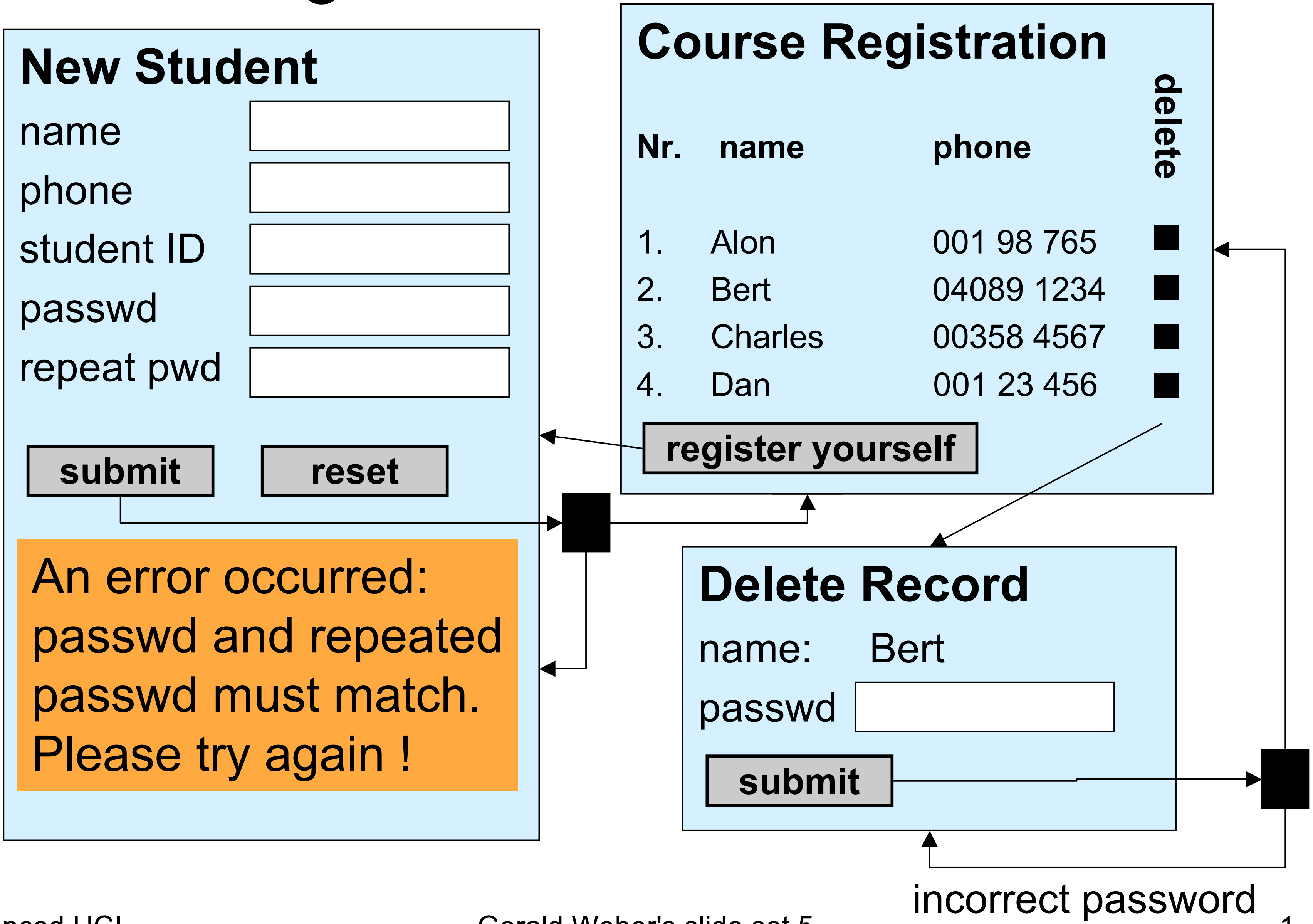
Screen Diagram



New Student

name

phone

student ID

passwd

repeat pwd

An error occurred: passwd and repeated passwd must match. Please try again !

Course Registration

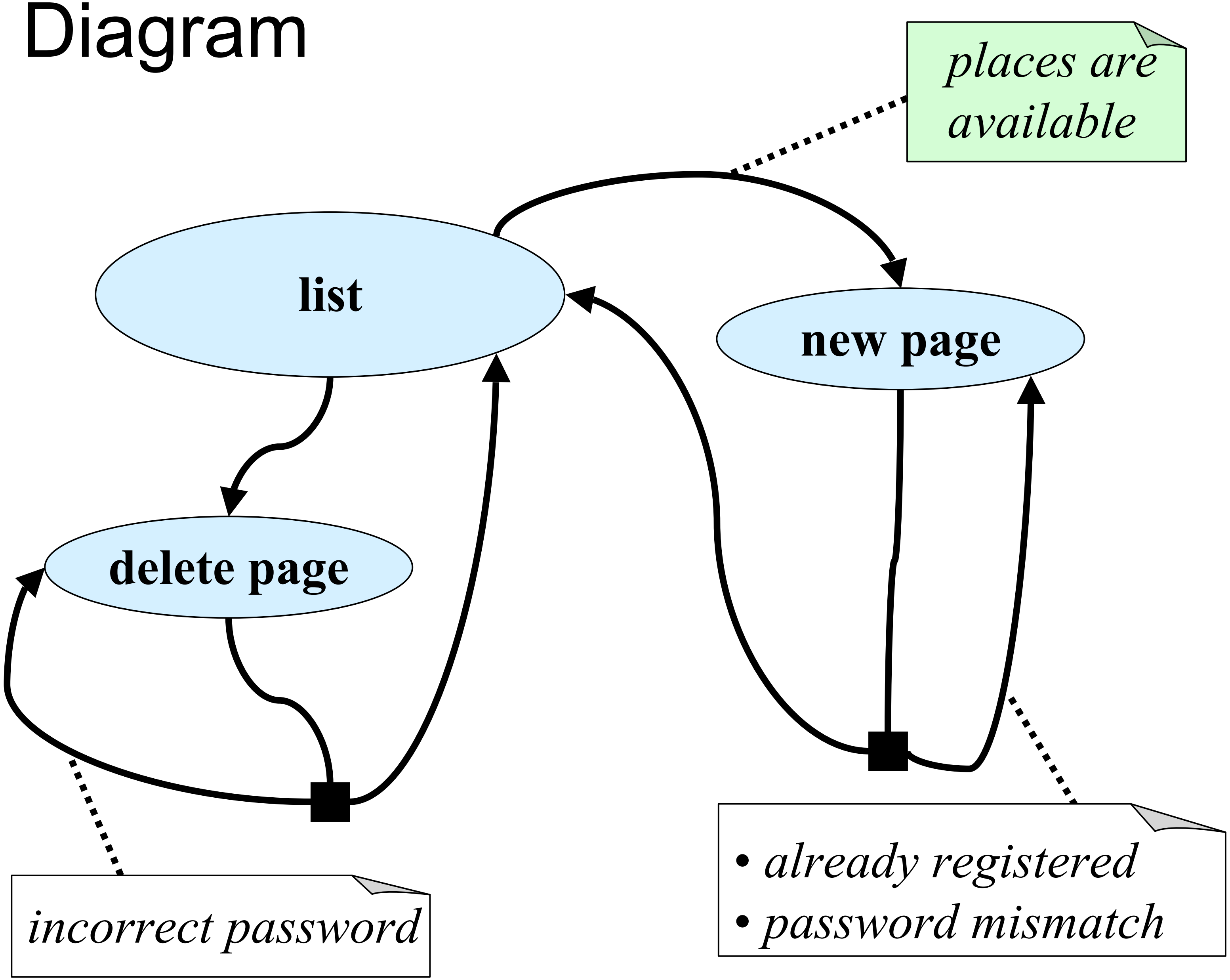
Nr.	name	phone	delete
1.	Alon	001 98 765	■
2.	Bert	04089 1234	■
3.	Charles	00358 4567	■
4.	Dan	001 23 456	■

