Requirements-Driven Software Service Evolution

Feng-Lin Li [1], John Mylopoulos [1] , Lin Liu [2],

1: Dept of Information Engineering and Computer Science, University of Trento, Italy
2: Software School, Tsinghua University, China

2012-12-20

Outline

- *Background*
- *Problem Statement*
- *The Feature-Oriented Approach*
- *A Case Study – EShop*
- *In Summary*

Service Evolution

- ***Service Evolution****: a continuous process of development of a service through consistent and unambiguous changes to requirements and domain assumptions.*
- *Key Challenges:* ***Forward compatibility****: a guarantee that an older version of a client application should be able to interpret and use newer message/data formats introduced by the service.*

Related Work

- *Taxonomy of evolutionary changes*
  - *Structural (interface), Behavior (interaction protocol), QoS*
- *Versioning*
- *Design Pattern*
  - *Dynamic binding, Client transparency*
- *Tool (Adaptor, Proxy)*
- *Model and Theory*
  - *Type theory*

Observations

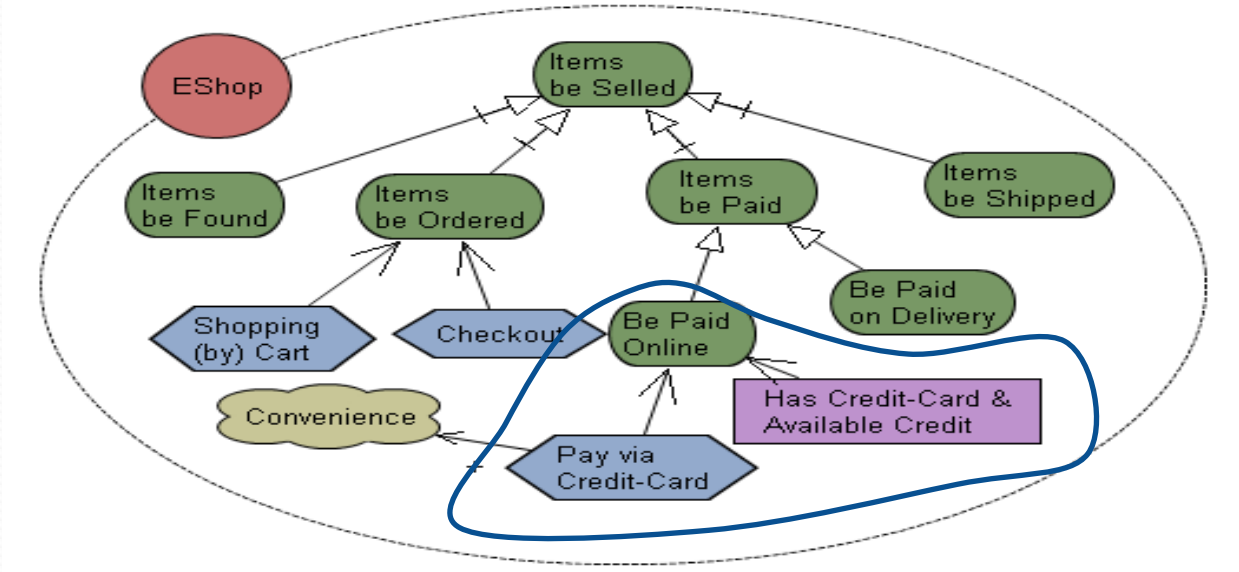
- Evolutionary changes are closely related to service interface (signature) and behavior (interaction protocol) [Li12].
- Current work
  - Focuses on the incompatibility between clients and servers
  - Pays little attention to **change propagation** from requirements to services

Starting Point: Requirements

- *D*, *S* |= *R* [Jackson&Zave95]
  - *D*: domain assumption, *S*: specification, *R* : requirement
  - E.g., Task "*Process Credit Card Payment*"(*S*),

    Domain Assumption "*Customer owns a Credit Card with Available Credits*"(*D*)

    |= Goal "*Item(s) be Paid Online*"(*R*).

