


Ontology Design Patterns and XD

Eva Blomqvist

eva.blomqvist@liu.se



scarlet

Relation Discovery on The Semantic Web

How does relate to ?

Relation types:

- ☒ All Types
- ☐ Inheritance
- ☐ Disjointness
- ☐ Named Relations

Strategy:

- ☒ Use one ontology
- ☒ Use more ontologies

Other parameters:

- ☒ Find first relation
- ☐ Find all relations
- ☐ Use inheritance depth

Examples:

- River vs. waterway
- Cocaine vs. narcotic
- Water vs. Solid
- Branch vs. Tree
- Coal vs. Industry
- Fish vs. Lobster
- Cholesterol vs. OrganicChemical
- Apple vs. Meat

city and **country** appear together in 54 ontologies.

The following relations were found:

- city subClass country
Because: city - subClassOf -> country
City - subClassOf -> County
In: http://www.simondfraser.co.uk/geo_ont.daml
County - subClassOf -> Country
In: http://www.simondfraser.co.uk/geo_ont.daml

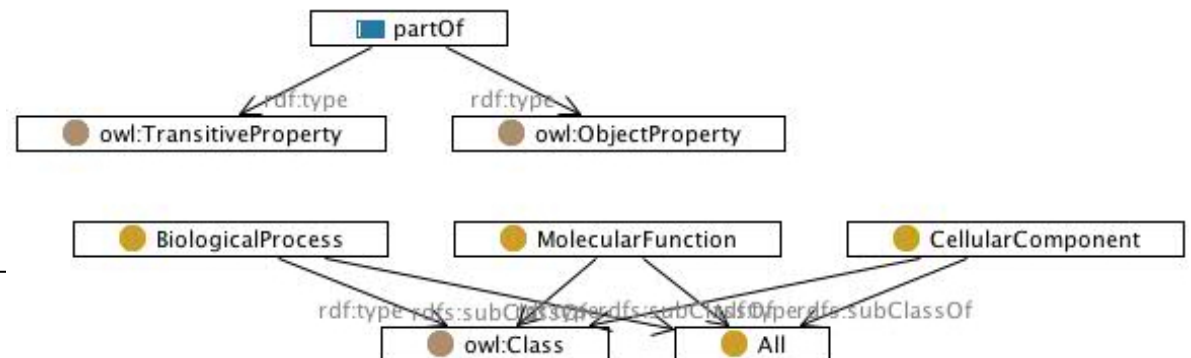
KMi

What we can do with OWL

- ... (maybe) we can check the consistency, classify, and query our knowledge base
- ... but, remember the Scarlet example
 - `City subClassOf Country`
- Logical consistency is not the main problem
 - e.g. `rdfs:subClassOf` can be wrongly used and still we have consistency
- Why is OWL not enough?
 - OWL gives us logical language constructs, but does not give us any guidelines on how to use them in order to solve our tasks.
 - E.g. modeling something as an individual, a class, or an object property can be quite arbitrary

Solutions?