Delete Record

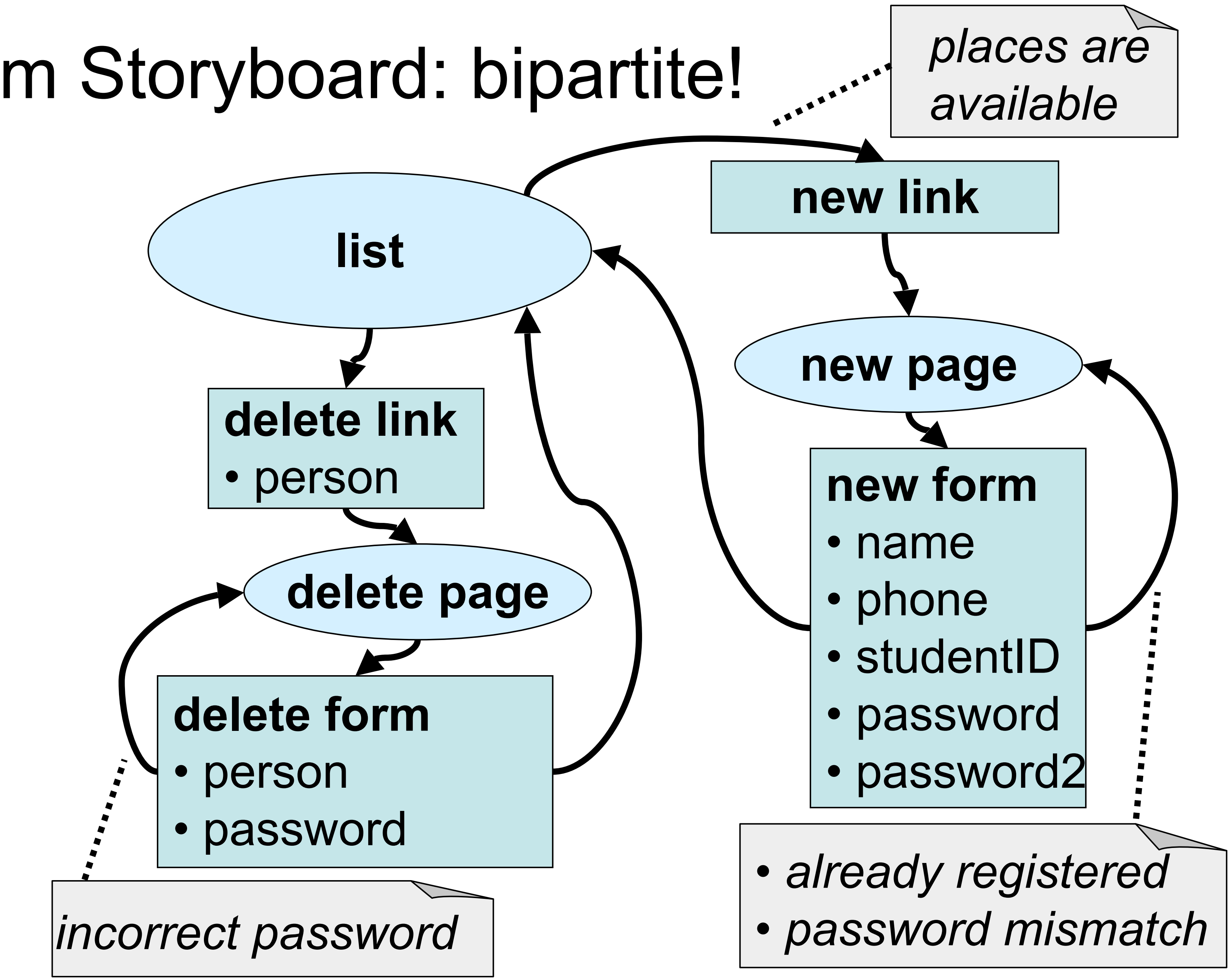
name: Bert

passwd

Page Diagram



Form Storyboard: bipartite!



Submit/Response Style Interfaces

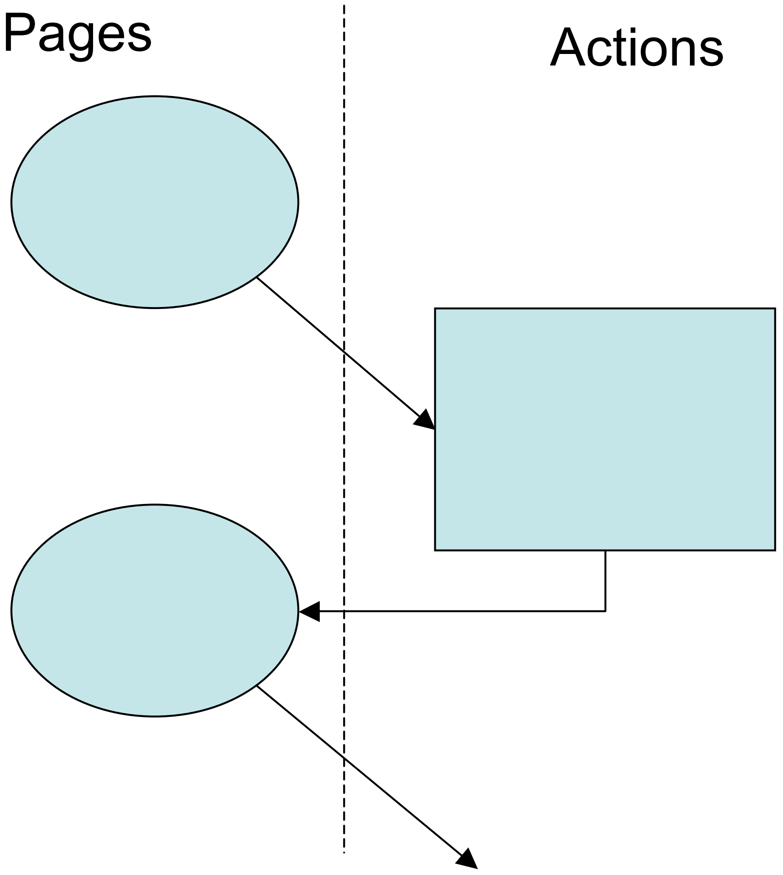
- The form-oriented interface style is technology-independent.
- we define a precise class of form-based interfaces: *submit/response style* interfaces (Form Oriented Analysis [Draheim, Weber])
- Two kinds of interactions
  - ephemeral interaction within a page: *page interaction*
  - triggering a page change: *submit, page change*
- Submit/response is form-based, but not vice versa
  - Web interfaces are submit/response style.
  - Other form-based interfaces are possible
    - form as a constantly updated view on data: desktop databases
  - tradeoff: less features, but simple definition

Two-Level Interaction

- Page interaction:
  - only referring to next page change
  - Can be usually reset (deleted, undone)
- page change (submit)
  - may irrevocably update the system state.
  - Delivers new system-generated page.
- Submit/response style interface
  - form-based interface with screen transactions
  - technology independency: web, mainframe/terminal, 4GL/client/server

Message-based interface model

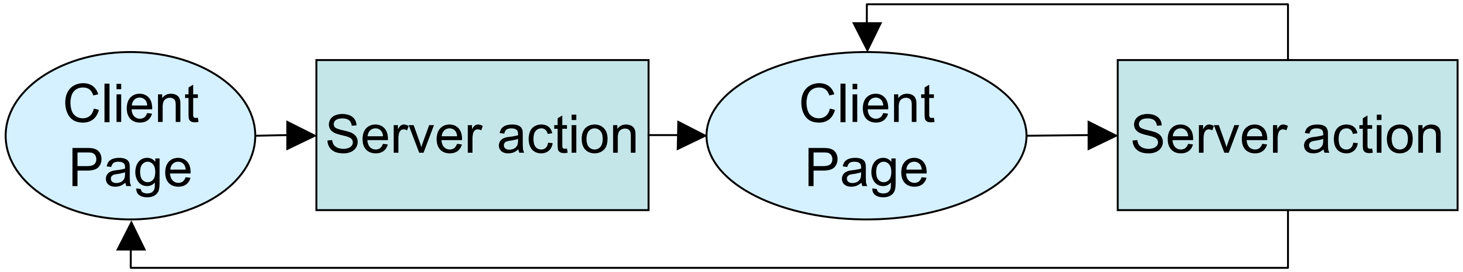
- Natural interpretations of submit/response style user interfaces
- System behavior: sequence of interchanging client pages and server actions.
- Thin client sends a message on submit
- Server responds automatically with a response page



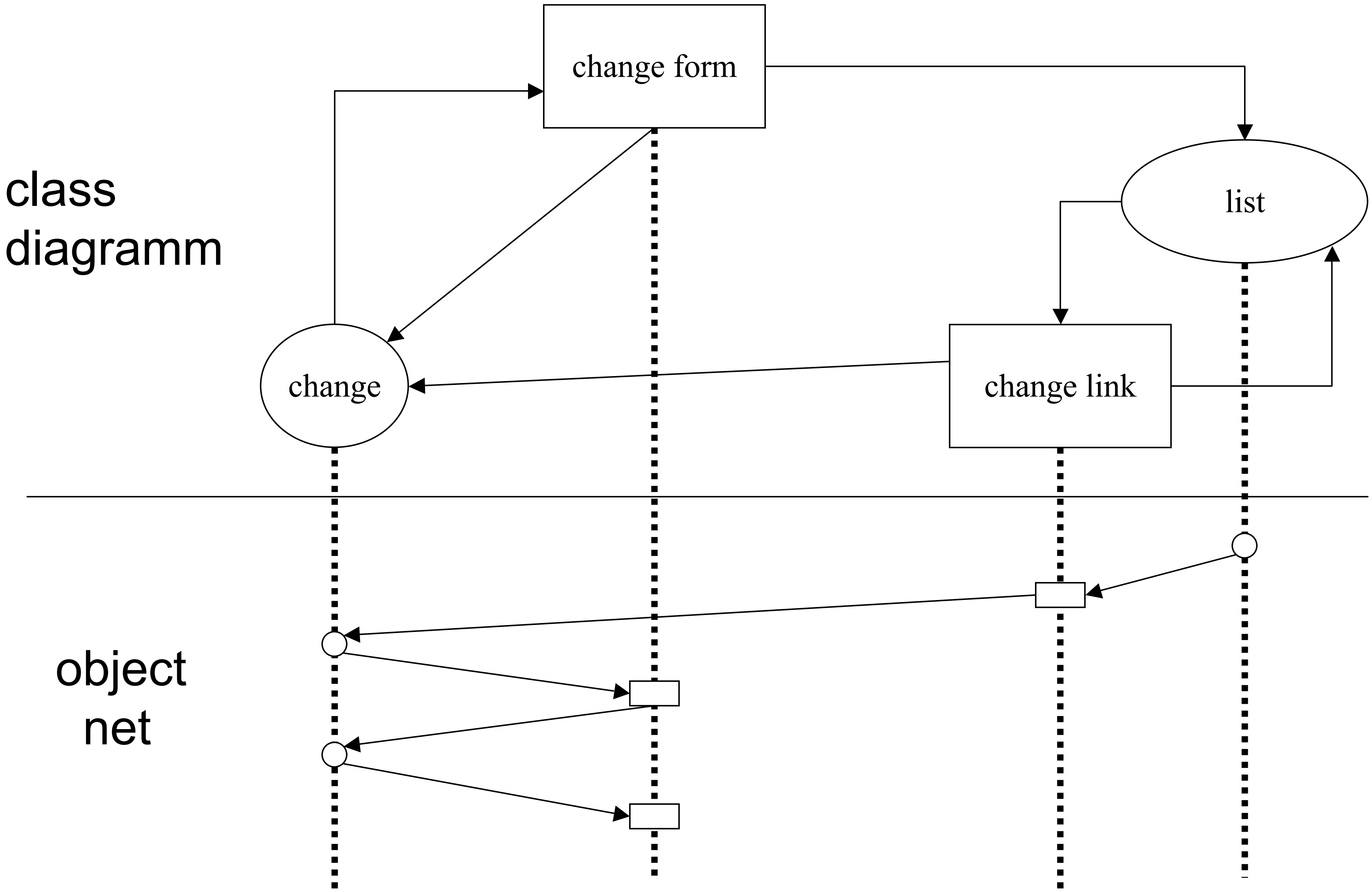
Basic Modeling Idea

- no need to model the ephemeral, fine grained interaction within one page
- modelling submit/response style user interfaces with a state transition diagram that
  - Is bipartite: states are partitioned into two sets:
    - client states (client pages)
    - server states (server actions)
  - Has a bijective mapping from states to message types