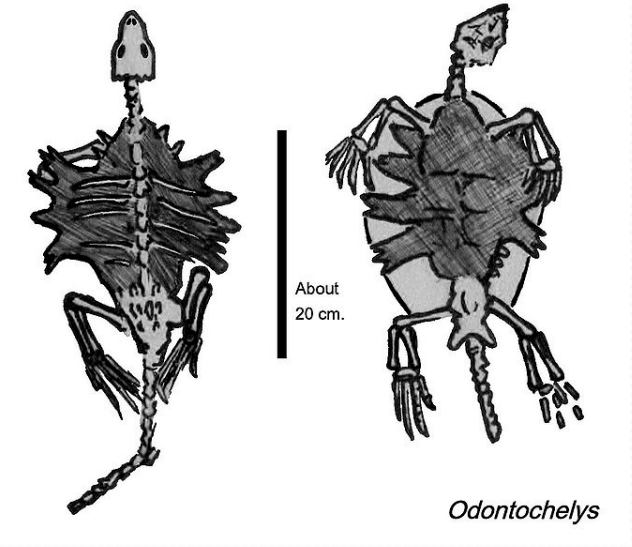


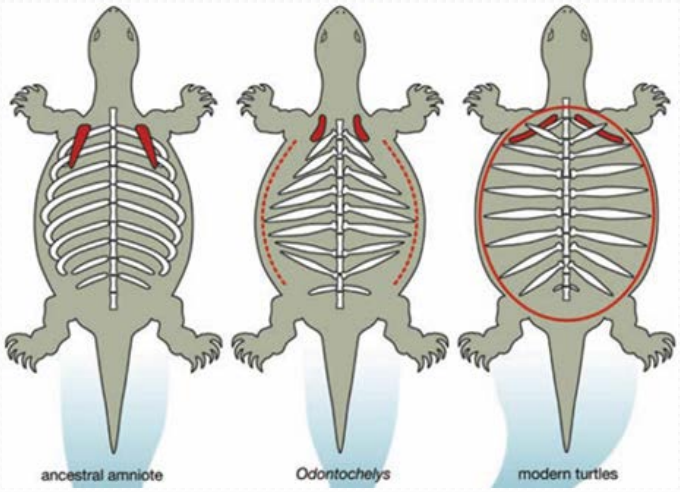
Problem Statement

- ***Prerequisite***: $D, S \models R$
- ***Propagation***:
  - (*Requirements Change*) if R changes to R', how to find a new specification S' so that $D, S' \models R'$ holds?
  - (*Environment Change*) if the domain assumption D changes to D', how to find a new specification S', so that $D', S' \models R$ still remains true?
- ***Traceability:***
  - (*Specification Change*) if specification S changes to S', does the entailment $D, S' \models R$ still remain true?

The Feature-Oriented Approach: A Motivating Example

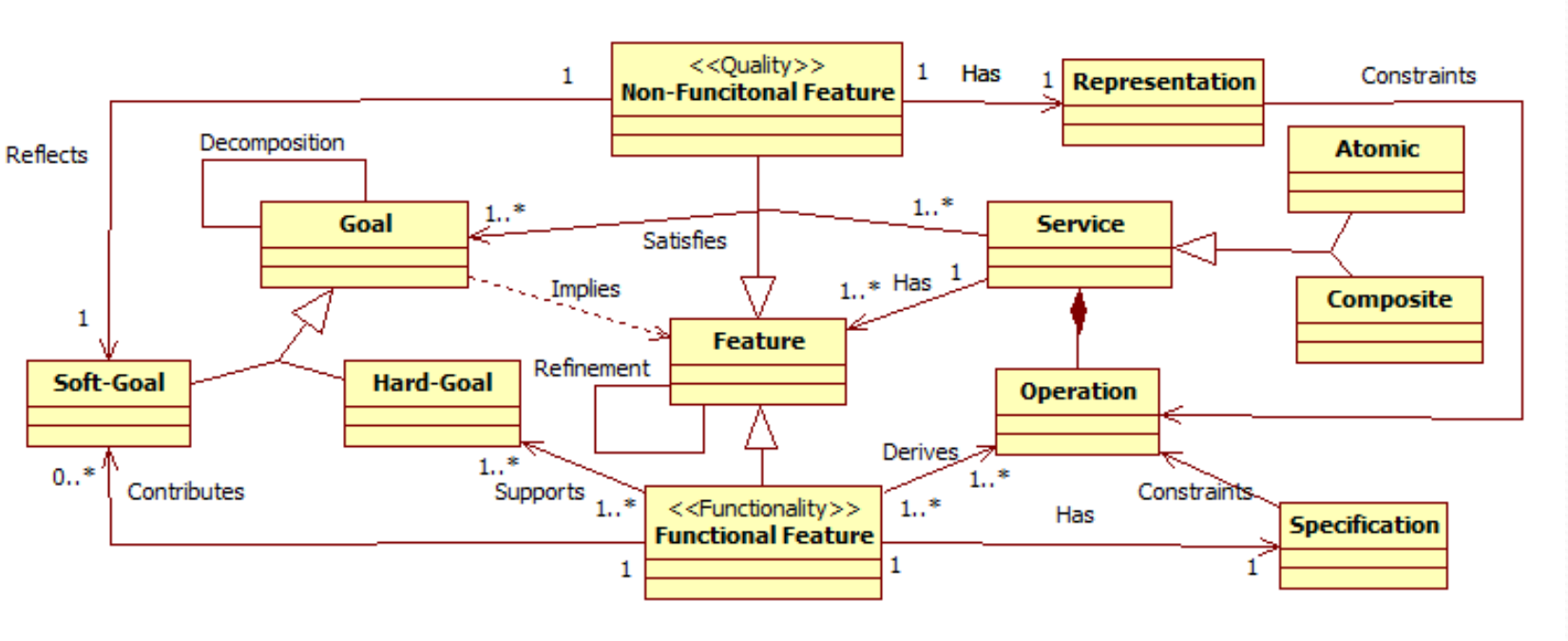
- *The evolution of turtle shell*
  - *Odontochelys (oldest turtle)*
  - *The turtle shells formed from the underside - plastron (chest) first*
  - *And then grew bony extensions of ribs and bone formation above backbones*
  - *Existing features are modified and put into second use.*





