NEW VERSION OF LECTURE G

CS4125
SYSTEMS ANALYSIS
SPRING SEMESTER 2010-2011

J.J. Collins
Dept of CSIS
University of Limerick

Lecture G (W4L2)

Analysis – Drawing the Class Diagram, Part 1

Analysis: Classes - Part I

What is Analysis?

- During analysis, the software elements necessary to satisfy the functional requirements are identified.
- The analysis model should be domain facing.
- Analysis is commonly referred to as requirements analysis, or conceptual class modelling in the set of object oriented methodologies.

Identifying Classes

- A good class model consists of classes of enduring domain objects which do not depend on particular functionality required today i.e. a sale (made through a web interface).
- One technique: develop an analysis domain model independent of any use cases, see the ICONIX method (Rosenberg with Scott, 1999).
- Or, model the dynamics of each use case.
  - The dynamics (behaviour) is governed by the set of classes taking part in that use case. The dynamics is specified in either a communication or sequence diagram.

Analysis: Classes - Part I

Step 1

- Identifying classes from Use Case Descriptions:
  - Data driven design (DDD): key domain abstraction using noun identification technique.
  - Responsibility driven design (RDD): process oriented.
- Steps
  - List potential candidate classes.
  - Discard inappropriate classes.
  - Separate list into strong and weak classes.

Analysis: Classes - Part I

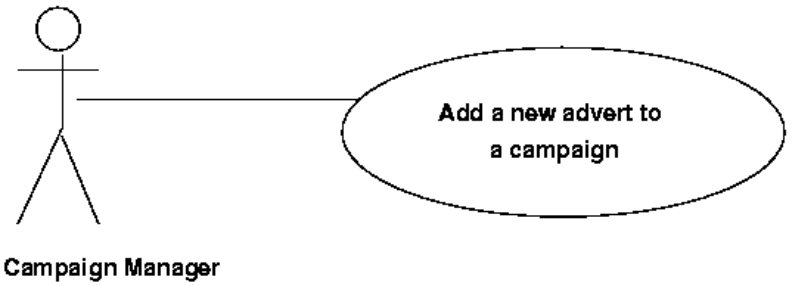
Recap

- Discarding candidates: question - has this class state, behaviour and identity?
  - Too vague or too specific.
  - Redundant.
  - Is it an event ?
  - Meta-language
  - Is it an attribute?
  - Is it an operation?
  - Is it an association?
- Draw the collaboration diagram that realizes the use case.
- Draw a communication diagram – why?
- Avail of Class Responsibilities Collaborations (CRC) cards to determine interactions.

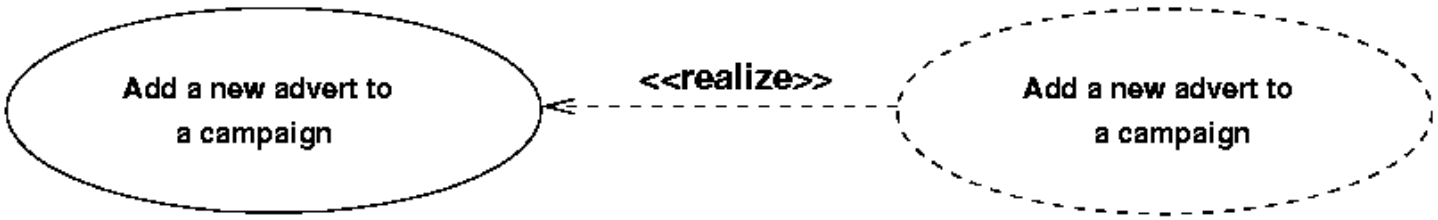
Analysis: Classes - Part I

Step 2

- Use Case Realisation (Dynamic Modelling)