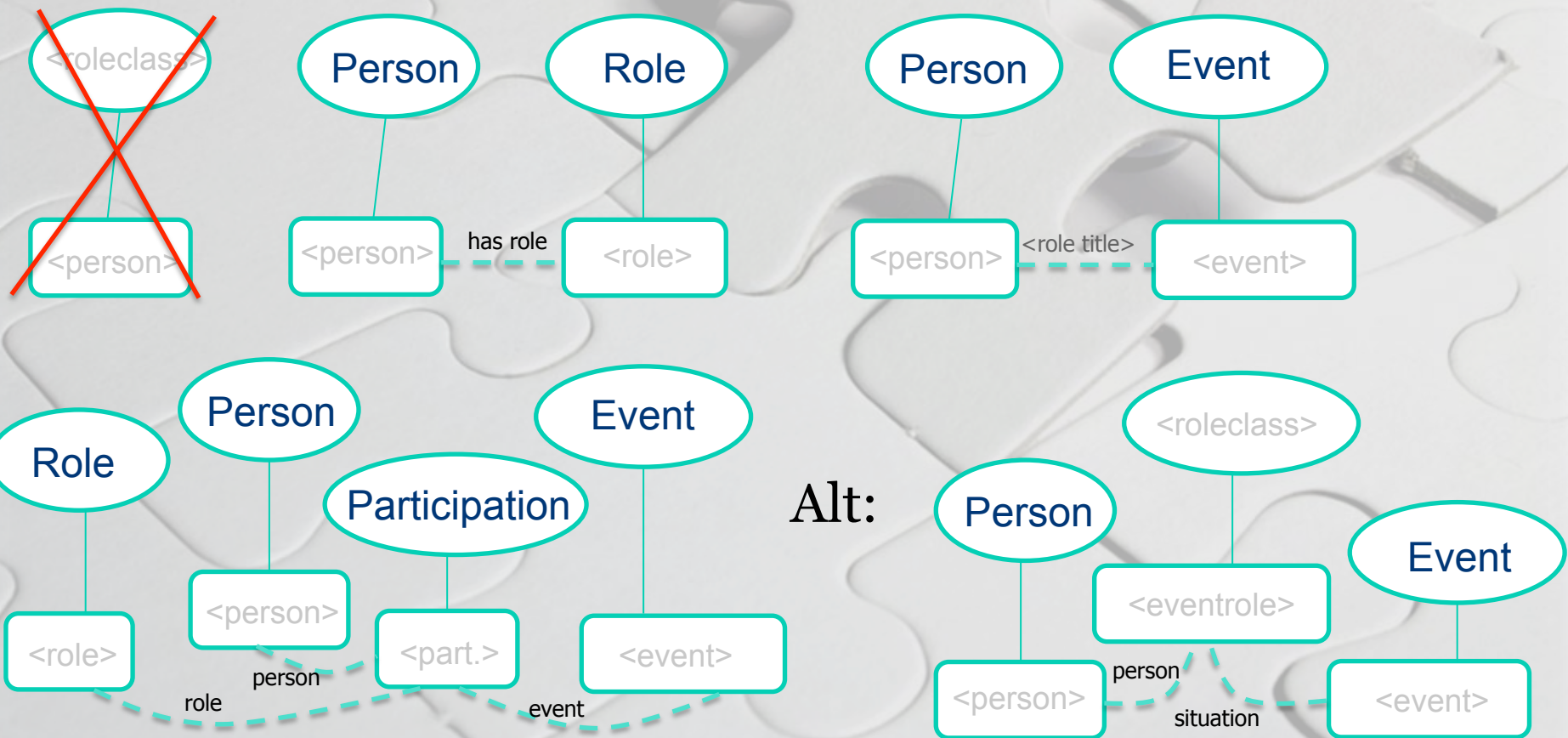
- OWL is not enough for building a good ontology, and we cannot ask all web users neither to learn logic, or to study ontology design
- Reusable solutions are here through Ontology Design Patterns, which help reducing arbitrariness without asking for sophisticated skills ...
- ... provided that tools are built for any user ☺



Various types of ODPs

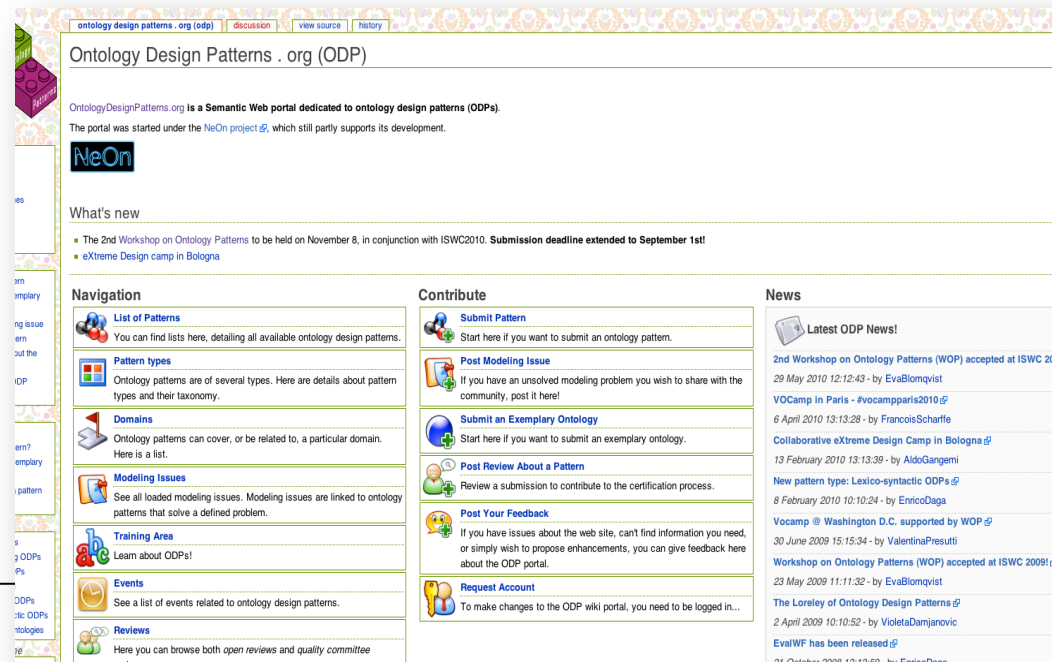
- Logical patterns – "workarounds" and shortcuts in modelling
 - Example: n-ary relations
- Content patterns – components with a non-empty signature, sometimes domain specific
 - Example: how to model roles
 - Can be used as "templates" or ideas for your own solution, or as components that are specialised
- Correspondence patterns, transformation patterns...

