Re-engineering a Credit Card Authorization System for Maintainability and Reusability of Components- A Case Study

Moonzoo Kim, et al

Appeared at *International Conference on Software Reuse* (ICSR) '06
Torino, Italy

Summary of the LG CAS Project

- We have re-engineered LG Credit card authorization system (CAS) for enhancing maintainability and reusability according to sound design principles
  - Reviewed CAS as well as its revision history and market requirement changes
  - Designed/extracted a feature model
    - domain analysis to figure out commonality and variability
  - Redesigned architecture and components based on the feature model with sound design principles

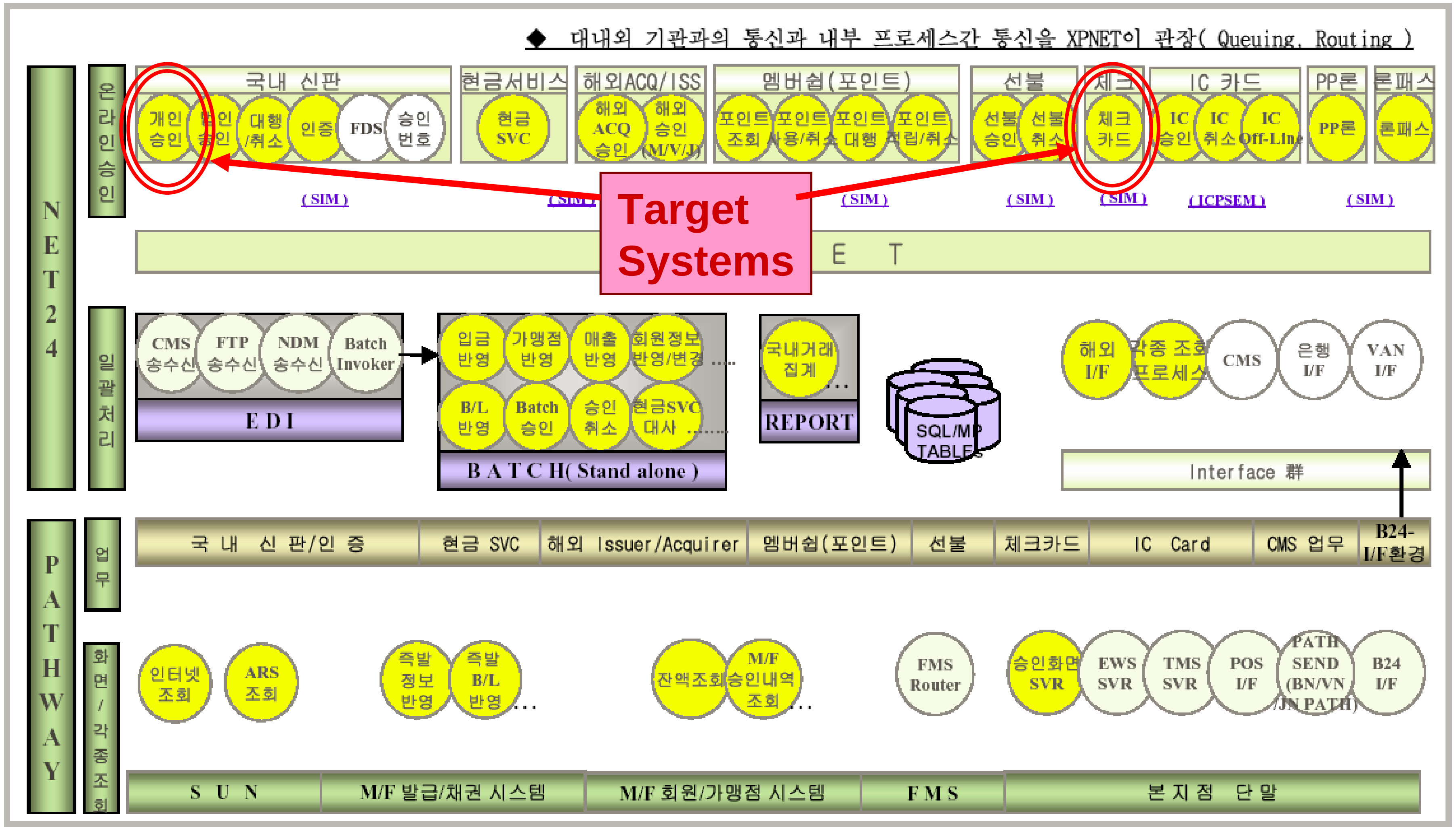
Outline

- Part I: Background on LG CAS
  - – Motivation
  - – Overview of LG Card Authorization System (CAS)
- Part II: Re-engineering LG CAS
  - – Re-engineering Principles
  - – Re-engineering CAS
- Part III: Lessons Learned
  - – Three Lessons Learned
  - – Conclusion

Part I :Background on LG CAS

- What LG CAS is
- Motivations
- Overview of CAS

Background



주) ● : 구축 범위, PP론 : Point Phone 론, SIM : Standard Internal Message

Background (cont.)

- LG-card co. ltd adopted a component based development method in 2004
  - They started to re-develop CAS by
    - converting hard-coded business rules into a database
    - Standardizing component interfaces
    - Applying component based management programs
      - Reuse rate measurement
      - Component library construction
      - Component reengineering
- Nevertheless, LG had difficulties in maintaining CAS
  - The developers added/updated components in an ad-hoc way at each update request ☹