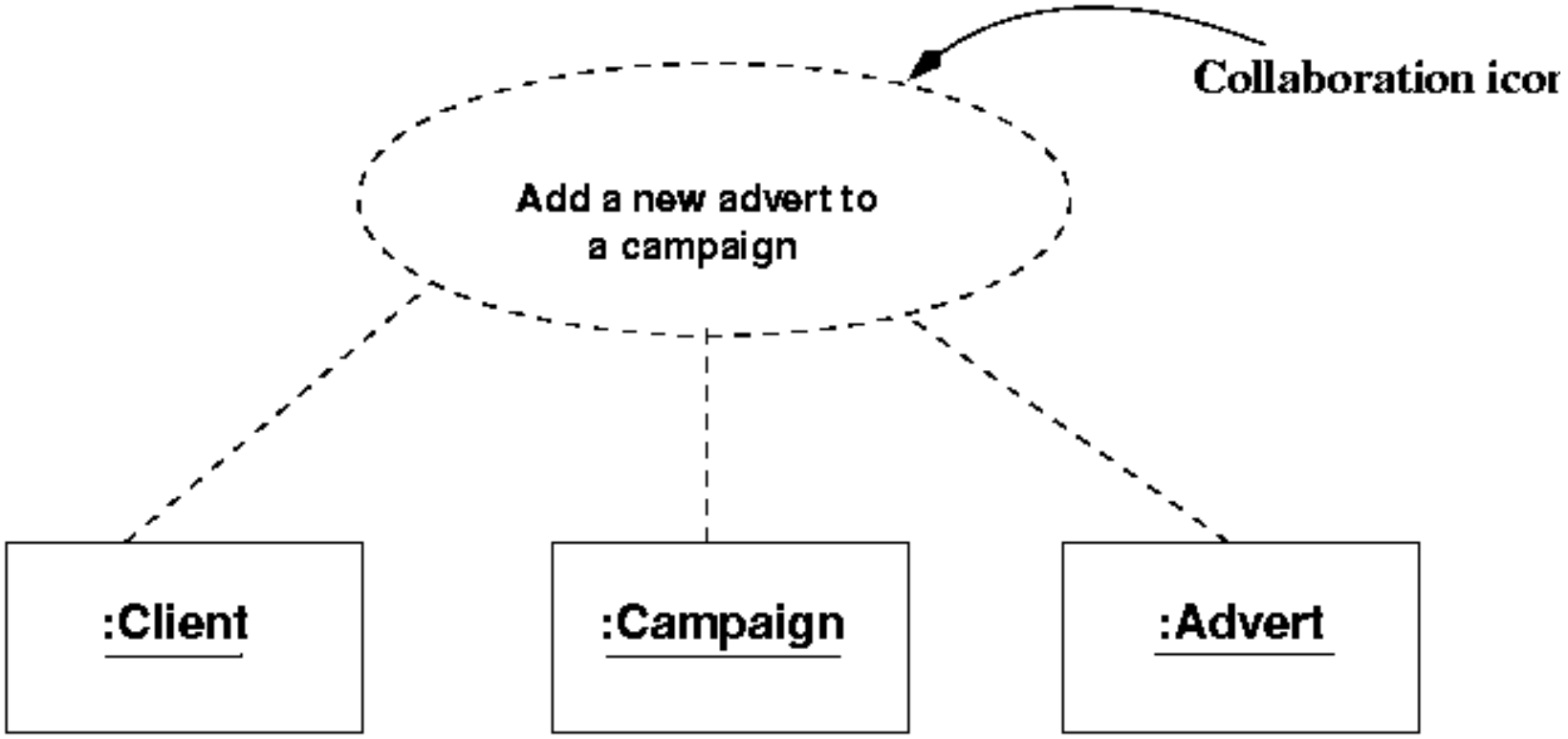


Use Case	Add a new advert to a campaign
	A campaign can consist of many adverts. Details of each advert are entered into the system by the campaign manager.
Primary Actor	Campaign Manager



The dependency arrow indicates the objects within the collaboration may reference elements within the use case