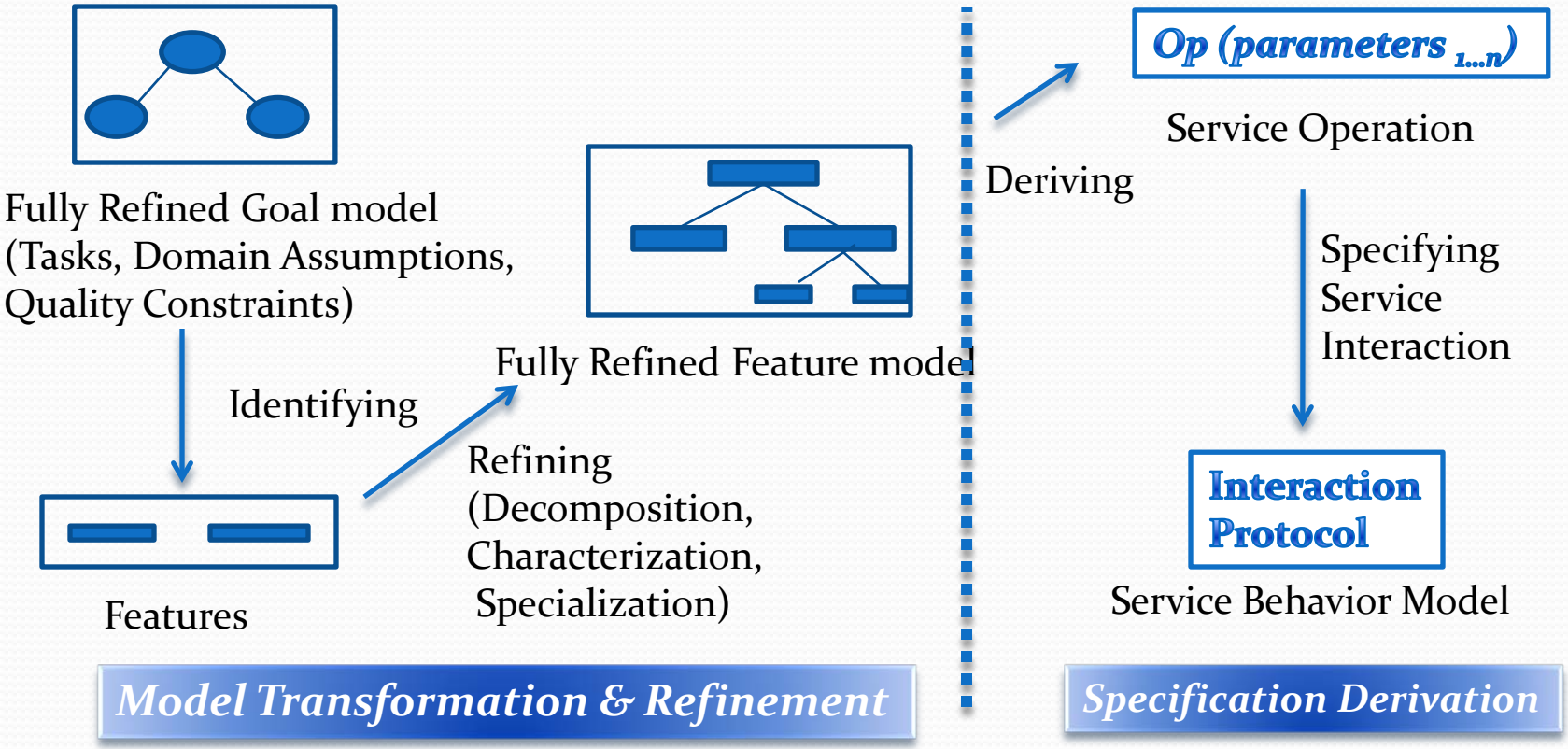
Services as Feature Configurations

- A service consists of features (functional, QoS)