Example - Role patterns (ODP)



Catalogues of ODPs

- Content ODPs are collected and described in catalogues, books, papers...
- The ontologydesignpatterns.org initiative maintains a repository of ODPs



The eXtreme Design methodology

Ontology Engineering Methodologies

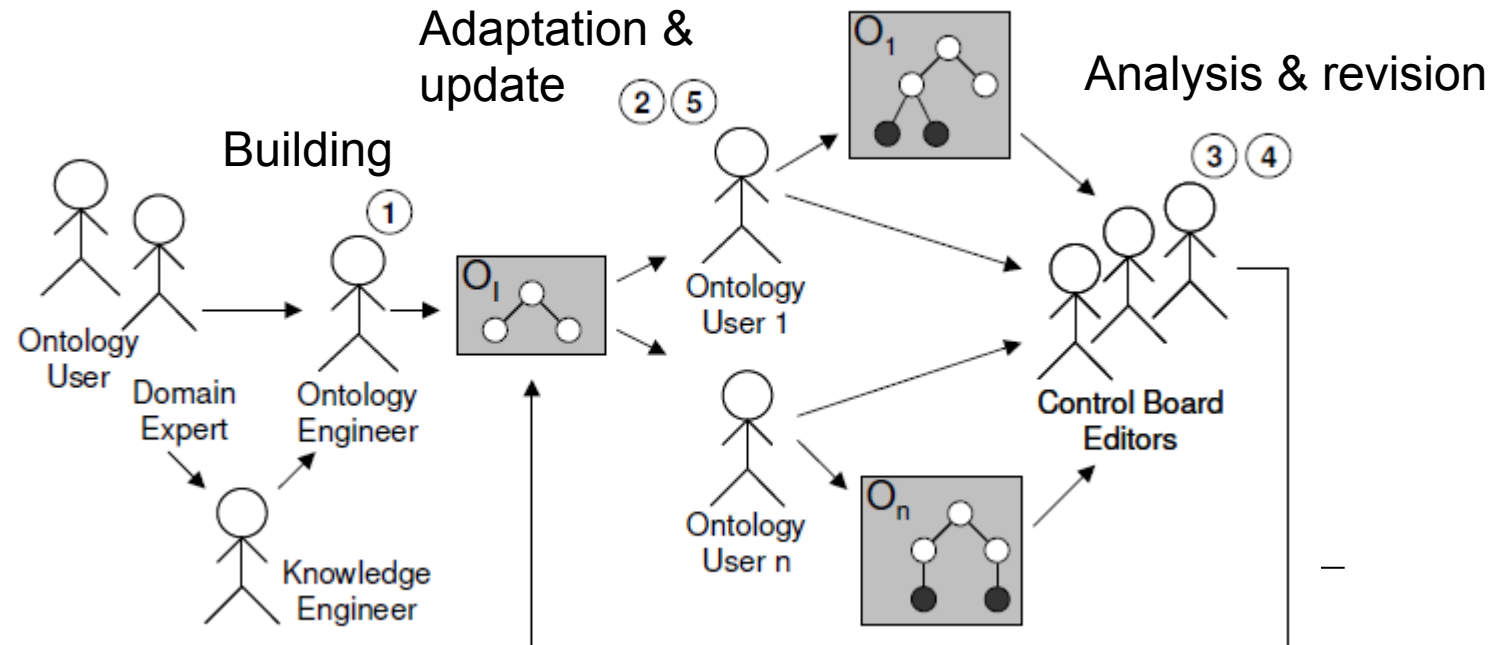
- Mostly focus has been on overall life-cycle and “model” of the methodology – rather than *how* to actually perform it
- Few are focused on reuse and the networked nature of web ontologies
- One of the most cited:
 - Ontology development 101 – Noy & McGuinness (2001)
 - Pre-OWL methodology
 - Traditional in the sense
 - It doesn’t have a specific task focus
 - It is a waterfall like method
 - Although detailed in some steps, no details on requirements or testing etc.
 - Basic steps for modelling
 - (1) Domain an scope
 - (2) Consider reuse
 - (3) Enumerate terms
 - (4) Develop class hierarchy
 - (5) Define the properties
 - (6) Define restrictions and constraints
 - (7) Create instances