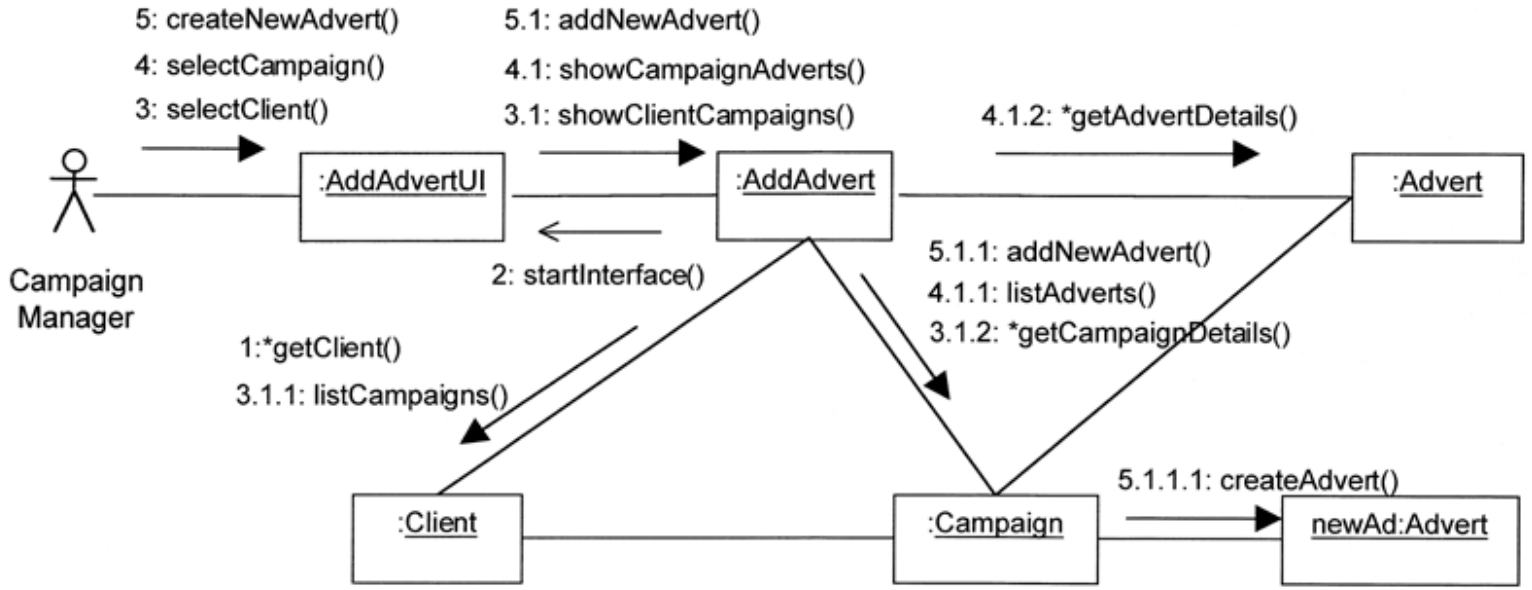
1. Analysis: Classes - Part I



Analysis: Classes - Part I

- Step 3: Communicatuion Diagram

Communication Diagram

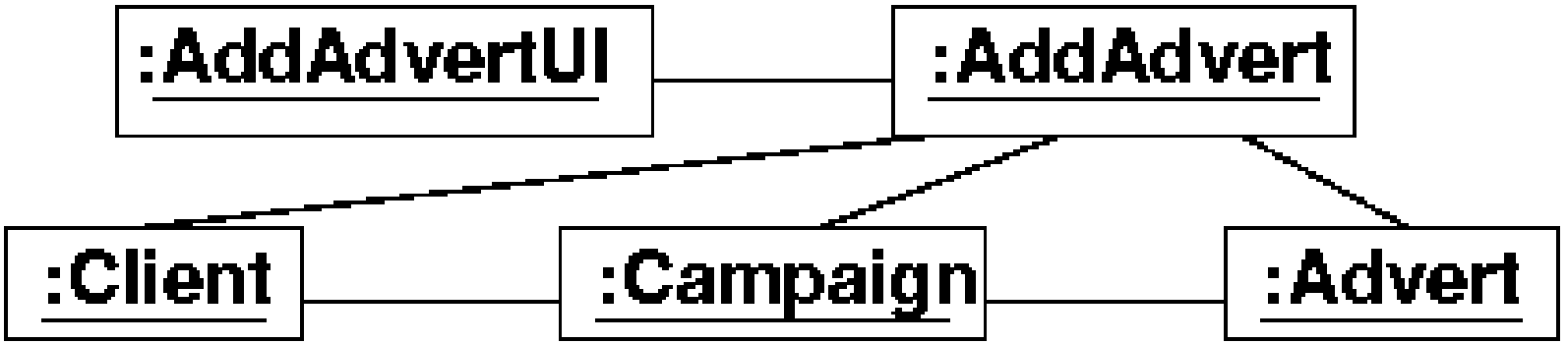


- Step 4: CRC Cards

Class Name: Campaign	
Responsibilities	***Collaborators***
Add a new advert	Advert
List adverts	

Analysis: Classes - Part I

- CRC cards:
  - Too many collaborators: indicates high coupling.
  - Too many responsibilities indicates possibly poor cohesion.
- Note that when modelling dynamics, we always refer to the objects that constitute the runtime system, and not the classes.
- One may wish to depict the set of classes necessary to realise the use case as an object instance diagram.

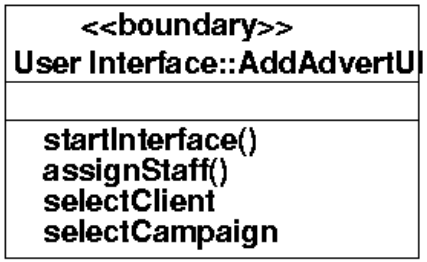


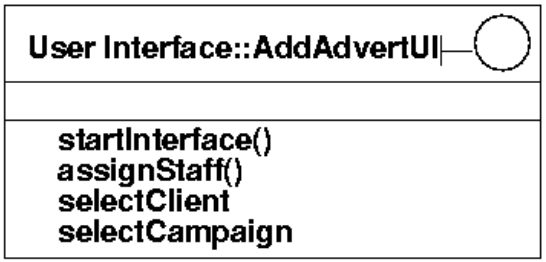
Analysis: Classes - Part I

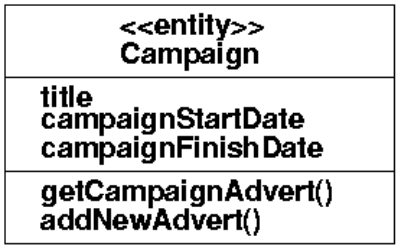
3 commonly used analysis class stereotypes, defined in the appendix to UML, aka Rational Profile:

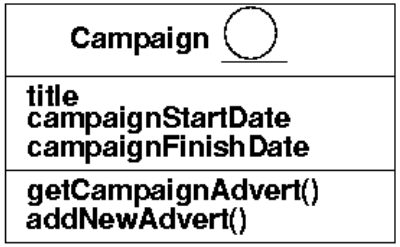
- Boundary classes
  - model interaction between the system and its actors (Jacobson et al. 1999), relatively abstract.
- Entity classes
  - model information and associated behaviour of some domain concept such as individual, a real life object or real life event (Jacobson et al. 1999).
- Control classes
  - represent coordination, sequencing, and transactions control of other objects (Jacobsen et al. 1999).
  - In the USDP, as in the earlier methodology Objectory, it was strongly recommended that there be at least one control class for each use case.
  - Often used to encode business rules or business logic.