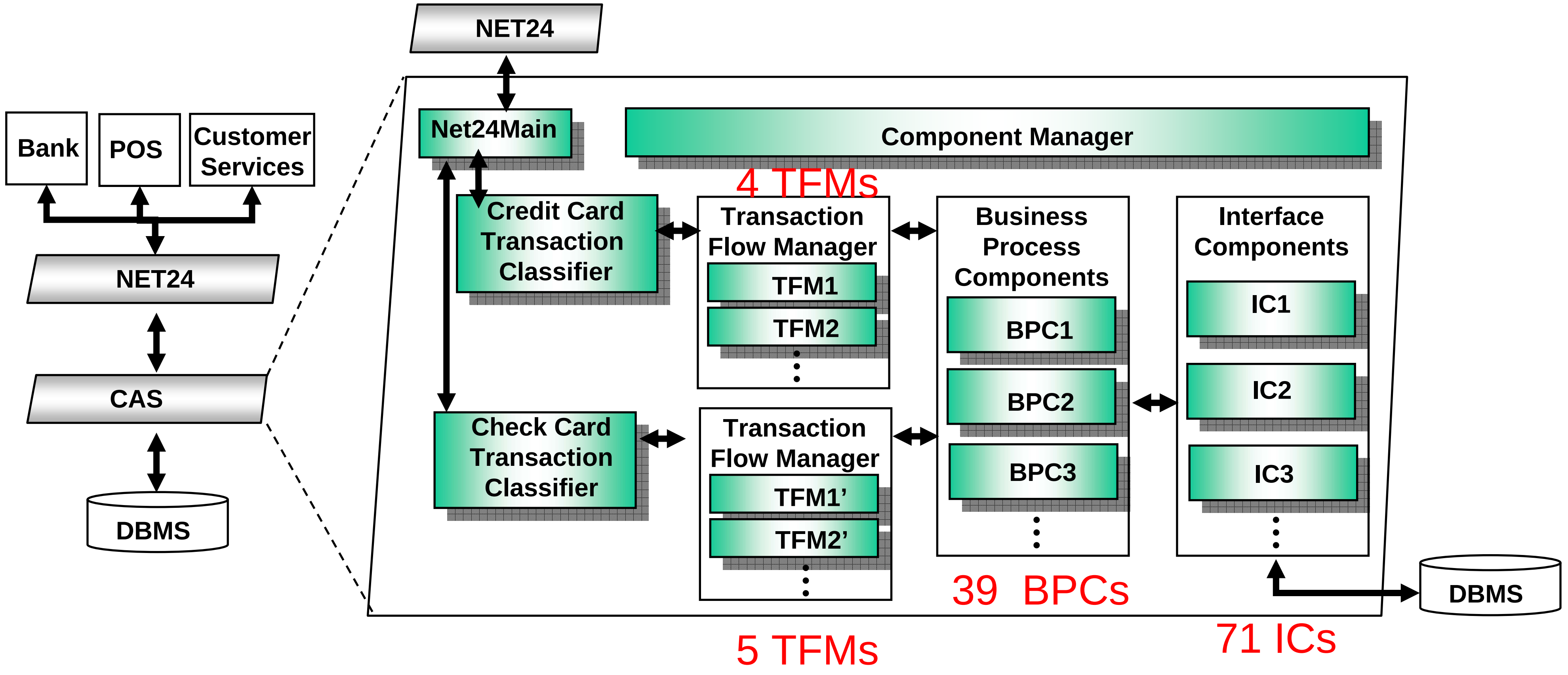
Motivation

- Frequent updates make maintainability as a crucial issue
  - Due to government law change, competition between card companies, etc
  - Complexity of system maintenance is increased
- CAS should be designed to
  - accommodate changing requirement easily
  - isolate effects of updates as much as possible

Motivation (cont.)

- But, new services have been added to CAS by simply adding new components
  - Specially developed for those services without consideration of common/reusable characteristics of the services
  - Lack of proactive design that anticipates updates of services based on market evolution
- Ad-hoc way of evolution resulted in redundant code and difficulty of understanding program behavior.
  - Newly added services or updates easily affected unnecessarily large segments of CAS and caused high maintenance cost

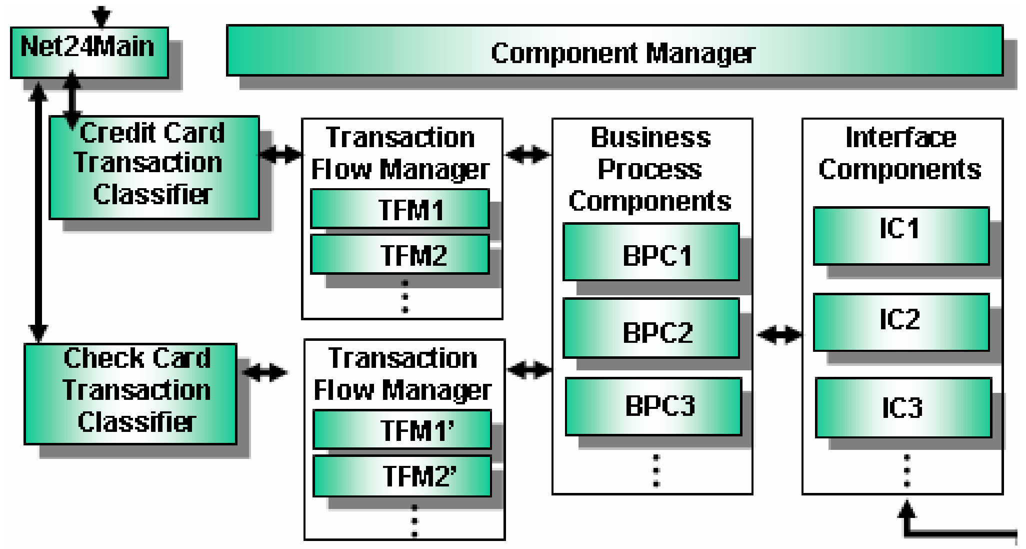
Overview of LG Card Authorization System



Overview of LG Card Authorization System

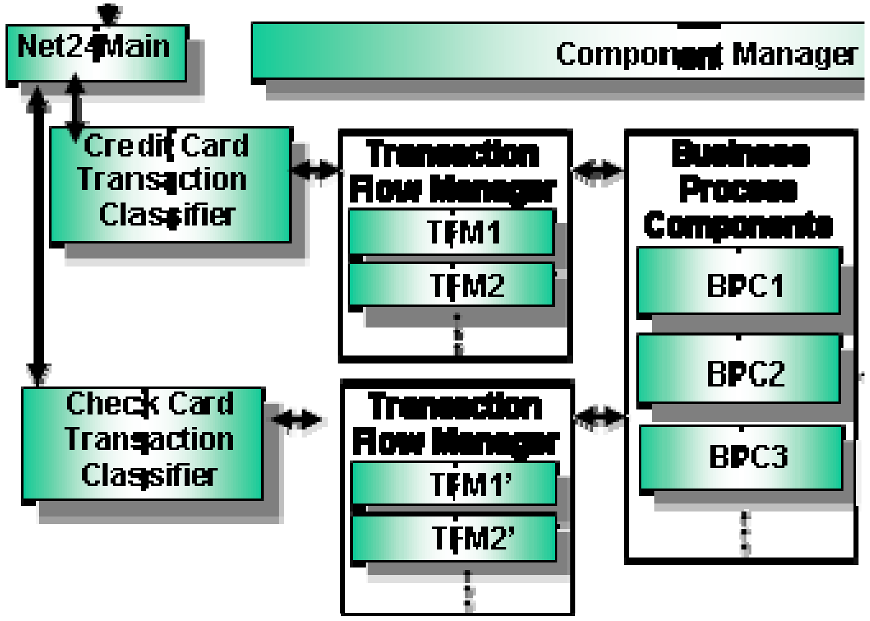
- **TC** classify transaction types and calls appropriate TFM
- **TFM** manage transaction flows by controlling business processes implemented in the **BPC** (explained later in detail)
- **IC** works as data holders communicating with the database system.
- **Component manager** handles orderly creation of these components preventing redundant instantiation.