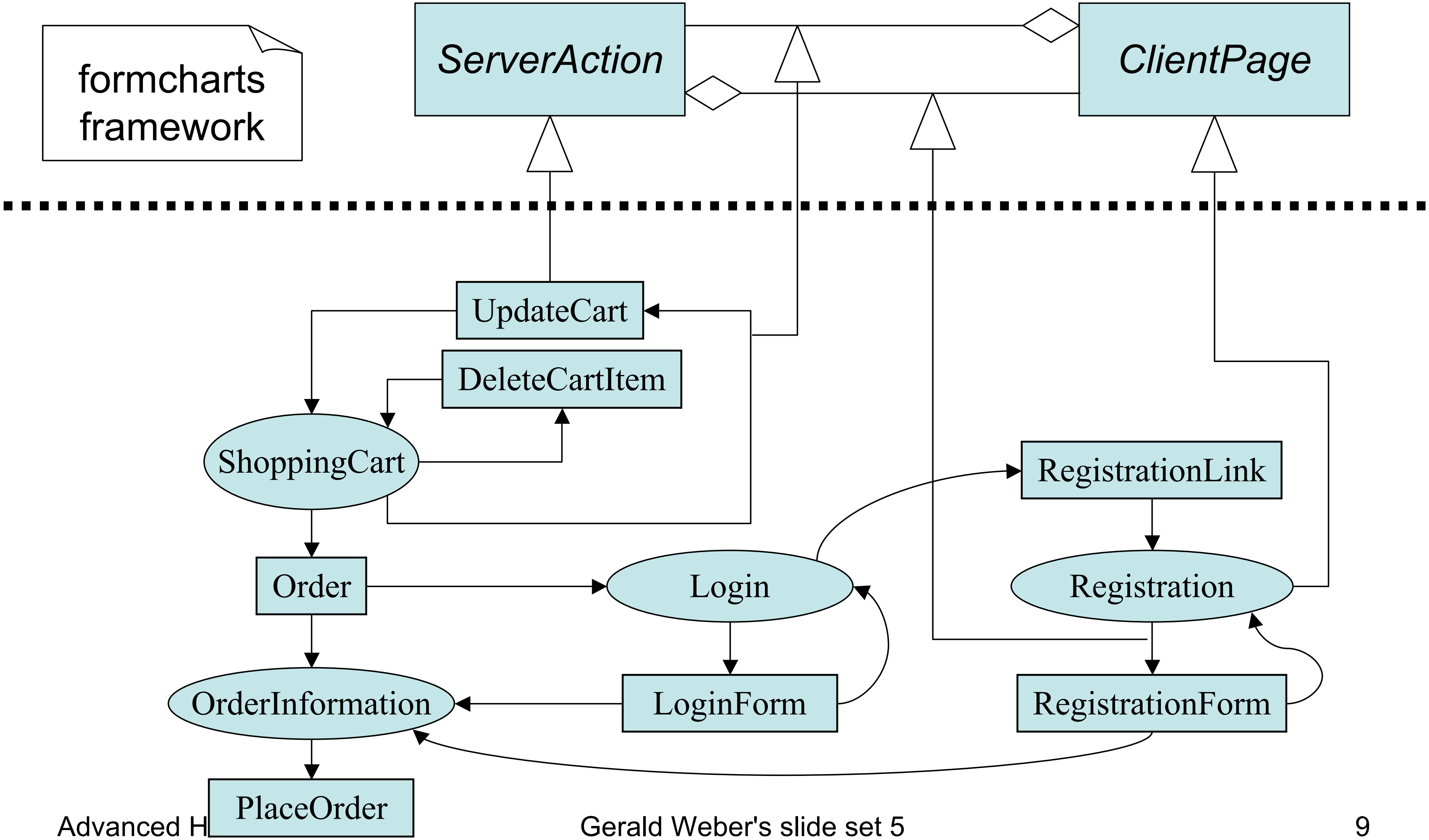
formcharts



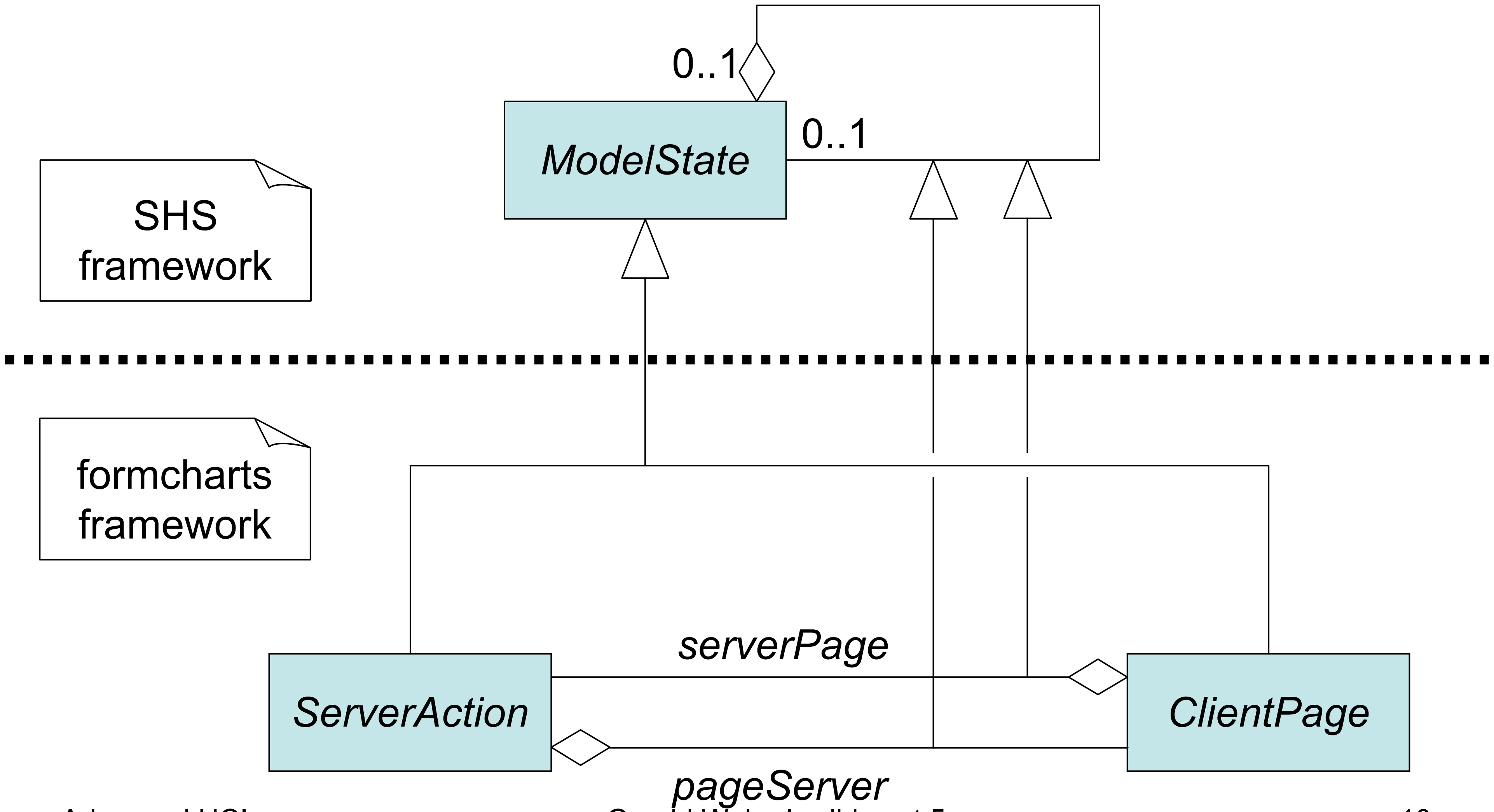
The object net over a formchart is a path