The Feature-Oriented Approach

- Methodology
  - Framework

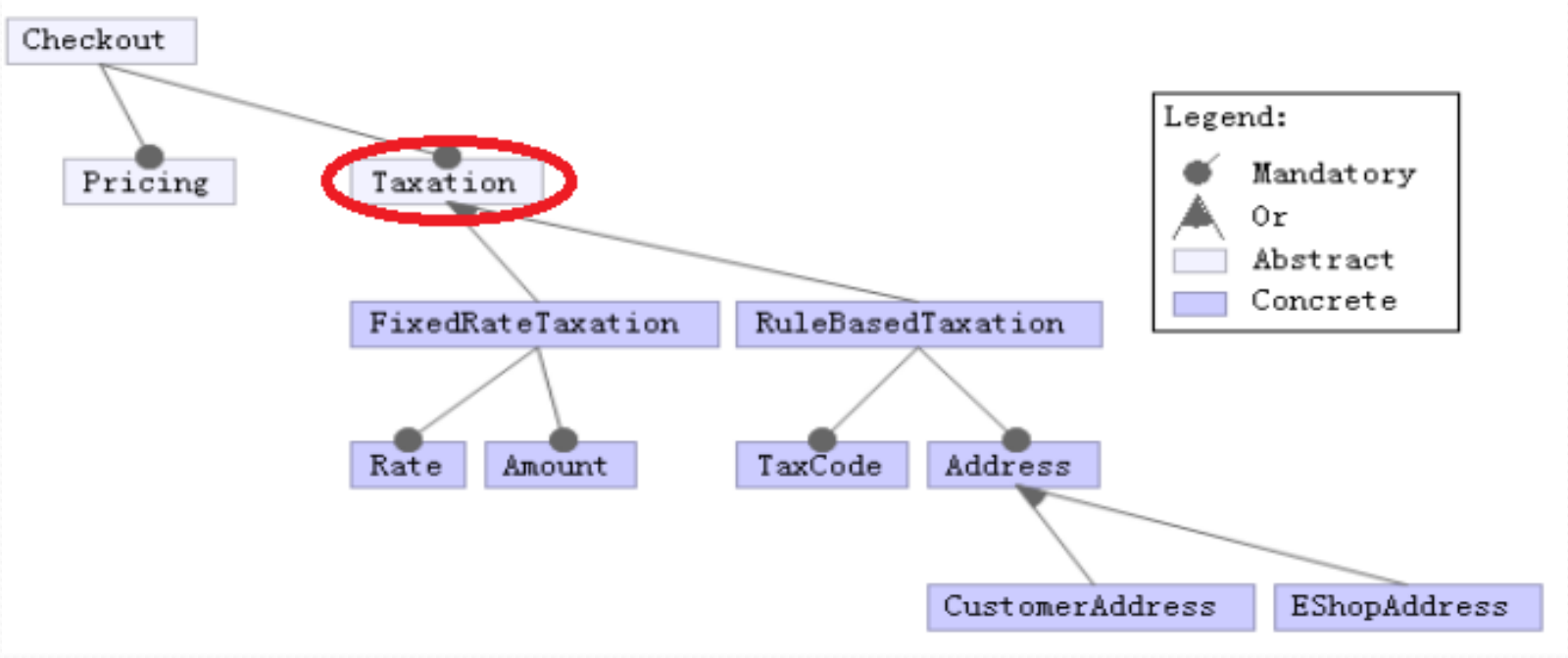


The Feature-Oriented Approach

- Methodology
  - (1) Identify ***features*** from fully refined goal model [Yu08]
    - An inner goal could be identified as an abstract feature
    - Regarding a leaf (operational) goal g_o, a), b) and c) could be a concrete feature
      - a) an OR-decomposed task of g_o;
      - b) a combination (all/partial) of the AND-decomposed tasks of g_o;
      - c) a or a cohesive set of quality constraint(s).
  - (2) Refine identified feature [Kang90]
    - Decomposition: a "*Checkout*" feature can be decomposed into "*Pricing*" and "*Taxation*"
    - Specialization : the "*Taxation*" could be specialized to "*Fixed-Rate Taxation*" and "*Rule-Based Taxation*"
    - Characterization: the "*Fixed-Rate Taxation*" feature has attributes "*Amount*" and "*Tax-rate*"

The Feature-Oriented Approach

- Methodology
  - (3) Deriving operations from feature model [Nguyen 10]



The Feature-Oriented Approach

- Methodology
  - (4) Specifying Service Behavior over Operations[Rinderle06][Broy07]
    - *On deriving the operations, we need to model service behavior, i.e. service interaction protocol (messaging)*
    - *Event - Condition- Action* language