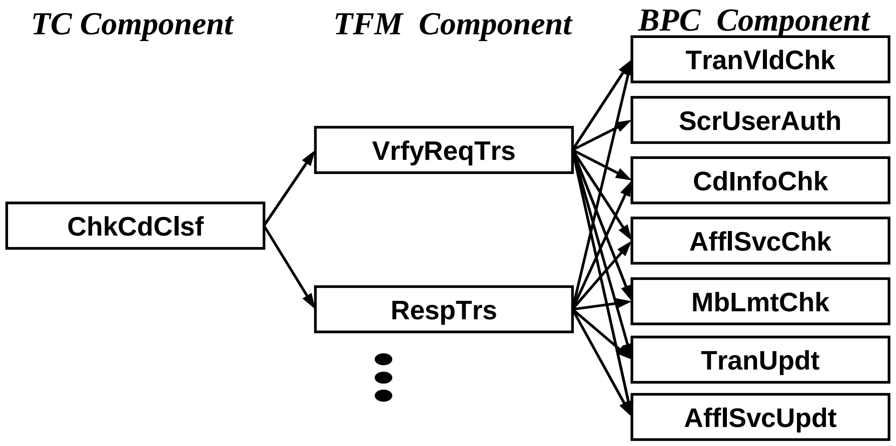
- Higher layer calls lower layer via call/return methods
  - TC calls a TFM, then the TFM calls BPCs, etc



An Example of Execution Flows

- When a user purchases a product using his/her check card, a purchase authorization request is sent from the sotre to CAS
  - CHkCDClsf classifies check card transaction
    - VrfyReqTrs checks if the requested transaction is valid or not
      - TransVldChk identifies the place where the transaction occurs
      - ScrUserAuth checks user's identification/passwd if purchase is from online store
      - …
    - RespTrs handles a transaction from the credit card issuer (bank)



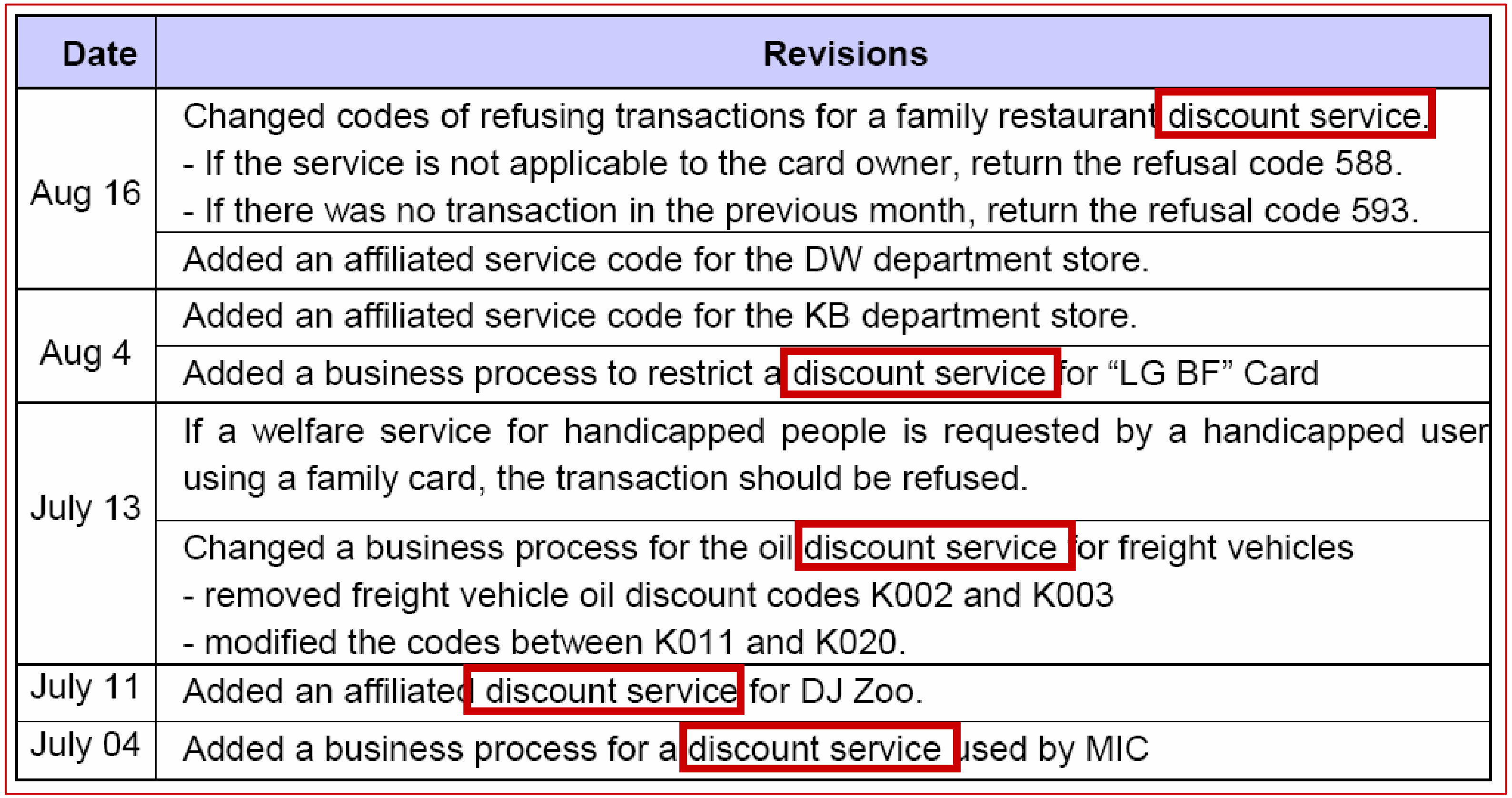


Re-engineering LG CAS: Part II

- *starting with domain analysis*
- *principles matter*
- *result of re-engineering*