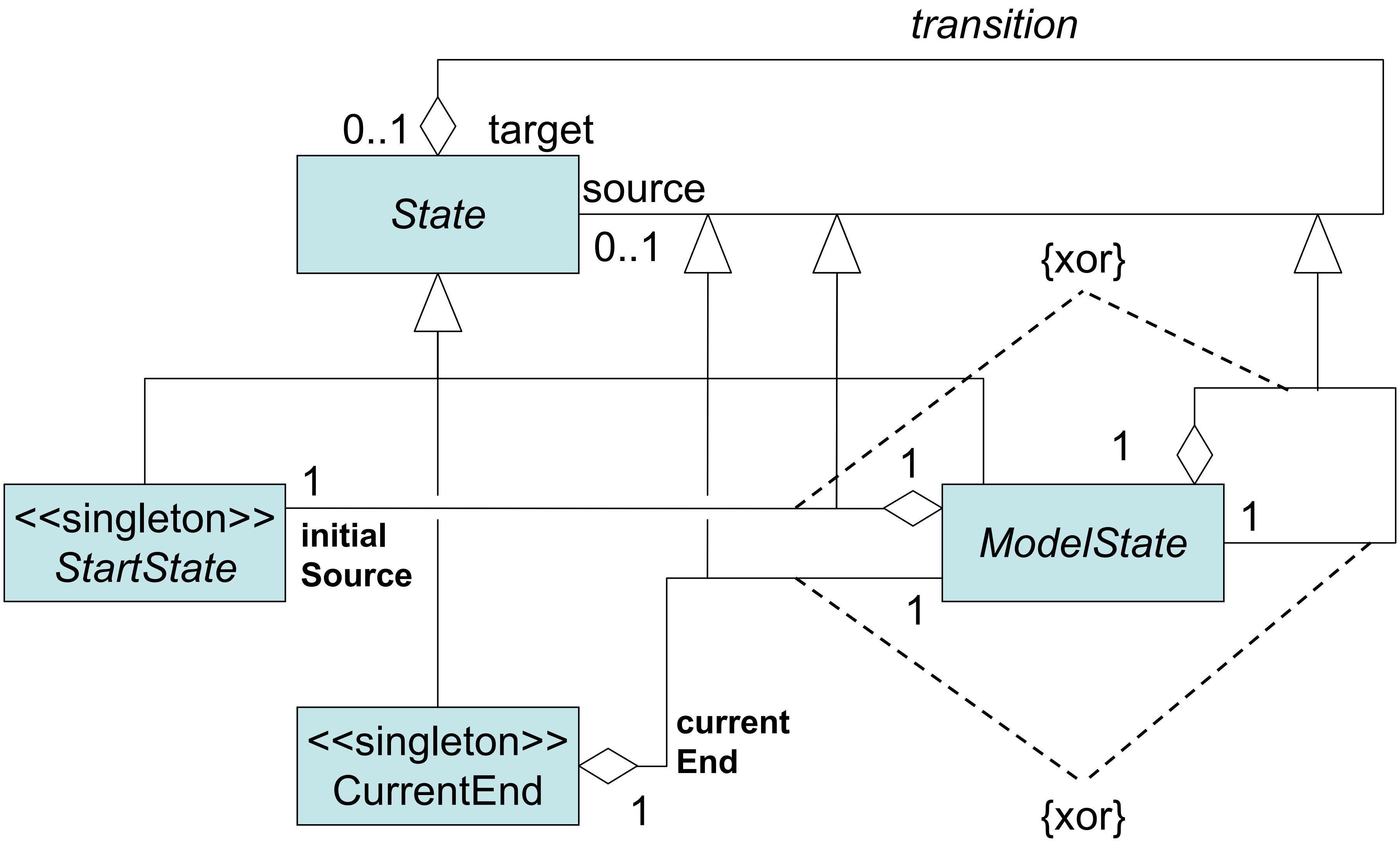
Formcharts semantics



A Framework for formcharts



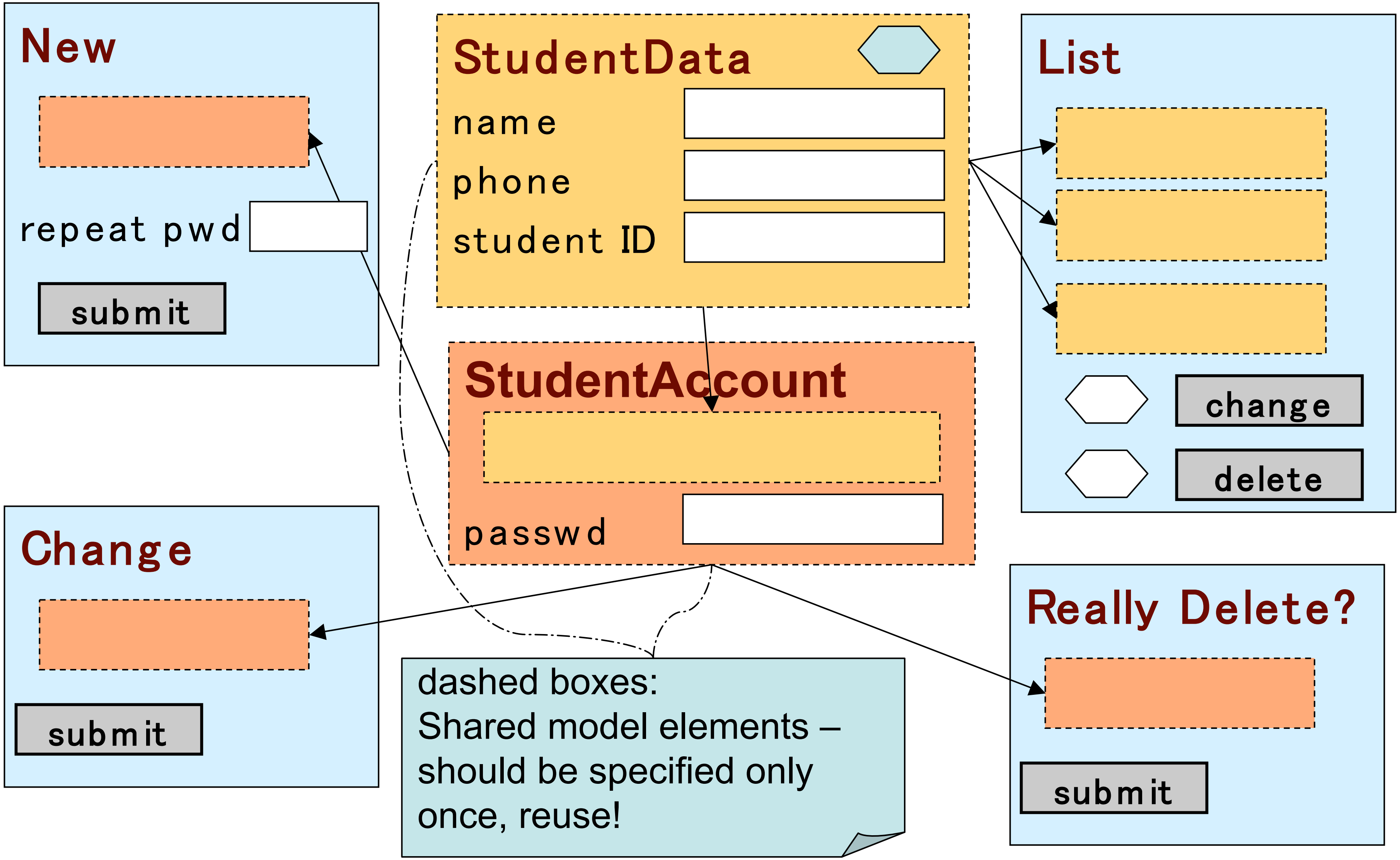
A framework for State History Diagrams



Form-Oriented Analysis

- Approach tailored to submit/response style interfaces
- System interface model is...
  - given by a bipartite finite state machine
  - related to a layered data model (message model, opaque references,information model)
  - annotated with dialogue constraints for…
    - specifying server reaction
    - narrowing dialogue capabilities
  - visualized by the form chart
- Modeling method allows tools for
  - generating a system from a model
  - reverse engineering: infer a model form the system

Visualization of reuse in a web application



New

repeat pwd

submit

StudentData

name

phone

student ID

StudentAccount

passwd

List

change

delete

Change

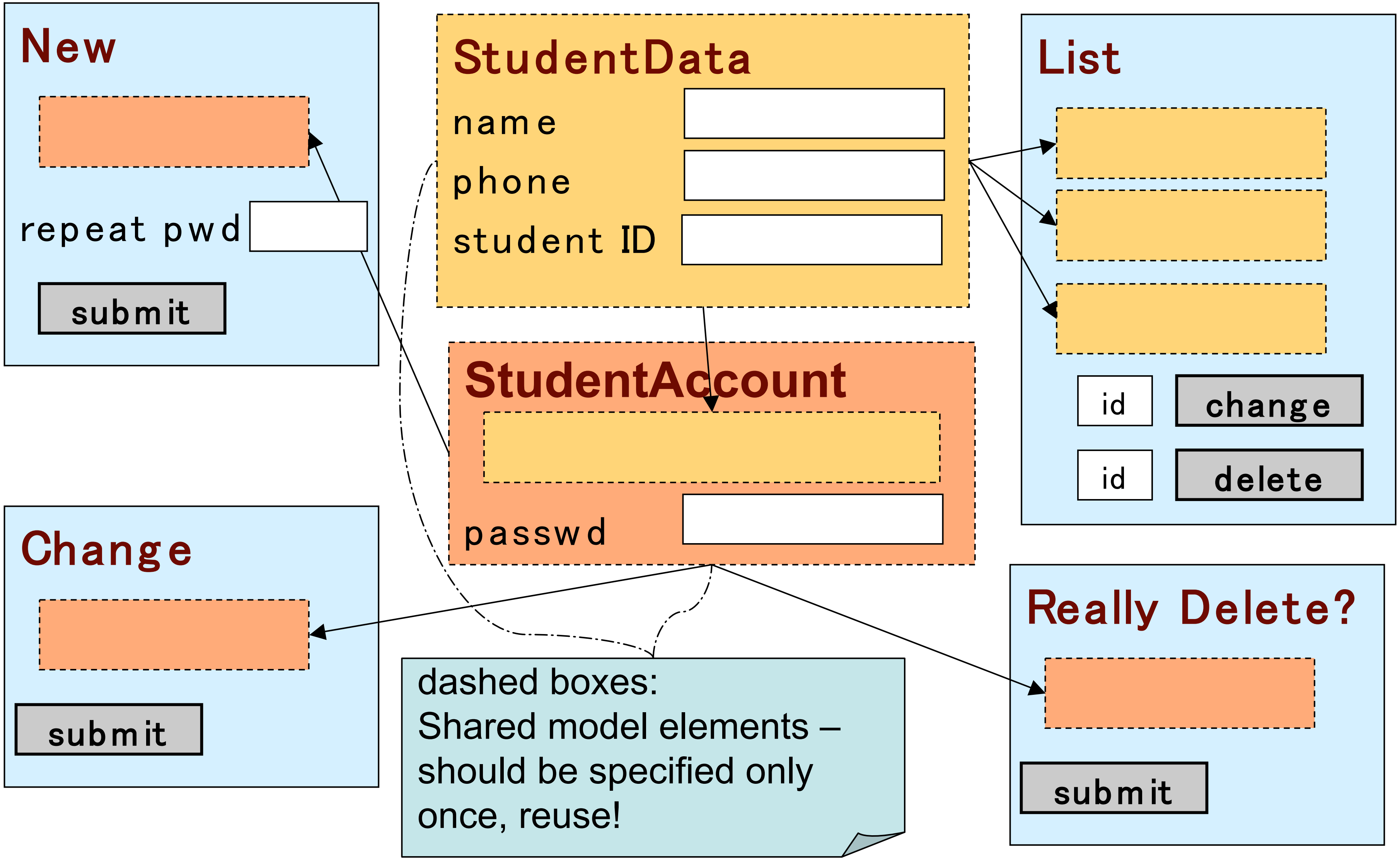
submit

Really Delete?