      - T_i *(label)*: ***event*** **[guard]** / ***action*** **[*effect*]**
      - ***Event*** and ***action*** are service operations in general
      - ***Guard*** are conditions, based on which corresponding action would perform
      - ***Effect*** usually leads to an proper state
      - ***E.g.*** T_1 : ***?taxation*** **[true]**/ ***calculateTaxValue***() **[*tax value returned*]**
      - On receiving the taxation request and relevant parameters (***?taxation***), the taxation service calculate and send back the corresponding tax value (***calculateTaxValue()***), then the service would transit into the ***tax value returned*** state.

The Feature-Oriented Approach

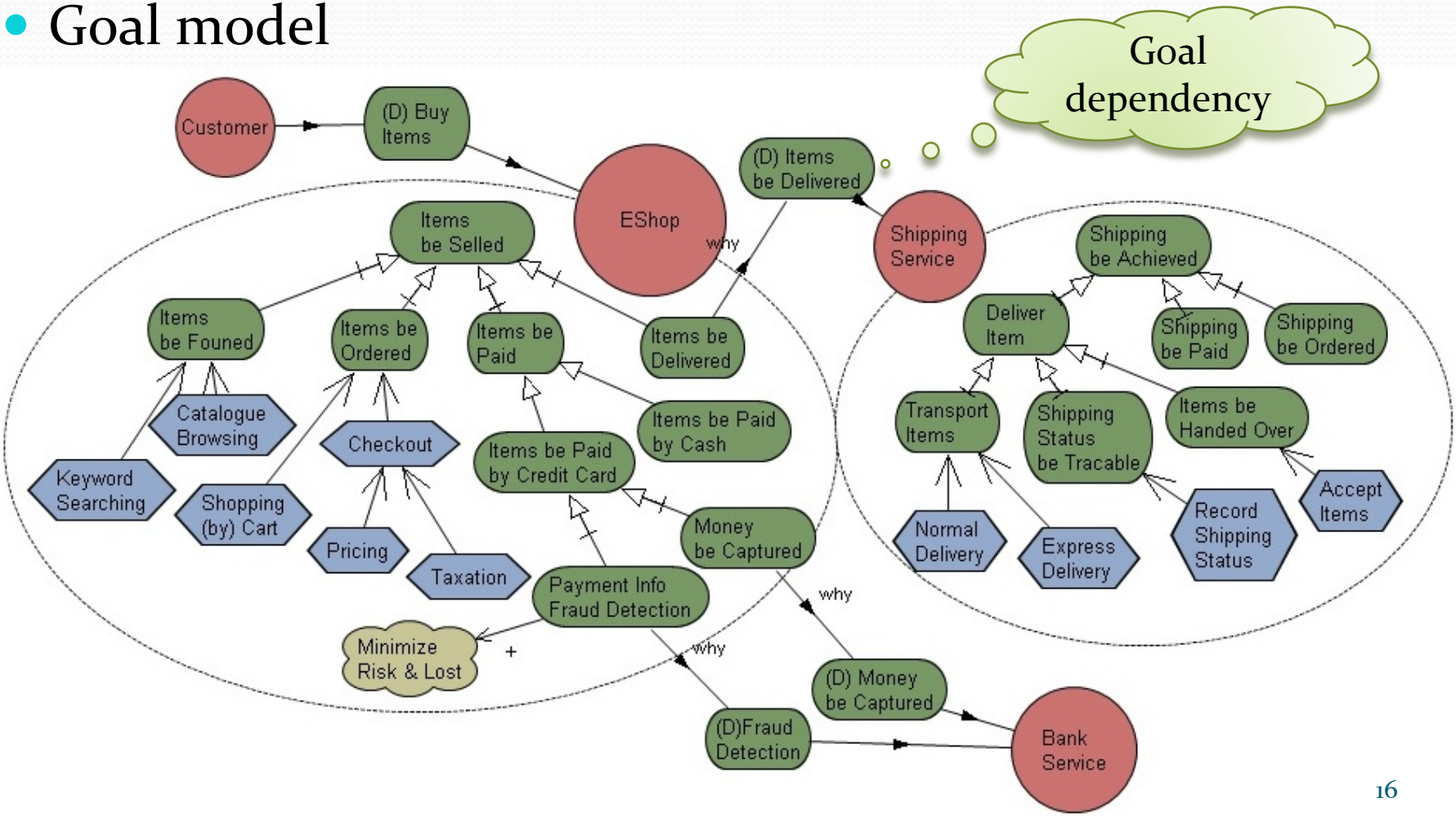
- Brief Sum-Up
  - Having a refined goal model (assumption),
    a) Identify feature from fully refined goal model
    b) Refine identified features
    c) Derive refined feature model to service operation
    d) Specify service behavior over operations
  - Evolution?
    - When requirements change, it will be reflected in goal model
    - Then feature could be changed, added or deleted correspondingly
    - Service operations would evolve synchronously
    - Concurrently, service interaction protocol would evolve