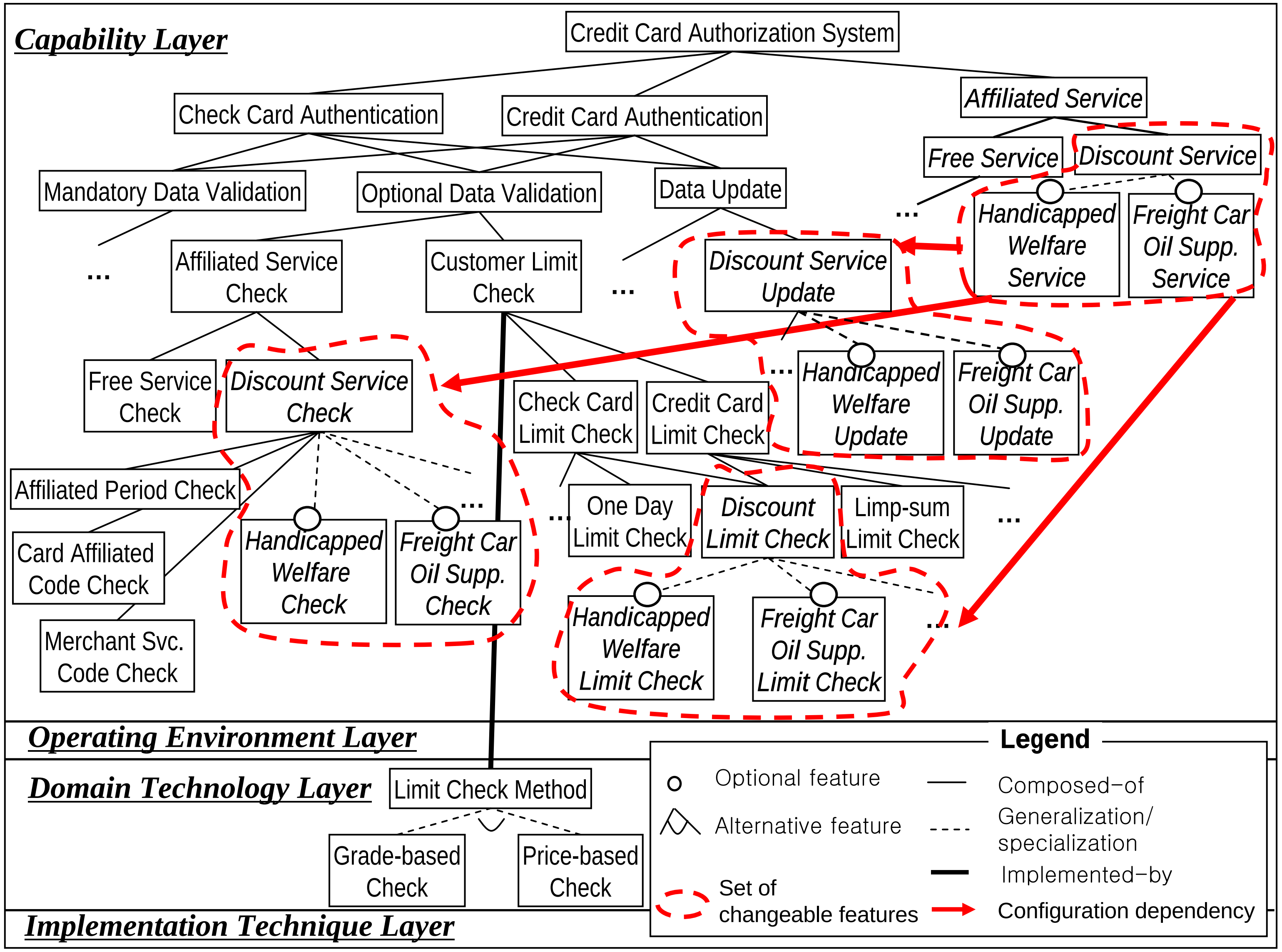
An Example of Revision History

- From the revision history, we can find frequently updated features
  – we can analyze variabilities of the CAS domain



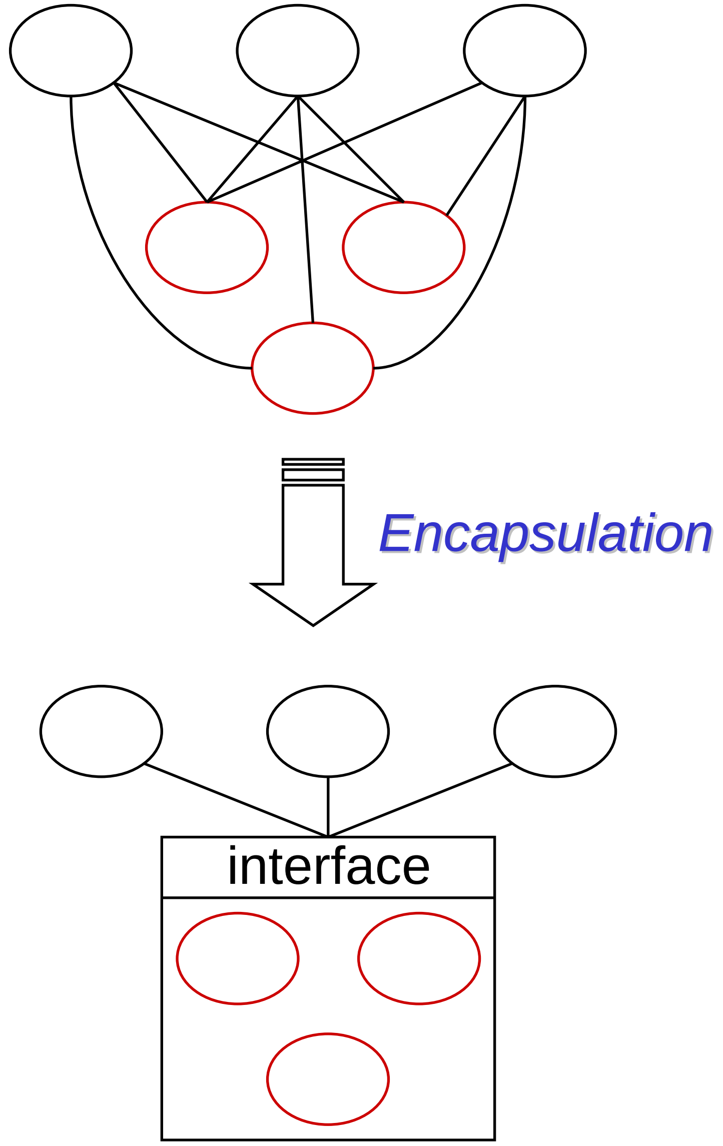
Date	Revisions
Aug 16	Changed codes of refusing transactions for a family restaurant discount service. - If the service is not applicable to the card owner, return the refusal code 588. - If there was no transaction in the previous month, return the refusal code 593.
	Added an affiliated service code for the DW department store.
Aug 4	Added an affiliated service code for the KB department store.
	Added a business process to restrict a discount service for “LG BF” Card
July 13	If a welfare service for handicapped people is requested by a handicapped user using a family card, the transaction should be refused.
	Changed a business process for the oil discount service for freight vehicles - removed freight vehicle oil discount codes K002 and K003 - modified the codes between K011 and K020.
July 11	Added an affiliated discount service for DJ Zoo.
July 04	Added a business process for a discount service used by MIC