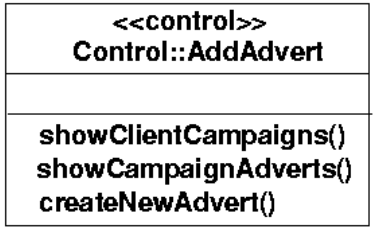
Analysis: Classes - Part I

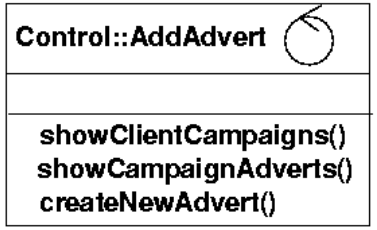


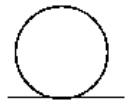








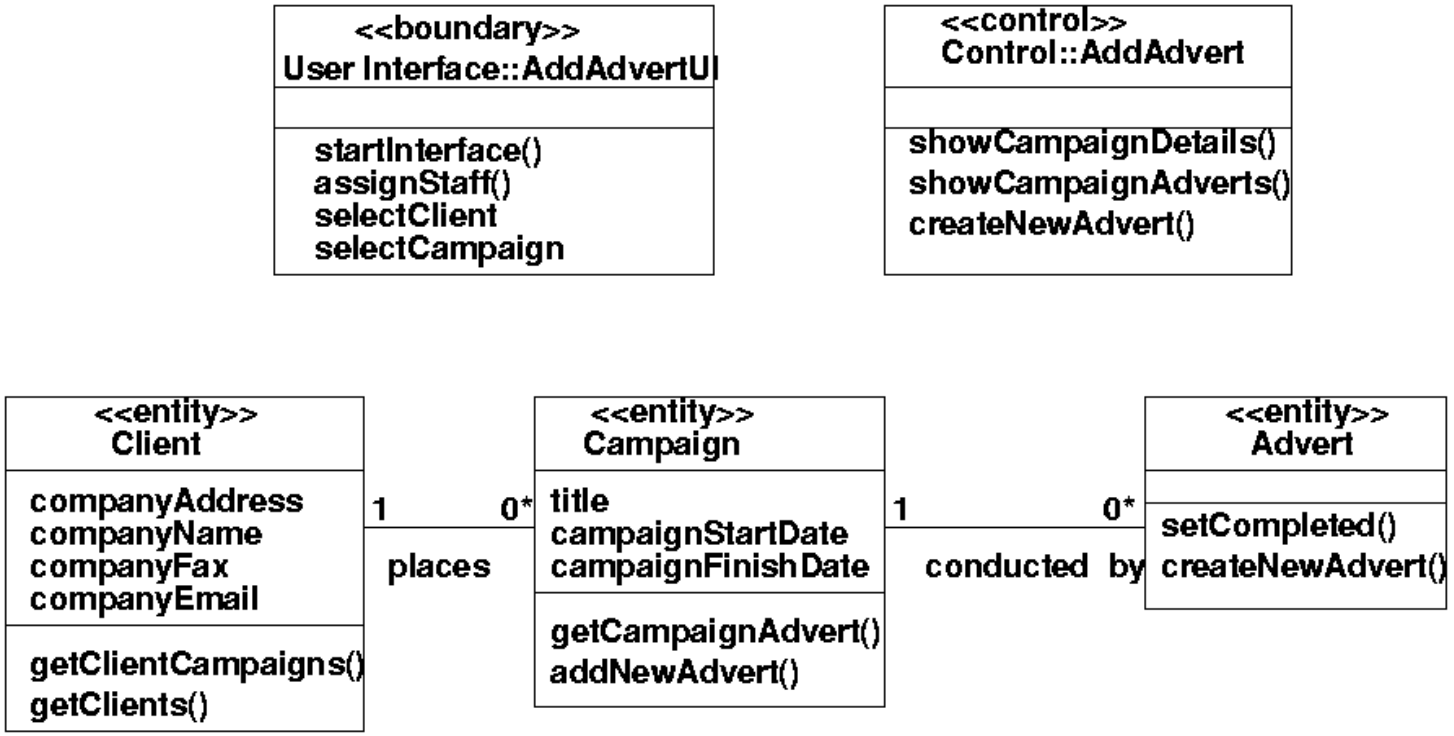




Analysis: Classes - Part I

Step 5

- Fragment of class diagram derived from use case realisation is called use case class diagrams.
- Why are there no associations between boundary, control and entity classes?



Analysis: Classes - Part I

- *Step 6:*
  - *Repeat steps 2 to 5 for each use case*
- *Step 7:*
  - *Create a first cut class diagram by integrating all the use case class diagrams into one.*
- *Step 8:*
  - *Identify the Associations*
  - Should be evident from collaboration and communication diagrams.