A Case Study - EShop

- EShop
  - It is owned by a store selling different kinds of items, such as book, audio tape and CD.
  - Roles: *customer*, *merchant*, *bank*, and *shipper*. For each role, there would be corresponding software service(s) play it (we omitted the taxation service here).
  - Customers are able to query items and specify their orders; *merchant* could handle orders, use the *bank* service to deal with payment transactions and depend on *shipper* to deliver physical items to customers.

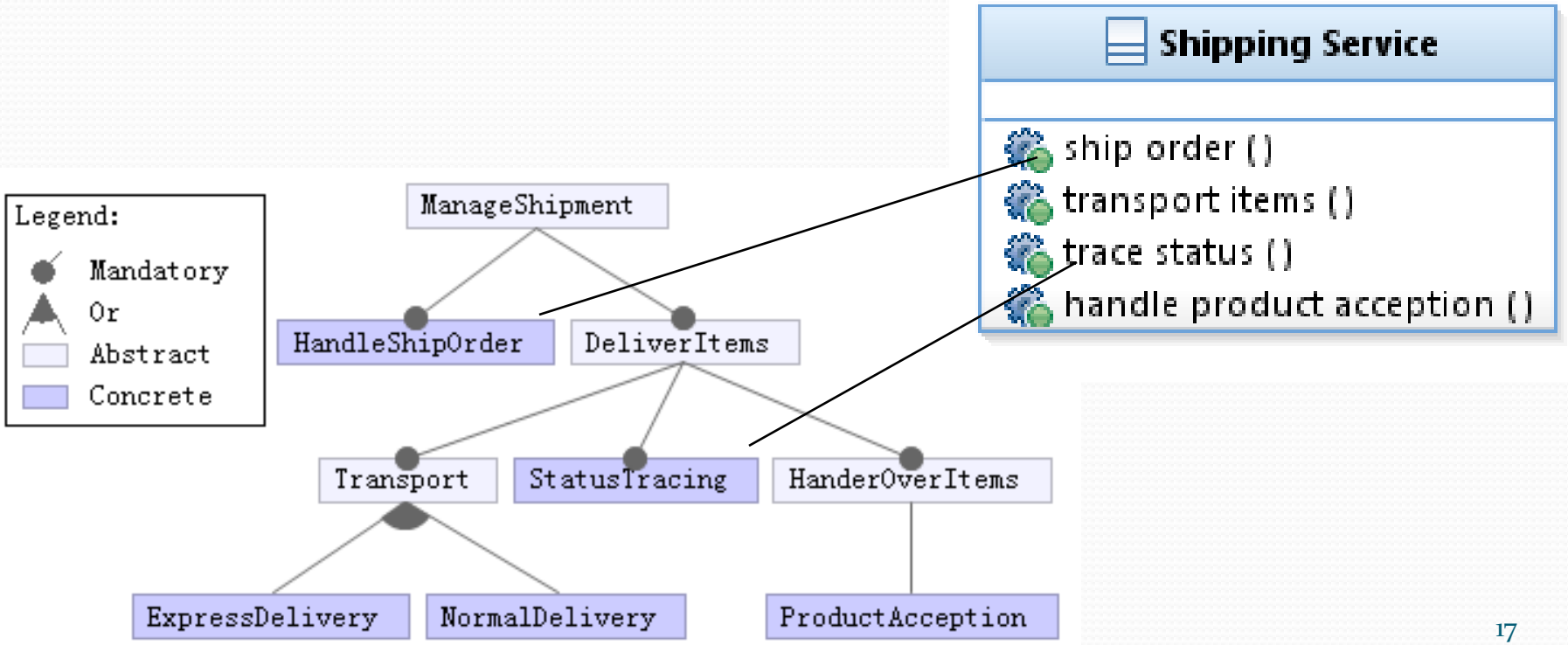
A Case Study - EShop

- Goal model



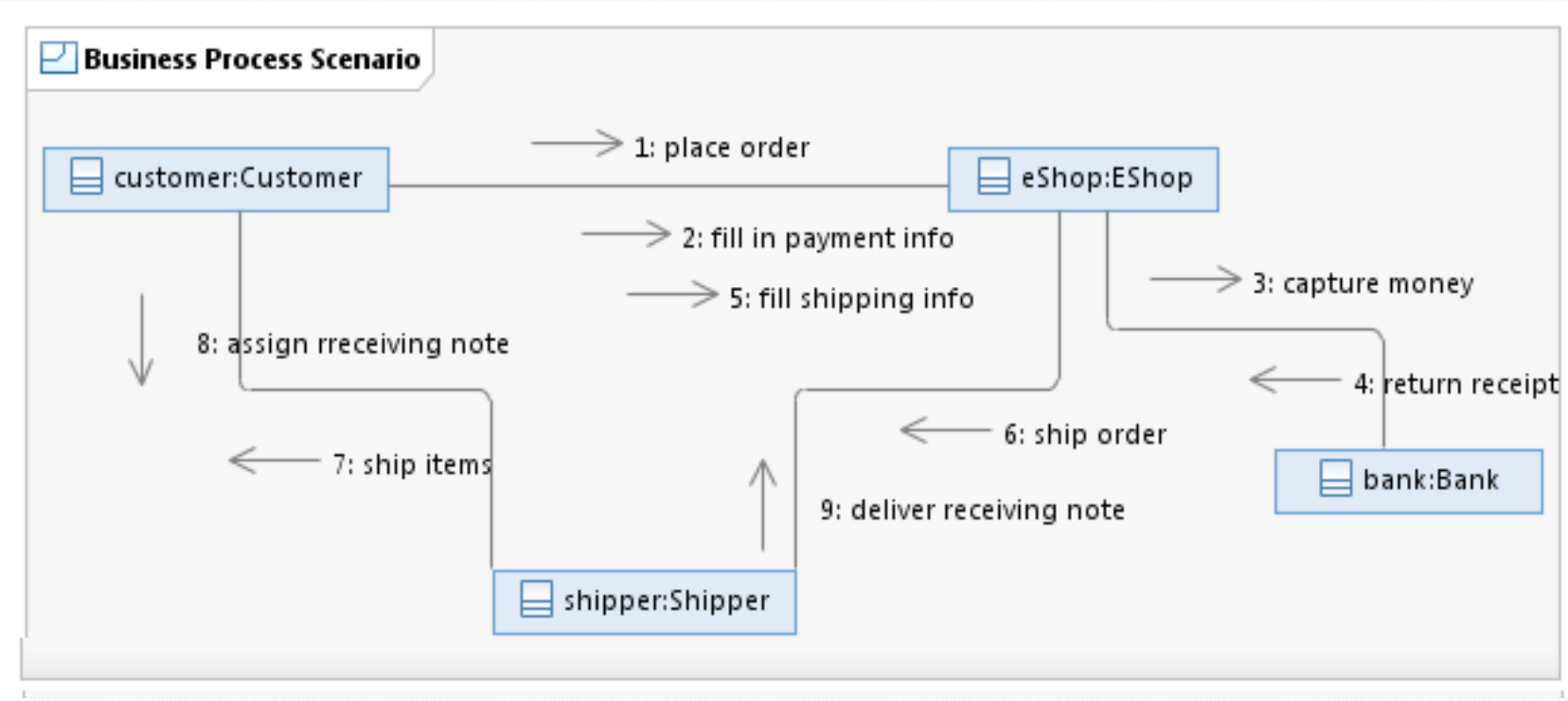
A Case Study - EShop

- A partial feature model and service class
  - Identify features from goal model
  - Derive service operations from refined feature model