A Feature Model of CAS



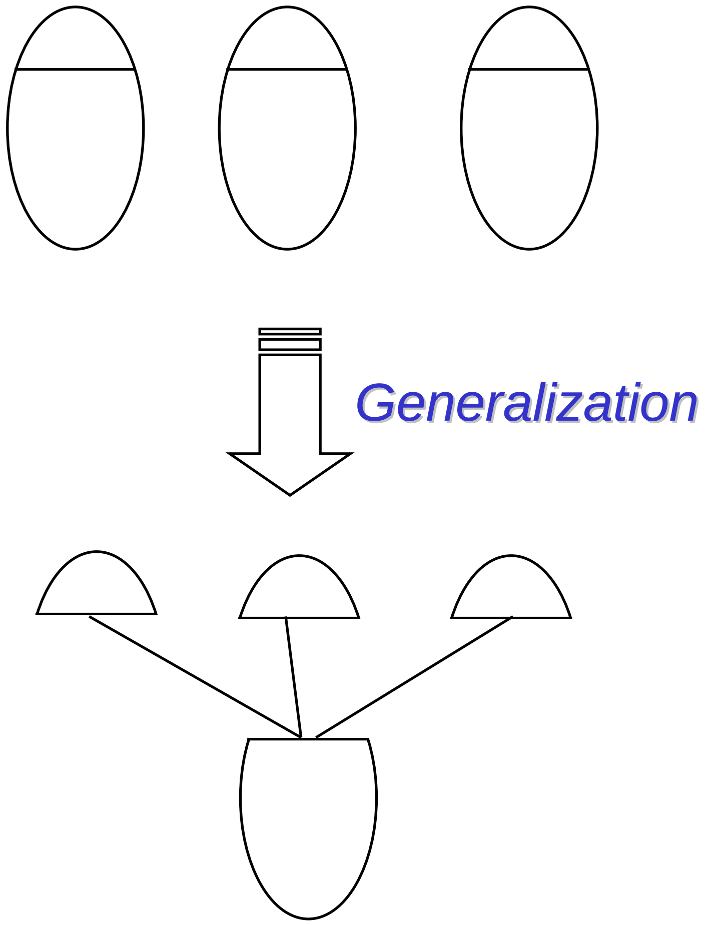
Three Re-engineering Principles I/III

- Encapsulation of Evolving Features
  - A complex system like CAS suffers from high degree of coupling among components
    - Difficult to understand
    - Hard to revise and maintain
    - Degrading evolvability



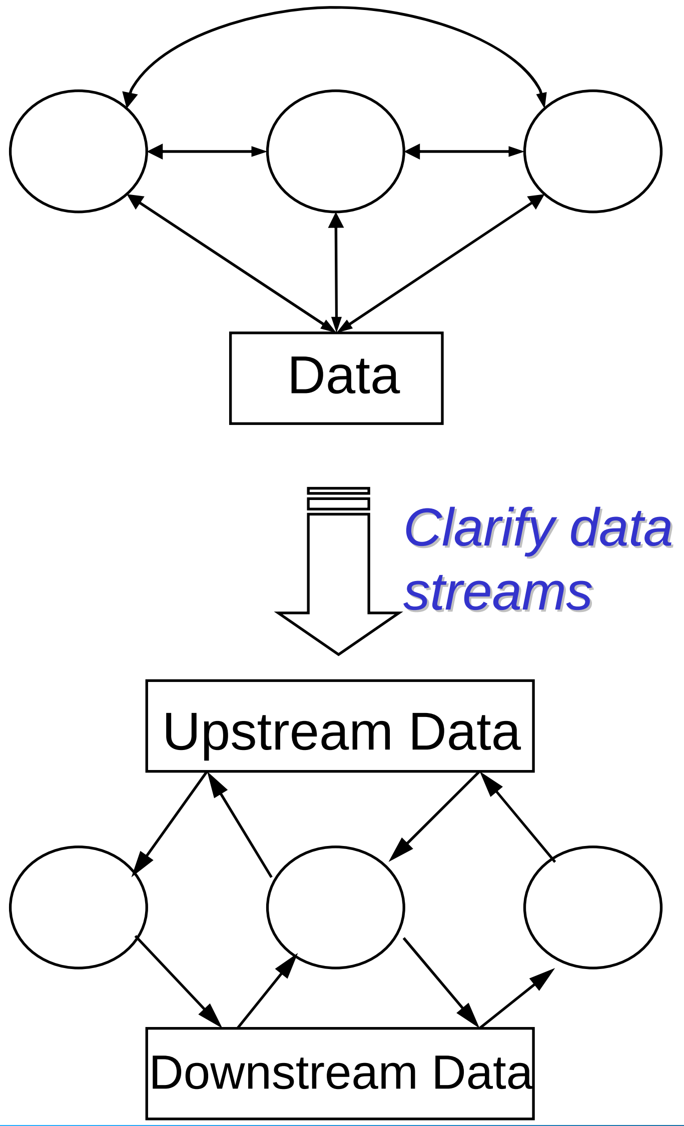
Three Re-engineering Principles II/III

- Generalization of Common Processes
  - In a large system, multiple components with slightly different services easily prevail
    - All redundant components should be modified altogether
    - It is hard to find which components is responsible for a specific behavior of the system



Three Re-engineering Principles III/III

- Separation of data streams
  - As typical of information processing systems, the main operations of CAS are to retrieve, process, and update data
  - Data streams/flows should be clearly visible to figure out system's behavior



Encapsulation of Business Processes into Modules

Business Process Table

1.Transaction Validity Check
2.Screen User's Authentication
3.Card Information Check
4.Member's Information Check
5.Franchise Associate's Validity Check
6.Member's State Check
7.Card Validity Check
8.Affiliated Service Check
9.Member's Limited Amount Check
10.Fraud Transaction Check
11.SMS Transmission Check
20.Credit-card Transaction Update
21.Check-card Transaction Update
22.Card Information Update
23.Domestic CAVV Transaction Update
24.Affiliated Service Update
25.Limits on Daily Card Usage Update
26.Member's Limit Update
…