Example: METHONTOLOGY (~1997)

- Waterfall-like process consisting of (overlapping) phases
 1. Specification – document requirements, scope, level of formality etc.
 2. Knowledge Acquisition – gathering and studying sources of information
 3. Conceptualization – structure the terminology identified in 1, going from glossary to logical formulas
 4. Integration – find and select other ontologies to reuse
 5. Implementation – represent in formal language using tool
 6. Evaluation – verification and validation
 7. Documentation

Example: DILIGENT (~2004)

- Based on theories for argumentation
- Intended for
 - Empowering domain experts in ontology engineering
 - Continuous and distributed construction and update



Why the name “XD”?

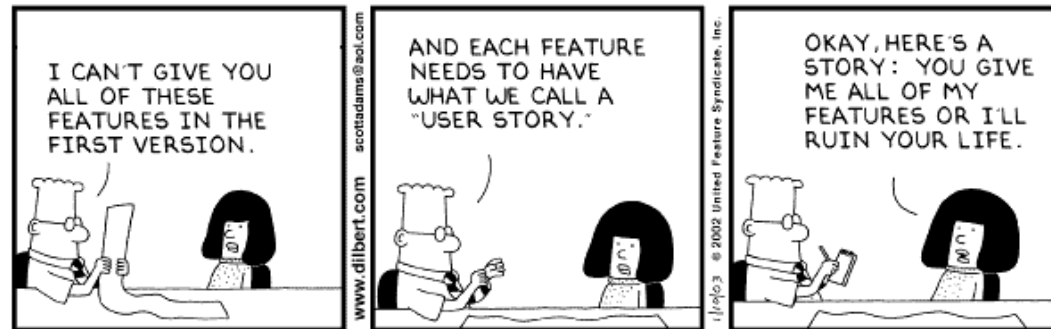
- Inspired by XP but with focus on good design
- An agile methodology for web ontology design
- Developed as part of the NeOn methodology



Copyright © 2003 United Feature Syndicate, Inc.

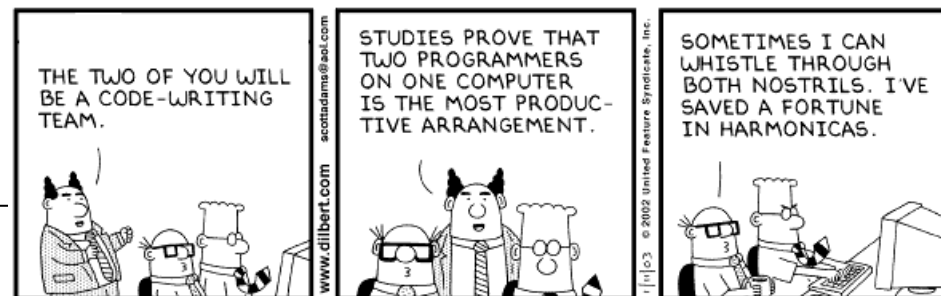
XD principles

- Customer/domain expert involvement and feedback
- "Customer" stories to derive CQs (+ restrictions/constraints, reasoning requirements)