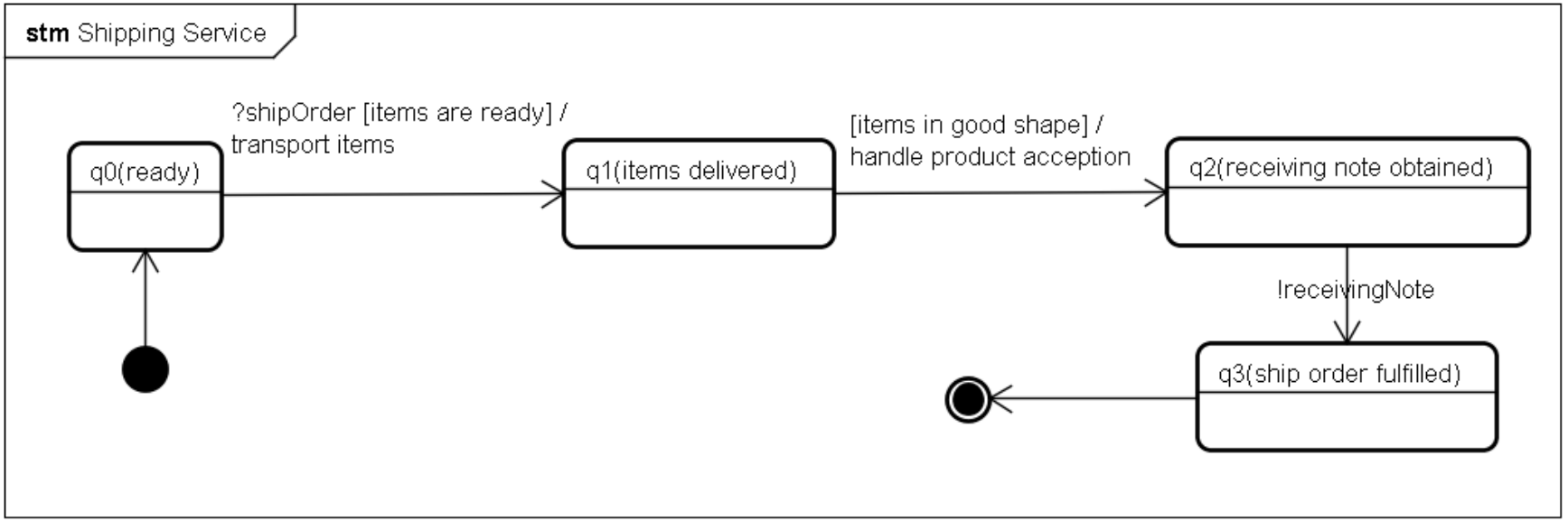
A Case Study - EShop

- A possible process scenario
  - Specify service interaction over messaging



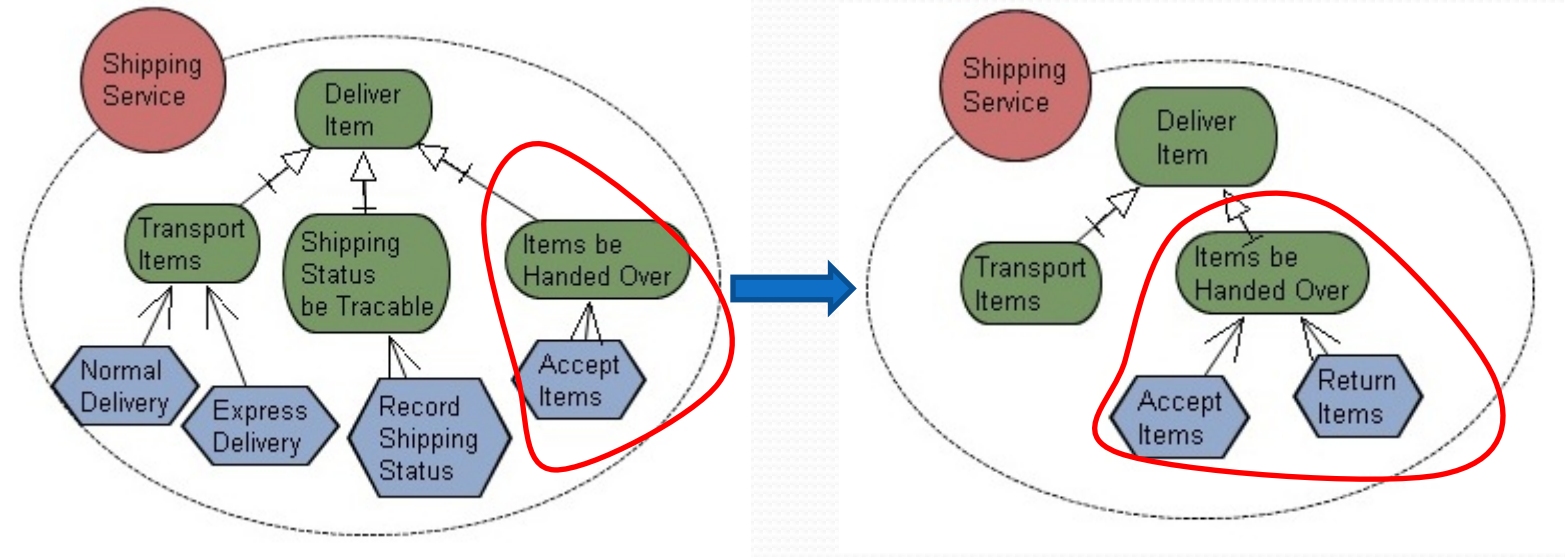
A Case Study - EShop

- The behavior model of the *shipping* service
  - Specify service interaction over service operations



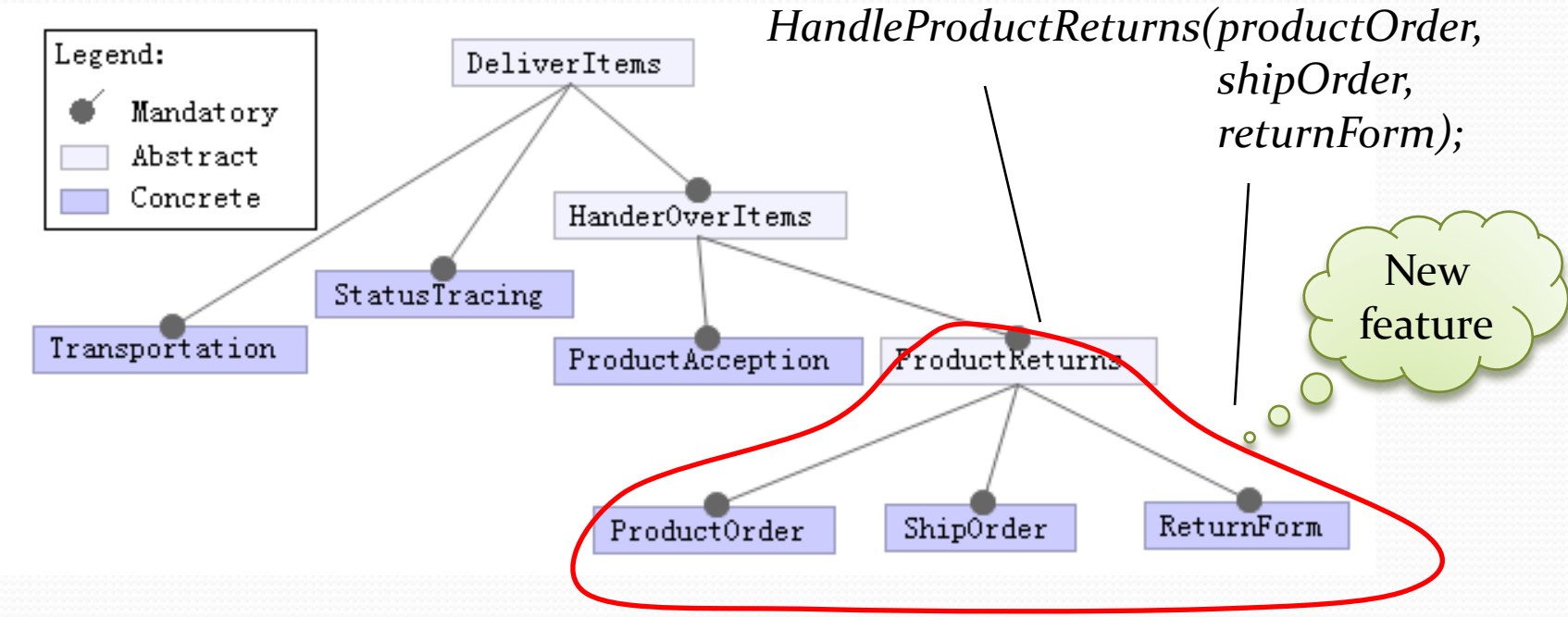
A Case Study - EShop

- An evolution scenario
  - When a customer finds out that the items are broken, he/she may won't accept the items and assign the receiving note.
  - The changed requirement is shown in goal model