- Association Rules: instances of Class A and B are associated if:
  - A sends a message to B.
  - A creates an instance of B.
  - A has an attribute whose values are B or a collection of B.
  - A receives a message with B as an argument.
- More on associations in lecture H

Analysis: Classes - Part I

- Step 9: Identifying Generalisations
- Top down approach: if an association can be described by the expression ``is a `` or ``is a kind of", then it can be modelled as a generalisation.
- Bottom up approach: if classes share attributes and messages, then consider abstracting similarities into a superclass.

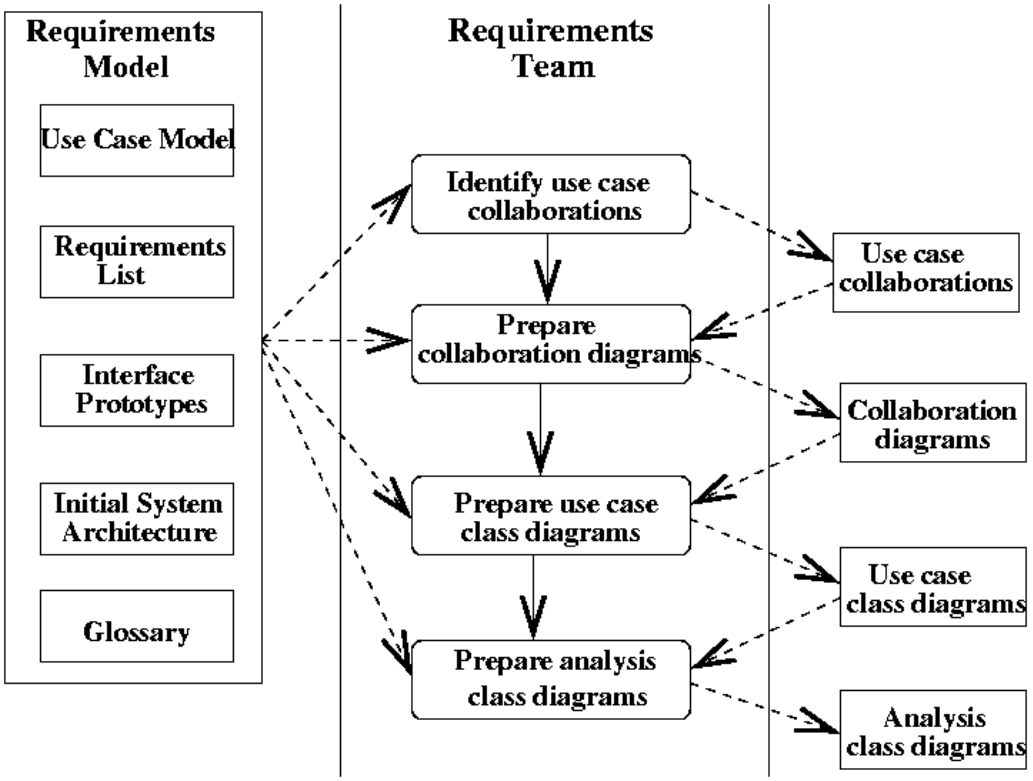
Analysis: Classes - Part I

2 issues w.r.t generalisation

- There must be no conceptual gulf between what objects of the two classes do on receipt message.
- **Behavioural subtyping** (Liskov substitution principle) is a more precise definition of these two statements.
- An object of a specialised class can be substituted for an object of a more general class in any context which expects a member of a more general class, but not the other way around.
- Suppose that a program expects to interact with an object that is an instance of class C, and that instead it is given an object s of class S, a subclass of C.
- If the Liskov substitution holds, the program will execute correctly at both the explicit syntactic and implicit semantic level.
- ***Fragile base class*** *problem*: coupling because of subclass's dependency on superclass. Changes to superclass may force recompilation of subclass.

Analysis: Classes - Part I

- Refine the use case class diagram to yield an analysis class diagram.
  - For each class, identify all its attributes and operations using CRC cards.
  - For each operation, specify its signature: return type, selector and parameter (argument) list.
- Analysis class diagram, aka conceptual class model



10. Reading

- Chapter 7 in Bennett et al.