submit

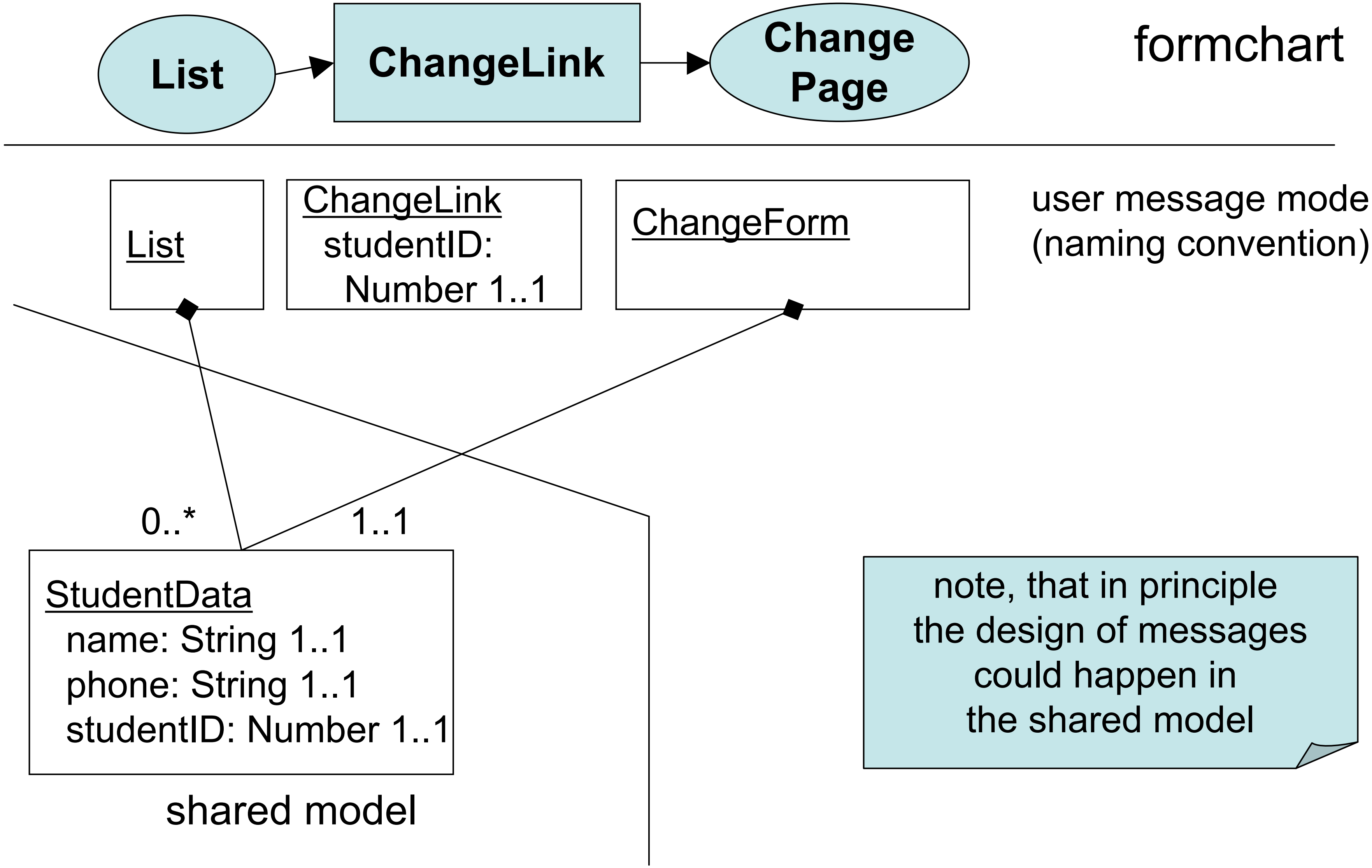
dashed boxes:
Shared model elements – should be specified only once, reuse!

Visualization of reuse without opaque IDs



dashed boxes:
Shared model elements – should be specified only once, reuse!

Specifying message types: class diagrams



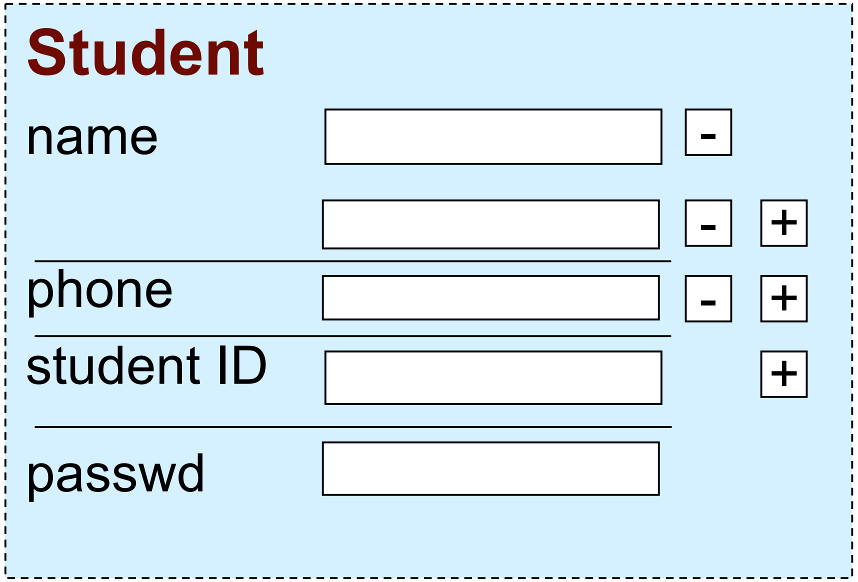
note, that in principle the design of messages could happen in the shared model

Multiplicities of attributes

- The concept of forms is technology independent.

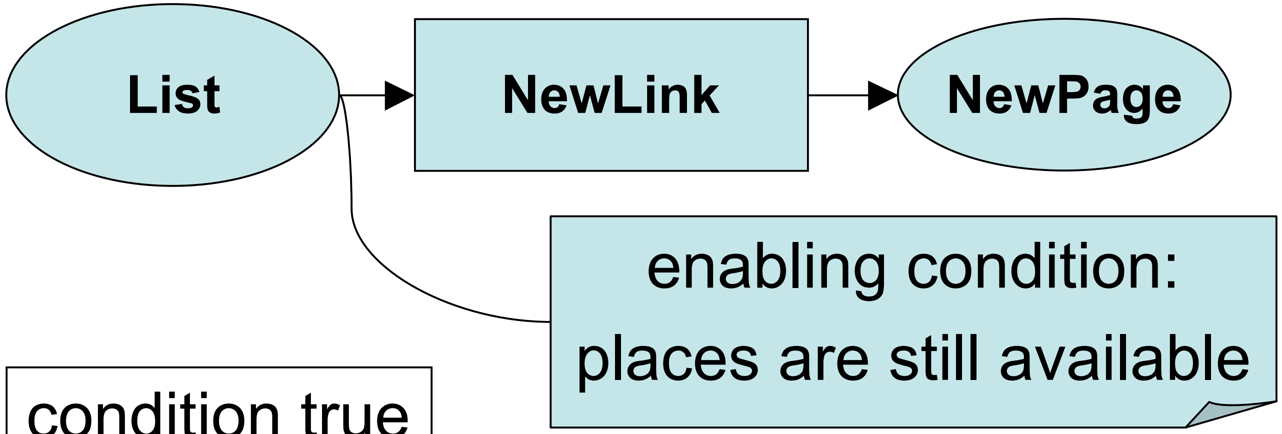
StudentData
name: String 0..*
phone: String 0..*
studentID: Number 1..*
passwd: Pwd 1..1

shared model entry



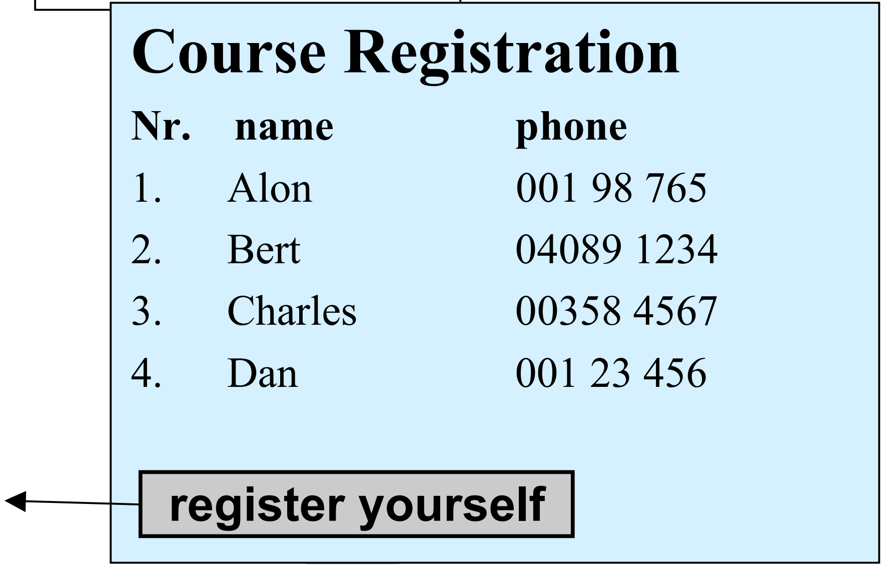
form-snippet

Enabling Condition



enabling condition:
places are still available

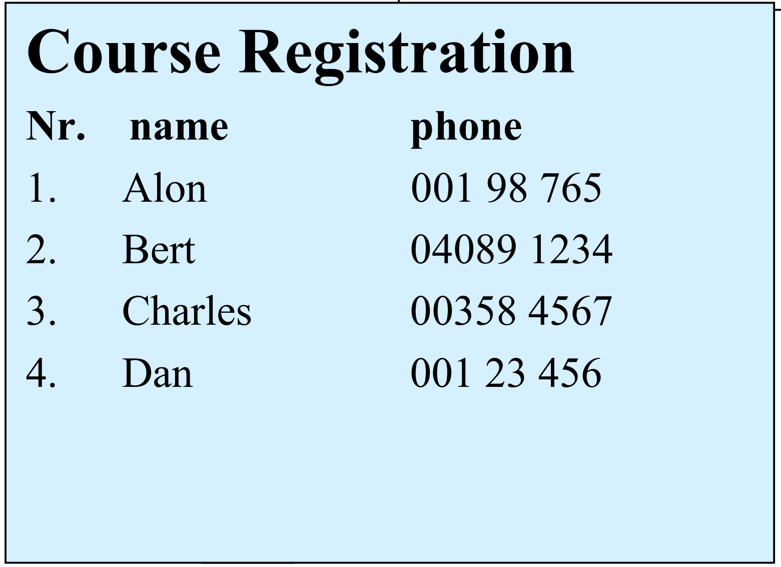
condition true



Course Registration

Nr.	name	phone
1.	Alon	001 98 765
2.	Bert	04089 1234
3.	Charles	00358 4567
4.	Dan	001 23 456