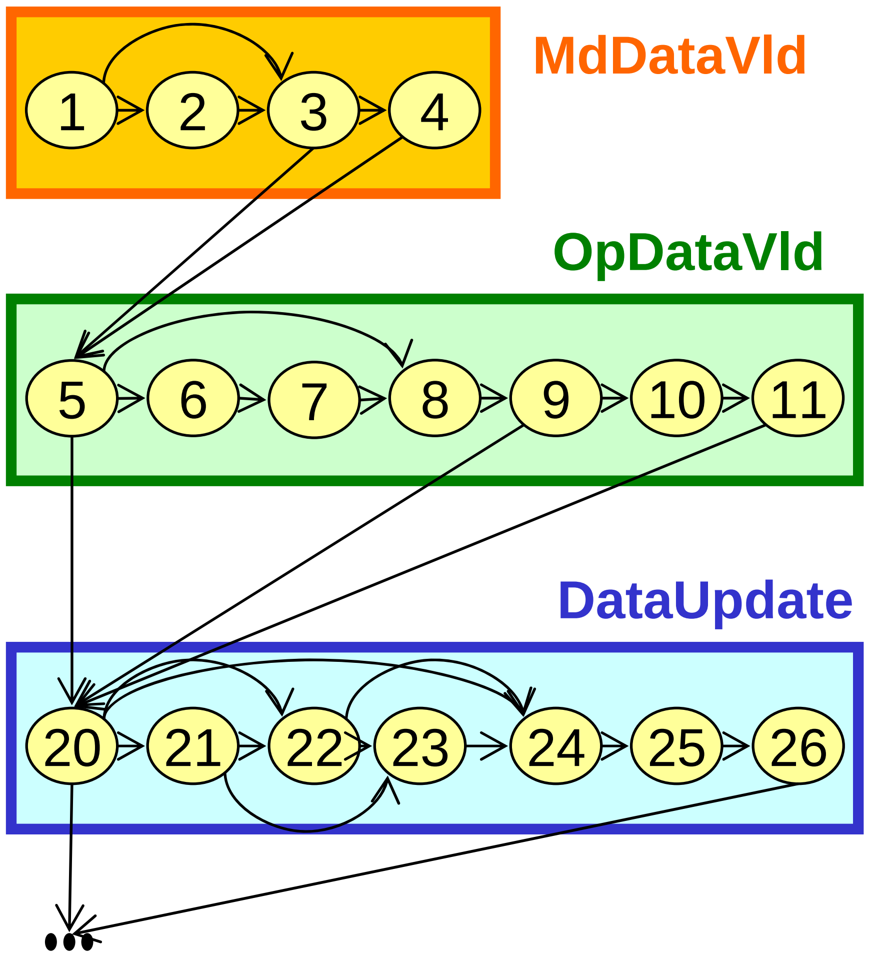
Mandatory Data Validation (1–4)

Optional Data Validation (5–11)

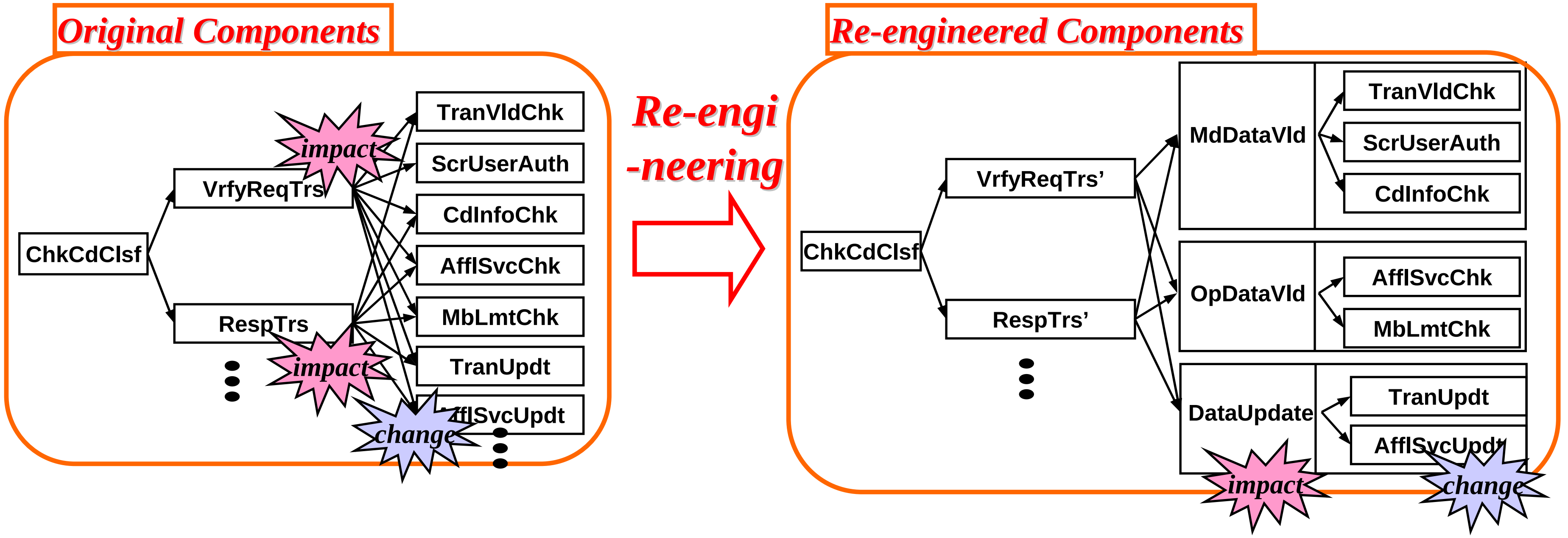
Valid-ation (1–11)