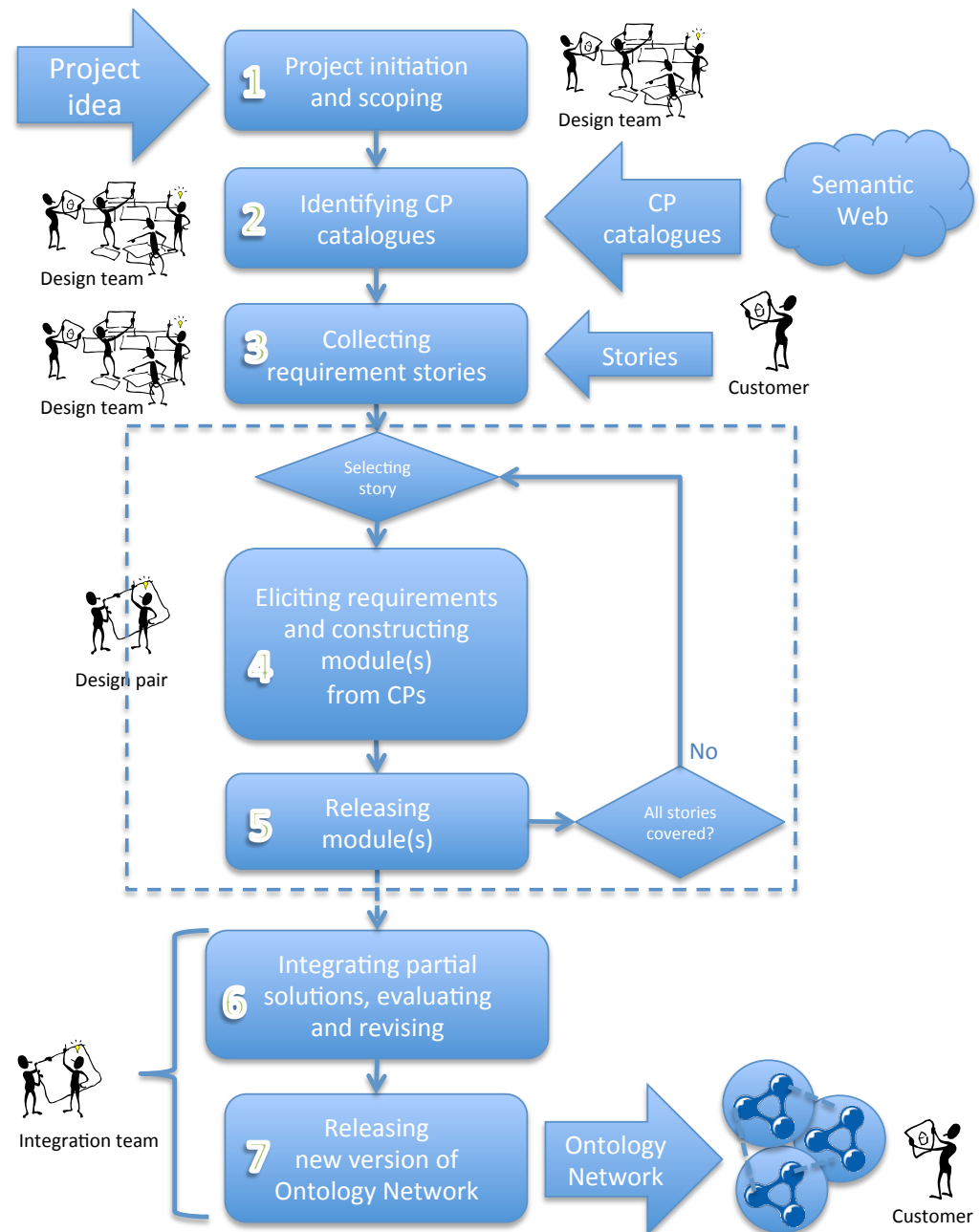
Copyright © 2003 United Feature Syndicate, Inc.