condition false



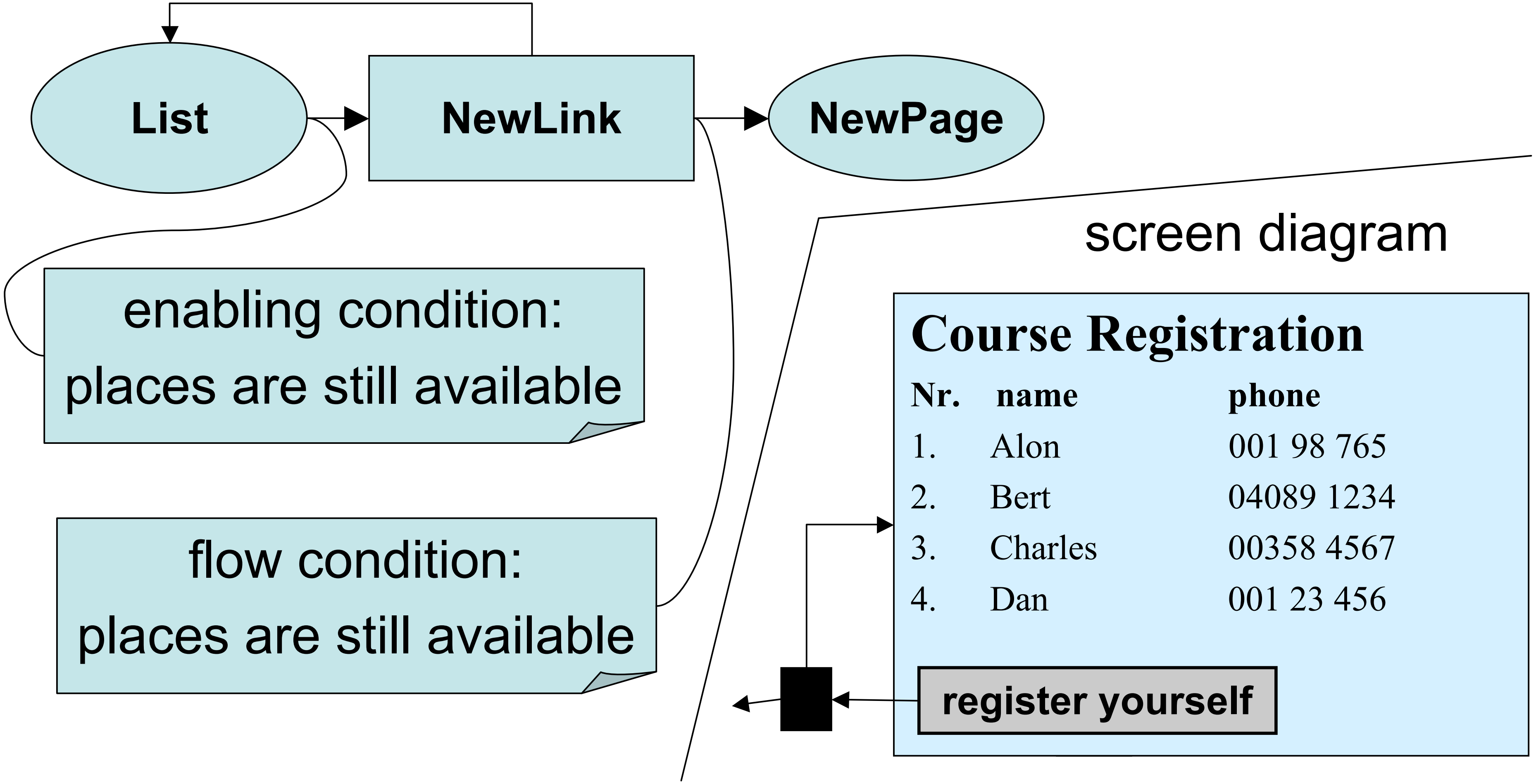
Course Registration

Nr.	name	phone
1.	Alon	001 98 765
2.	Bert	04089 1234
3.	Charles	00358 4567
4.	Dan	001 23 456

enabling conditions are evaluated at page creation time

A multi-user aware model

remember: pages are immutable reports



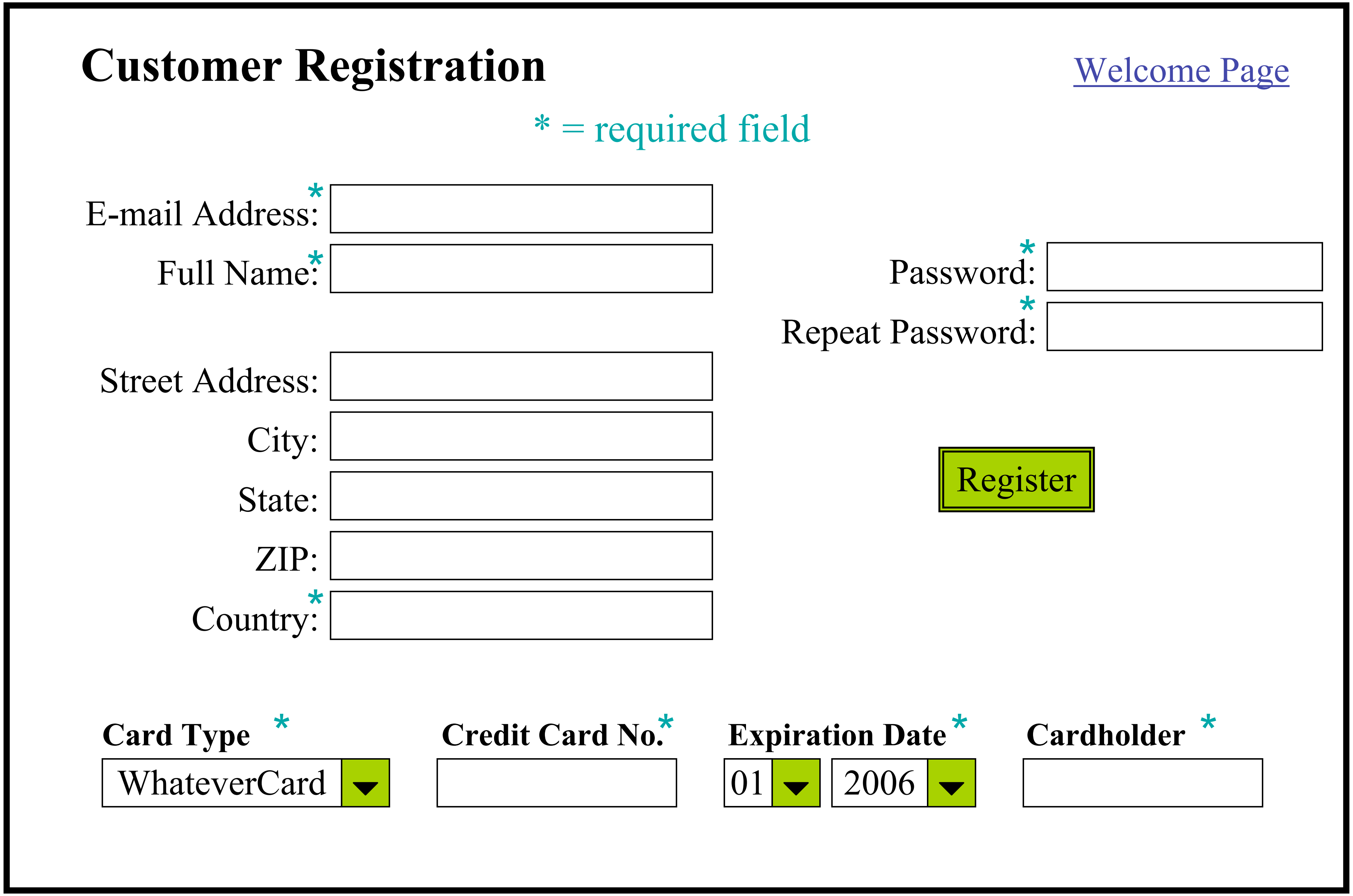
enabling condition:
places are still available

flow condition:
places are still available

screen diagram

Course Registration

Nr.	name	phone
1.	Alon	001 98 765
2.	Bert	04089 1234
3.	Charles	00358 4567
4.	Dan	001 23 456



Customer Registration

Welcome Page

* = required field

E-mail Address:*

Full Name:*

Password:*

Repeat Password:*

Street Address:

City:

State:

ZIP:

