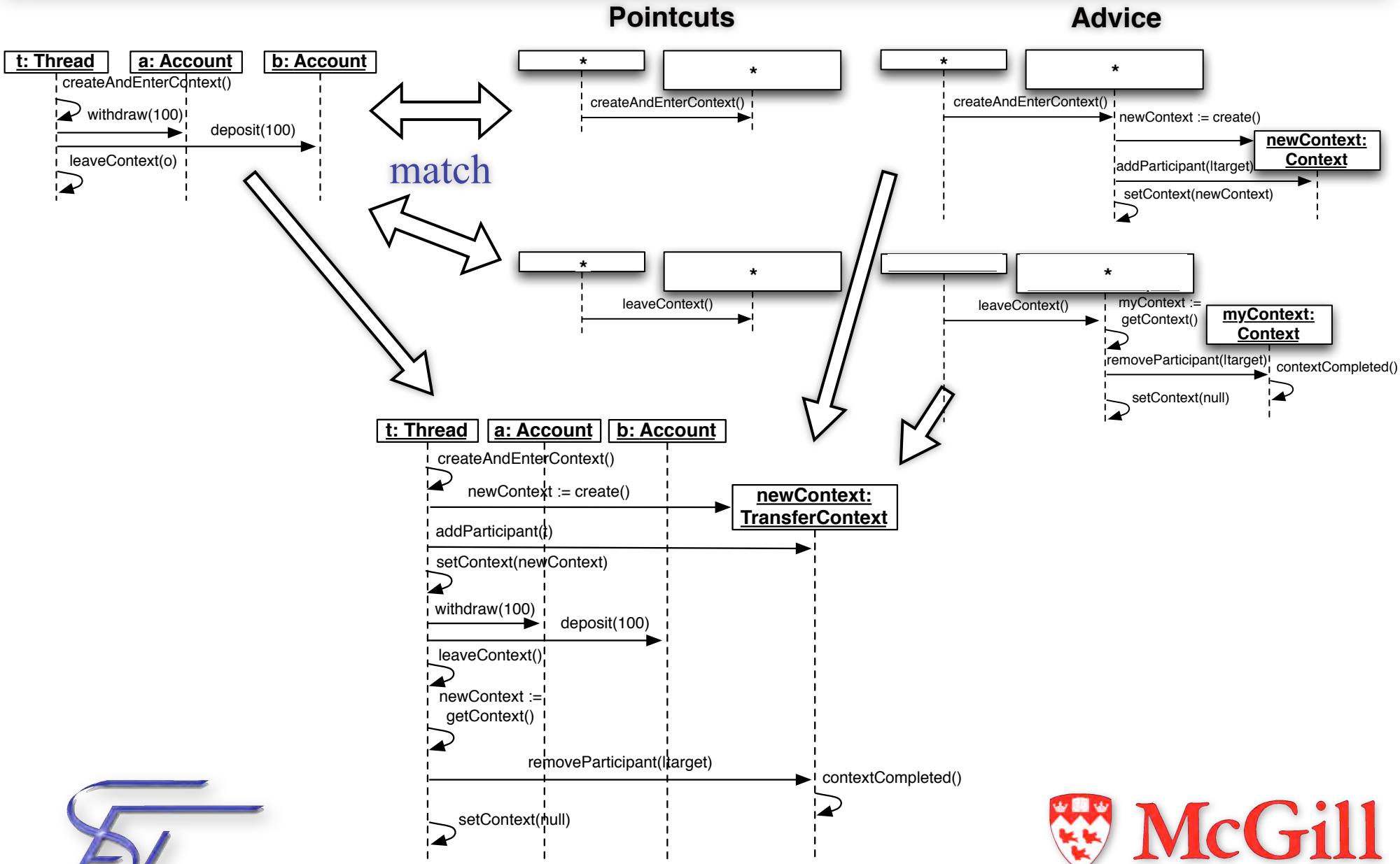


McGill

State Diagram Weaving



Sequence Diagram Weaving



TouchRAM Tool

- Intuitive editing using multi-touch gestures
  - Support for simple (and advanced) gestures
- Simultaneous support for multi-touch (TUIO) as well as mouse / keyboard input
- For more info / download:
  - http://www.cs.mcgill.ca/~joerg/SEL/TouchRAM.html
  - Youtube: http://www.youtube.com/watch?v=l8LMqwwRPg4

- Projects available!
  - Multi-user support
  - State diagram support
  - Feature diagram support
  - Code generation
  - Model versioning

Promises of COSD

- Models are smaller, and thus become easier to understand
- Concerns are modelled separately, and thus easier to understand
- Concern models can be individually reused
- Concern composition is done separately, and thus concern interactions can be better understood
- Concern compositions can be done in multiple different views, and model verification techniques can be used to validate the composed models against each other

McGill

TAs

Omar Alam
McConnell Engineering, room 322
Email: Omar.Alam@mail.mcgill.ca
Office hours: Tuesday 2:30 - 3:30 (or send email)

McGill

Books on using UML for SE (1)

- Craig Larman:
  Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design,
  First Edition, Prentice Hall, 1998.
  ISBN: 0137488807
  - Note: The new second/third edition of the book is based on the Rational Unified Process (RUP) rather than the Fusion process

McGill

Books on Fusion (2)

- Fusion book, english version:
  D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes and P. Jeremaes:
  Object-Oriented Development - The Fusion Method, Prentice Hall, 1994.
- Fusion book, french version:
  D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes et P. Jeremaes:
  Fusion: la méthode orientée objet de 2ème génération, Masson, 1996.

McGill

Books on UML (3)

- James Rumbaugh, Ivar Jacobson and Grady Booch. The Unified Modeling Language Reference Manual, 2nd edition. Object Technology Series, Pearson Higher Education, 2004. (ISBN 0-321-24562-8)
- Warmer, J.; Kleppe, A.: The Object Constraint Language: Getting your models ready for MDA. Second Edition. Object Technology Series, Addison–Wesley, Reading, MA, USA, 2003. (ISBN 0-321-17936-6)
- UML Specification (available for download from the OMG website)

McGill

Grading

- 3 graded homework assignments (3 x 10%)
  - Can be done in groups of 2 students (always different groups - one grade per group)
- Mid-term exam (30%)
  - Individual
- Take-home final (40%) end of November
  - Can be done in groups of 2 students - again different groups

McGill University values academic integrity. Therefore, all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures (see http://www.mcgill.ca/integrity for more information).

McGill

Questions?

McGill

Questionnaire

- For you
  - Evaluate your O-O knowledge
- For me
  - To help me plan the course
- For all
  - Have some fun!

McGill