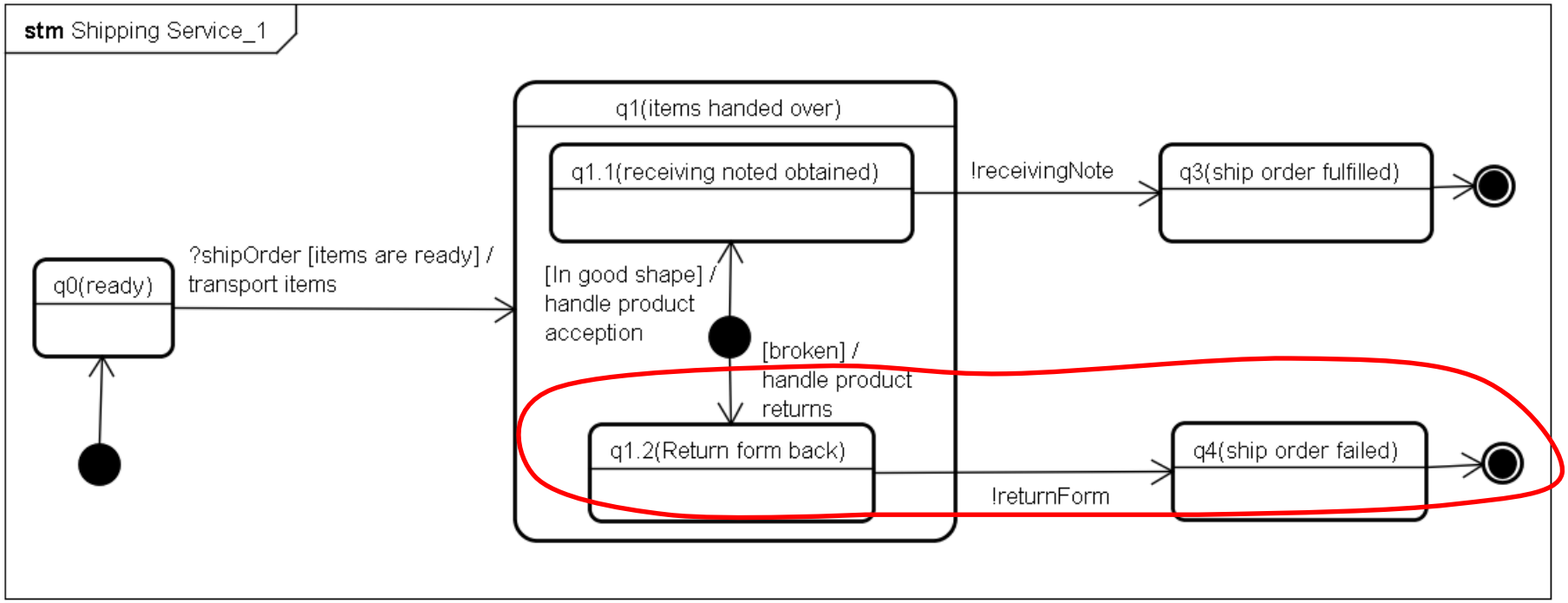
A Case Study - EShop

- The changed feature model
  - A new feature “product return” would be identified
  - Correspondingly, a new service operation would be derived



A Case Study - EShop

- The evolved service behavior model
  - Evolve the behavior model when operations change



In Summary

- Key Challenges
  - How to derive service operations from feature rationally?
  - How to specify service behavior over operations systematically?
- Contributions:
  - Being different from the current work that focus on the interaction compatibility between service and clients in evolution, we center on the change propagation from requirement to service.
    - What is going to evolve
    - How will it evolve

In Summary

- Future Work:
  - How to resolve the influence of service evolution?
  - How to handle the evolution of non-functional feature?
  - How to deal with the change traceability problem?

Reference

- [Li12] Feng-Lin Li, Lin Liu, John Mylopoulos. "Software Service Evolution: A Requirements Perspective" In Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual, July 16-20, 2012, pp 353-358
- [Jackson&Zave95] M. Jackson and P. Zave, "Deriving specifications from requirements: an example," in Software Engineering, 1995. ICSE 1995. 17th International Conference on, 1995, p. 15–15.
- [Yu08] Y. Yu, J. C. . do Prado Leite, A. Lapouchnian, and J. Mylopoulos, "Configuring features with stakeholder goals," in *Proceedings of the 2008 ACM symposium on Applied computing*, 2008, pp. 645–649.
- [Kang90]K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," *DTIC Document*, 1990.
- [Nguyen 10] T. Nguyen and A. Colman, "A Feature-Oriented Approach for Web Service Customization," in *Web Services (ICWS), 2010 IEEE International Conference on*, 2010, pp. 393–400.
- [Broy07] M. Broy, I. H. Krüger, and M. Meisinger, "A formal model of services", ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 16, no. 1, p. 5-es, 2007.
- [Rinderle06] S. Rinderle, A. Wombacher, and M. Reichert, "Evolution of process choreographies in DYCHOR", On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, pp. 273-290, 2006.

Q & A

Welcome Questions!