- ODP reuse and modular design (ontology networks)
- Collaboration and integration
- Task-oriented design, verified by tests
- Pair design

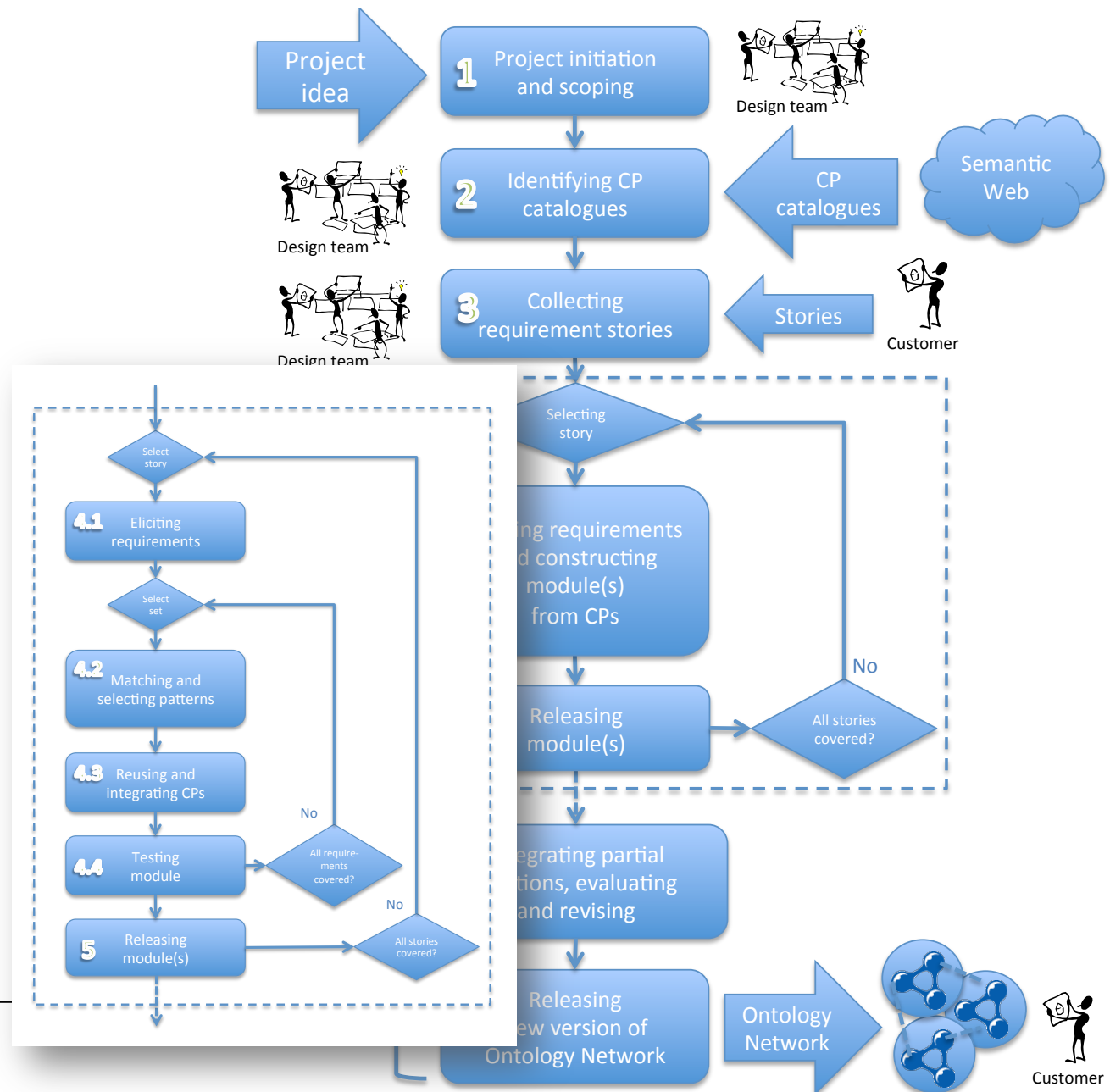


Copyright © 2003 United Feature Syndicate, Inc.

XD Iteration



XD Iteration



Things to note about XD

- Can be adapted to various settings
 - Pairs or individual development?
 - Roles of ontology engineers and other experts
 - Adapt the level of communication and control
- You quickly have a tangible result
 - Rapid prototyping of ontologies?
- Integration step is crucial and may involve lots of refactoring

www.liu.se