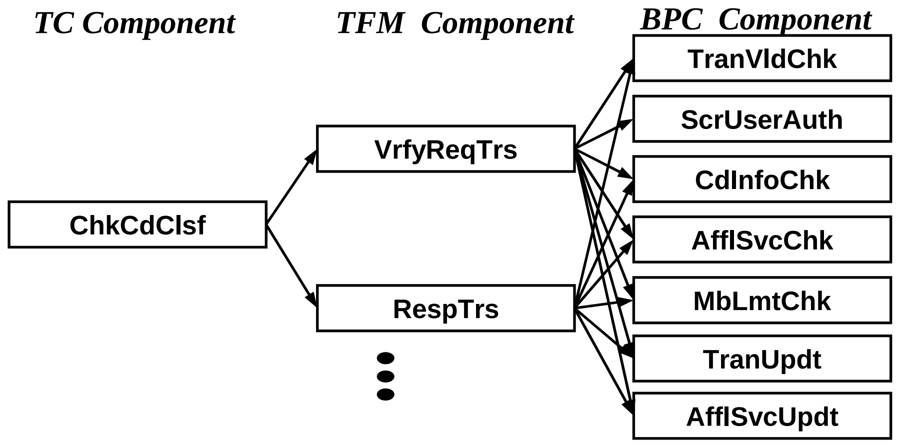
Update (20–26)

Business Process Workflow

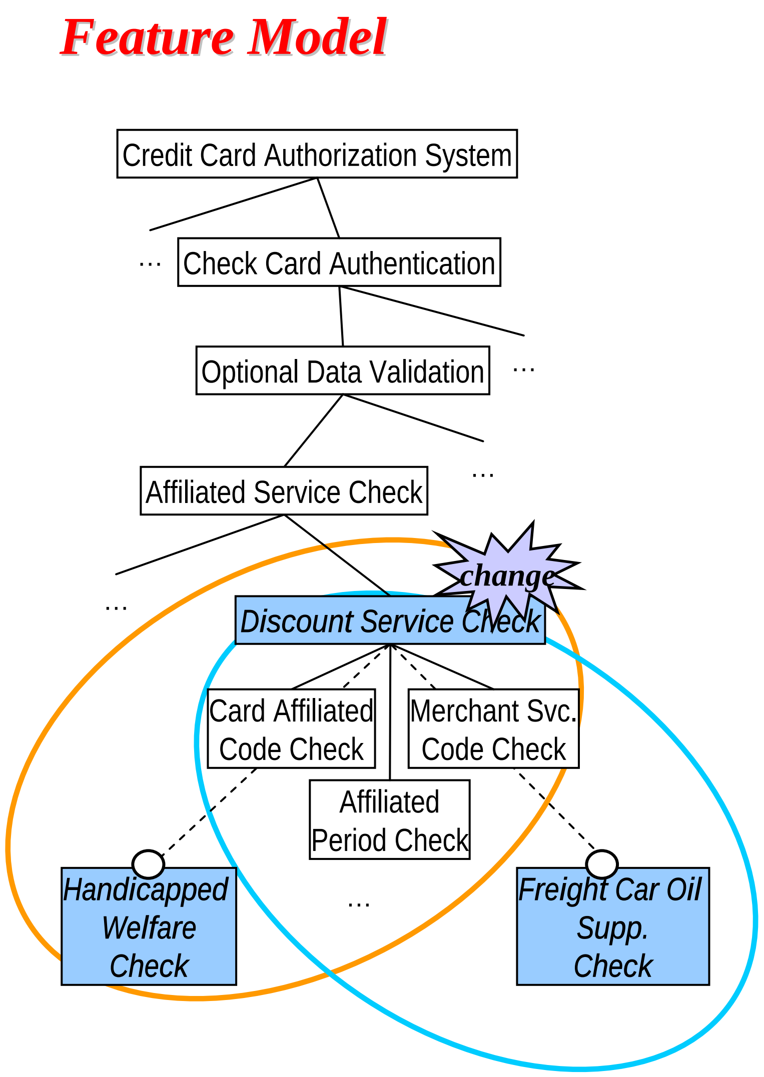


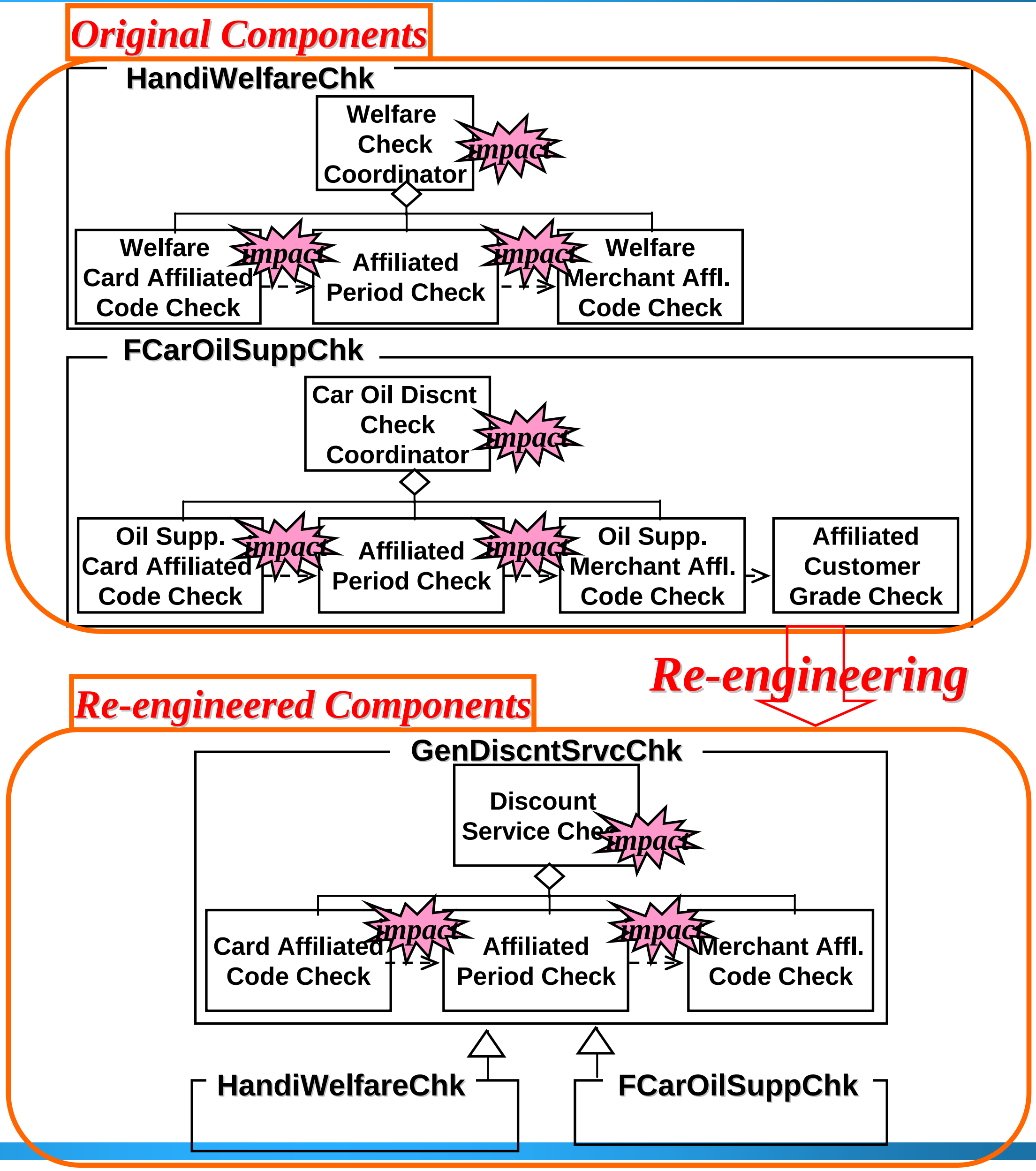
Re-engineering Principle1: Modularization Based on Data Usage