Country:*

Register

Card Type * WhateverCard ▼

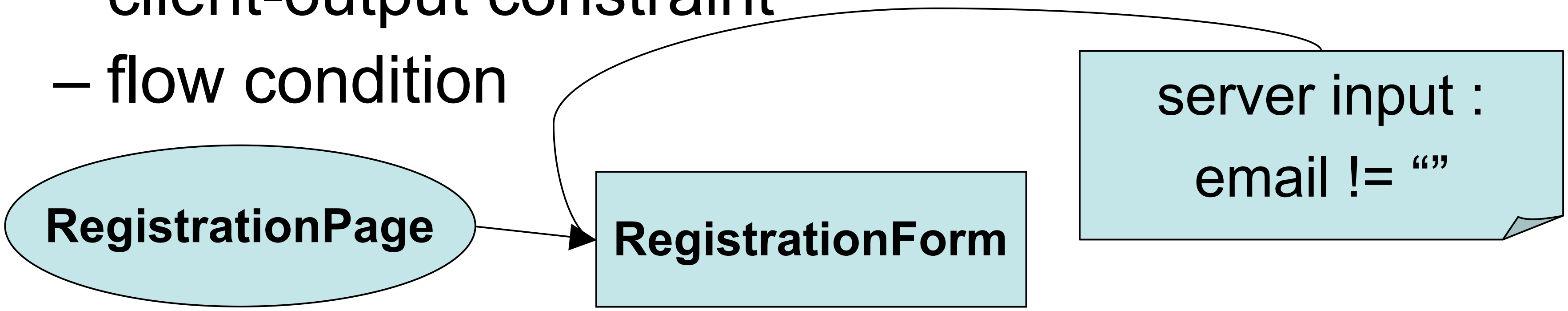
Credit Card No.*

Expiration Date * 01 ▼ 2006 ▼

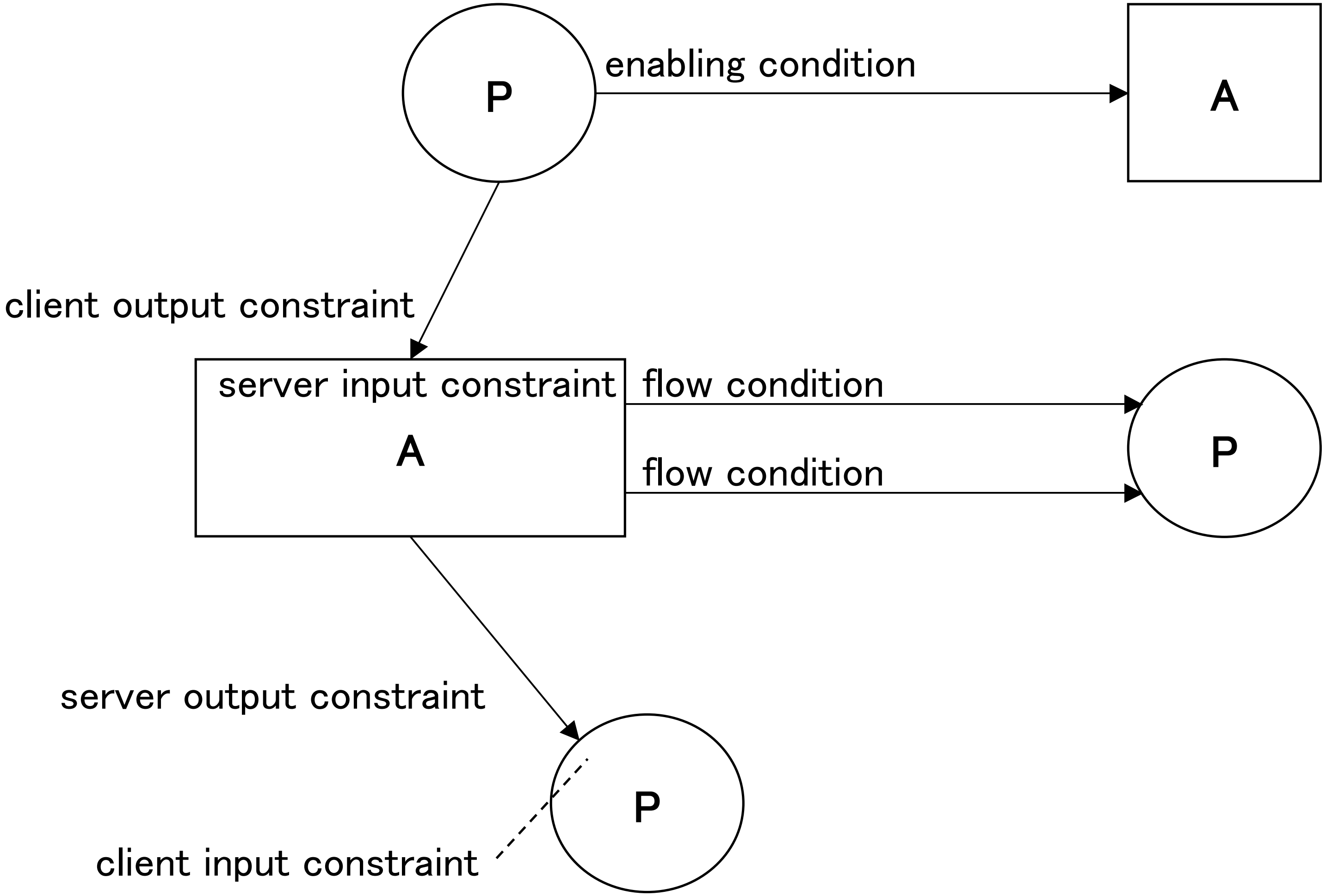
Cardholder *

Required fields and server input constraints

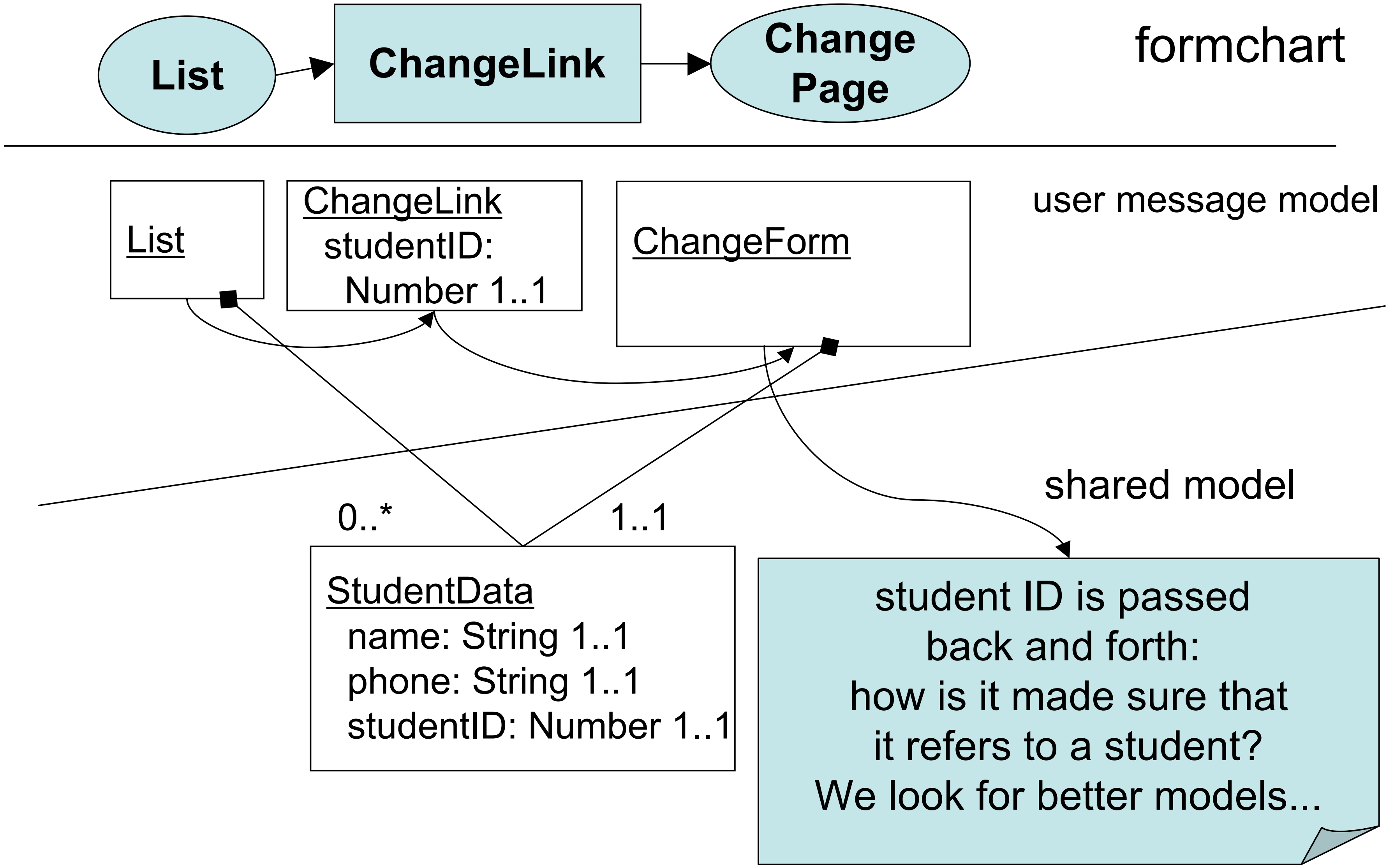
- Required fields are an example of a server input constraint.
  - server input constraint are supposed to be transformed into other constraints in the later specification process.
  - Express a separation of concerns.
- can be remodeled as
  - client-output constraint
  - flow condition