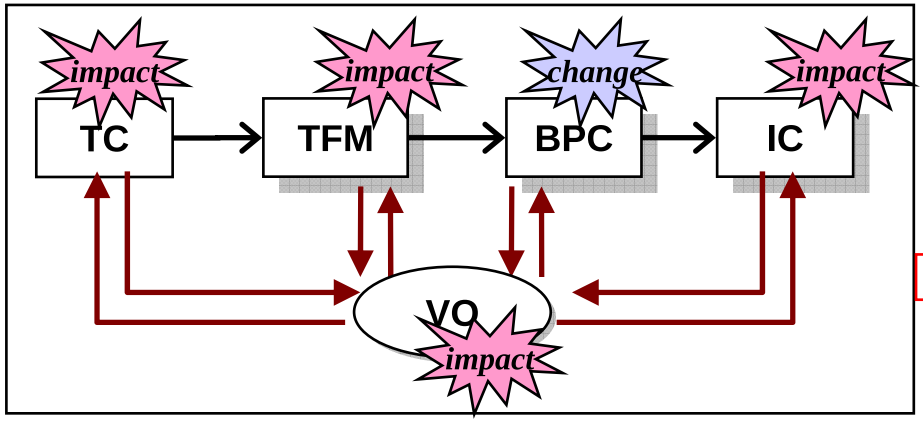
Re-engineering Principle 2: Commonality

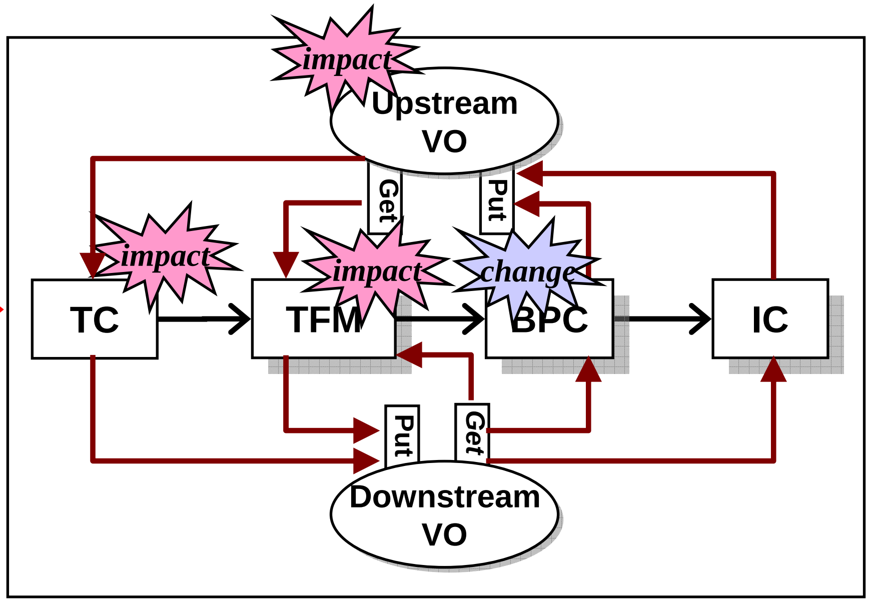


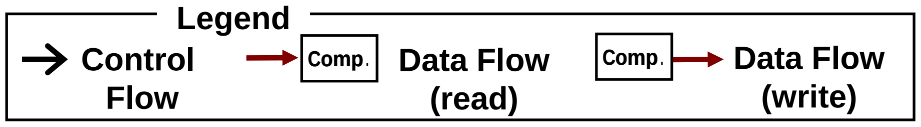


Original Components
HandiWelfareChk
Welfare Check Coordinator
impact
Welfare Card Affiliated Code Check
impact
Affiliated Period Check
impact
Welfare Merchant Affl. Code Check
FCarOilSuppChk
Car Oil Discnt Check Coordinator
impact
Oil Supp. Card Affiliated Code Check
impact
Affiliated Period Check
impact
Oil Supp. Merchant Affl. Code Check
Affiliated Customer Grade Check
Re-engineering
Re-engineered Components
GenDiscntSrvcChk
Discount Service Check
impact
Card Affiliated Code Check
impact
Affiliated Period Check
impact
Merchant Affl. Code Check
HandiWelfareChk
FCarOilSuppChk

Re-engineering Principle3: Separated Data Streams







Part III: Lessons Learned

- necessity of proactive re-engineering
- management of commonality and variability
- broad coverage of a feature model for system analysis

Necessity of Proactive Re-engineering

- Proactive reengineering is not optional, but essential
- Lack of analysis on commonality and variability of services required when a system is designed
- Thus, developers tend to revise the system in an ad-hoc manner without considering how the system should be designed for better maintainability
  - High degree of redundancy and component coupling
  - Poor maintainability => high maintenance cost

Management of Commonality and Variability

- Dependency relationships between features of the feature model is useful to recognize the effects of service changes/additions
- Generalization/specialization relationships also helped use to encapsulate similar components into generalized ones and to adopt new services more conveniently

Broad Coverage of a Feature Model for System Analysis

- The feature model successfully provided guidelines for analyzing the CAS system in abroad way
  - From architectural issues to component refactoring
    - Features of higher level are related to system assets of a large scale
    - Features at leaf nodes are mostly related to small objects
- A carefully built feature model can be used for analyzing a system in various levels of abstraction

Conclusion

- Design principles are truly useful in practical situations
  - Spend time to understand and memorize
  - Stick to the principles
- Proactive design must be prepared before developing any meaningfully large system
  - Otherwise, you will pay back later with larger cost
  - Consider domain analysis as an essential part of your system design