Formchart notational elements

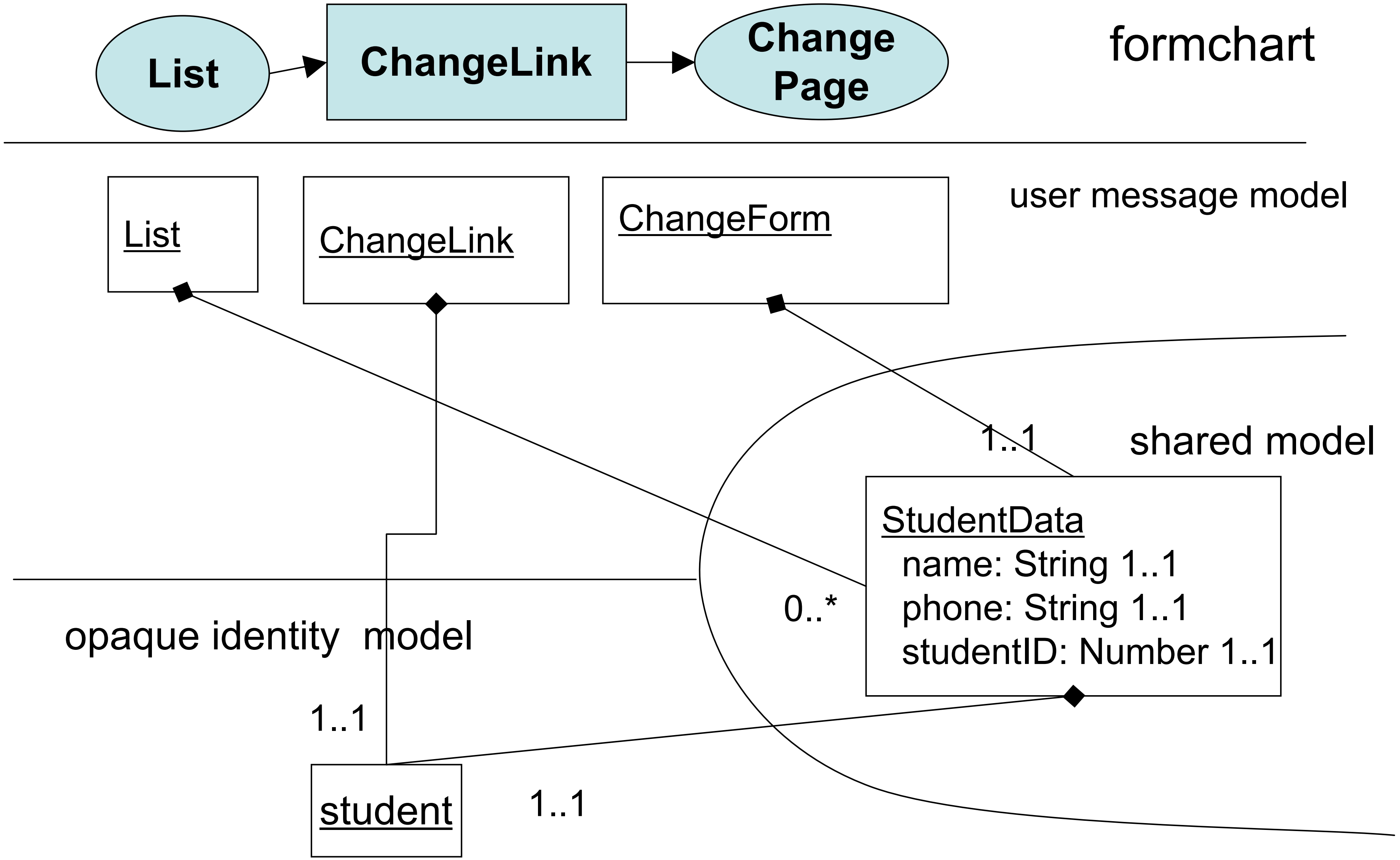


problem: open transmission of identity

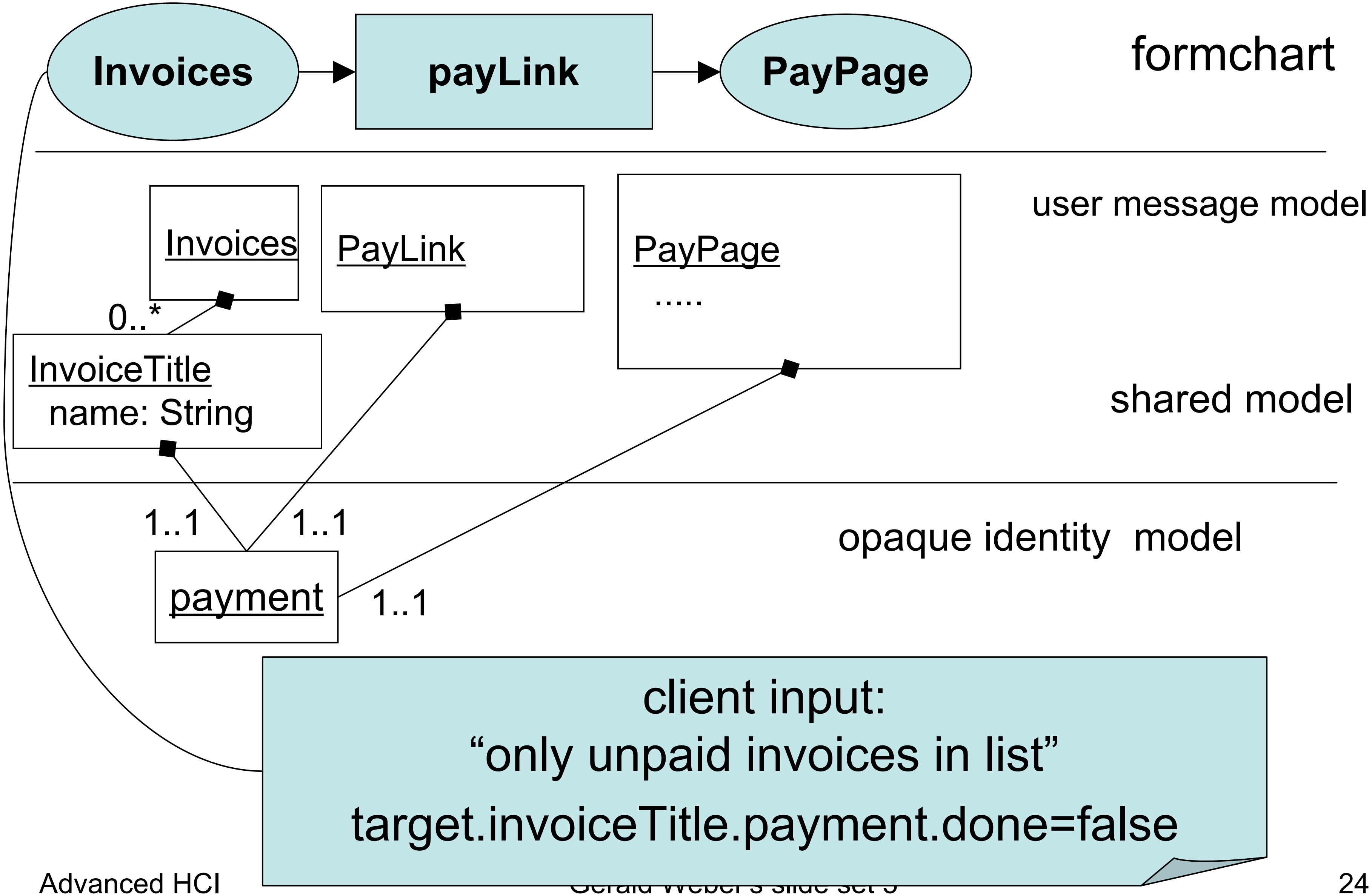


student ID is passed back and forth: how is it made sure that it refers to a student? We look for better models...

Solution: opaque identities

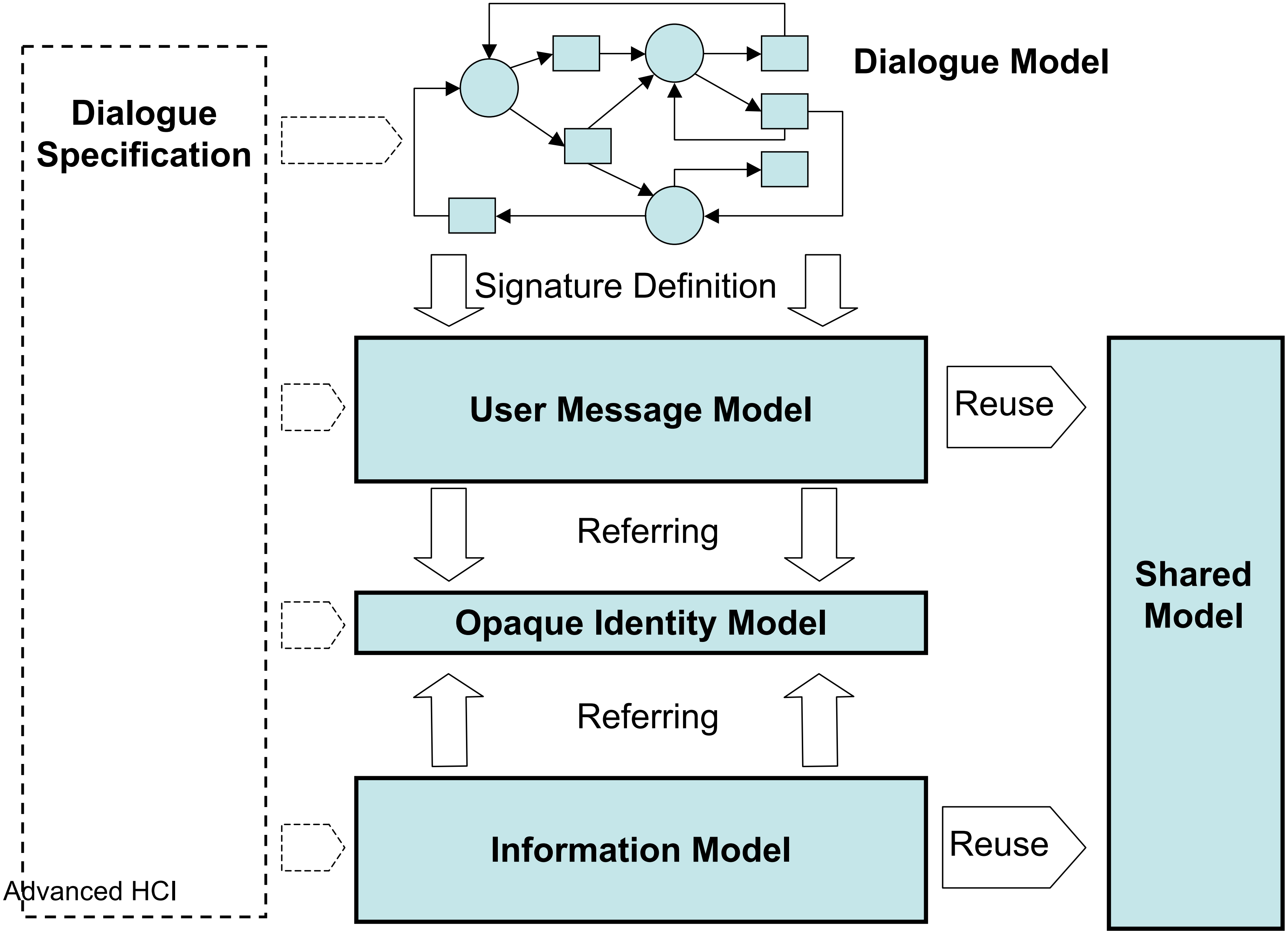


Client input specification



client input:
“only unpaid invoices in list”
target.invoiceTitle.payment.done=false

Form-oriented specification



Dialogue Model

Dialogue Specification

Signature Definition

User Message Model

Reuse

Referring

Opaque Identity Model

Referring

Information Model

Reuse

Shared Model