Knowledge Transformation from Task Scenarios to View-based Design Diagrams

Nima Dezhkam
Kamran Sartipi

{dezhkan, sartipi}@mcmaster.ca

Department of Computing and Software
McMaster University
CANADA

SEKE'08
July 1, 2008

1

Outline

- Task Scenarios
- Scenarios in knowledge extraction
- Proposed framework
  - Scenario generation
  - Scenario decomposition
  - Design construction
- Fast-food restaurant case-study
- Conclusions

2

Task scenarios

We define a "scenario" as a structured narrative text describing a system's requirements in terms of system-environment interactions at business rule level.

- Different scenario representations:
  - Simple text
  - Graphical representation
  - Relational algebra, etc.
- Common applications of scenarios:
  - Requirement elicitation and analysis,
  - Design representation,
  - Testing
  - Maintenance

3

Scenarios in Knowledge Extraction

- Enhancement of scenario generation by using scenario schemas
- Formal representation of scenarios using tabular expression is introduced in order to simplify the tasks of scenario verification, validation and integration
- Schema definition for semantic model of scenarios to help requirement refinements
- Modular representation of the scenarios to support the reusability of the scenarios in different design contexts

Proposed Framework for Scenario to Design Diagram Transformation

Transforms a set of text-based scenarios into two types of design diagrams, as: Data and Function.

Properties:

- Uses a **scenario syntax** that allows us to define well-structured scenarios.
- Uses a **scenario schema** to parse the scenarios and populate an object base of actors, actions, and dependencies.
- Uses **Guidelines** for transforming the elements in the object base into design diagrams.

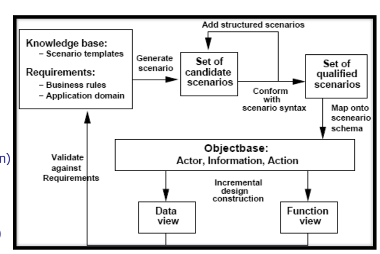
5

Proposed Framework …

Stage 1
(Scenario generation)

Stage 2
(Scenario decomposition)

Stage 3
(Design construction)



6

Scenario Syntax

Stage 1: Scenario Generation

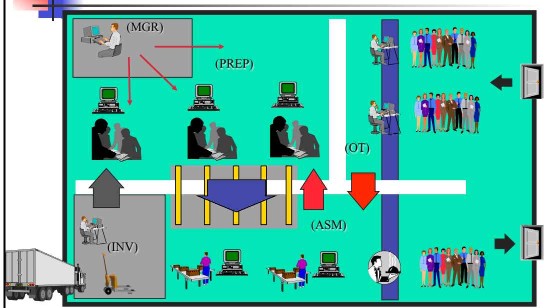
$$Scenario \;:\; \{Actor + \{Constraints\}^{0..M}\}^{1..N} \;+$$
$$\{Action + \{Constraints\}^{0..M}\}^{1..N} \;+$$
$$\{Working\ Information + \{Constraints\}^{0..M}\}^{1..N}$$

- **Example scenario:** *"Order taker adds a menu item to an incomplete order."*

7

Fast Food Restaurant System

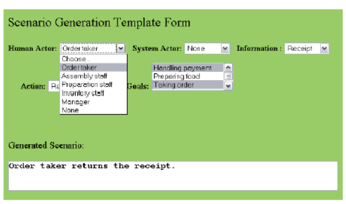
Physical view



Sample Scenario Template Form

Stage 1: Scenario Generation

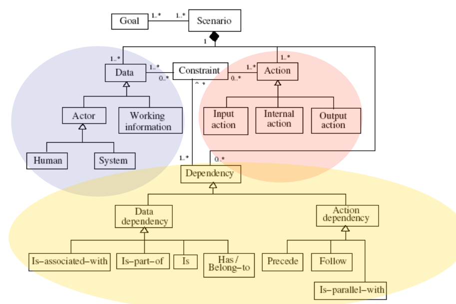
- A scenario template forms the knowledge- base of a fast-food restaurant system



9

Proposed Scenario Schema

Stage 2: Scenario Decomposition



10

Example of Scenario Decomposition: One of the 12 Scenarios

Stage 2: Scenario Decomposition

- Sample Fast-food scenario: Scenario #1: *"Order taking station computes and reports the price of the orders."*
- Decomposed scenario:

$$goal = taking\ order\ \&\ handling\ payment$$
$$actor|_{System} = order\ taking\ station$$
$$information = order,\ price$$
$$action|_{Internal} = compute\ price$$
$$action|_{Output} = report\ price$$
$$data\ dependency|_{Is\ associated\ with} = (1, OT\ station, n, order)$$
$$data\ dependency|_{Belong\ to} = (price, order)$$
$$action\ dependency|_{Precede} = (compute\ price, report\ price)$$

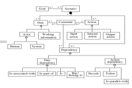
Objectbase Created from 10 Scenarios

Stage 2: Scenario Decomposition

$Index$	$Actor\|_{System}$	$Actor\|_{Human}$	$Working\ information$	$Action\|_{Input}$	$Action\|_{Internal}$	$Action\|_{Output}$
1	OT Station	-	order,price	-	compute price	report price
2	OT Station, ASM station	-	paid order	-	-	send paid order to ASM station
3	-	order taker,OT station	ID&password	-	login to system	-
4	-	order taker	order	-	initiate order	-
5	-	order taker	menu item,unpaid order	-	add/remove menu item	-
6	-	order taker,OT station	cash-in	enter cash-in	-	-
7	-	order taker	order	-	defer payment	-
8	-	order taker	order	-	review	-
9	-	order taker	unpaid orders	-	call-back	-
10	-	order taker	change/receipt	-	-	return change/receipt

Index	Is-associated-with	Belong-to	Is-part-of	Follow	Precede
1	-	(price,order)	(report price, compute price)	(report price, compute price)	-
2	-	-	(1,paid order,1,order)	(send paid order to ASM station, report price), ...	-
3	-	(ID&password,order taker)	-	-	(login to system, send paid order to ASM station), ...
4	(1,order taker,n,customer order)	-	-	(initiate order, login to system)	(initiate order, compute price)
5	(n,menu item,1,order)	-	-	(edit order, initiate order), ...	(edit Order, compute price), ...
6	-	(cash-in,order)	-	(enter cash-in, report price), ...	(enter cash-in, send paid order to ASM station), ...
7	-	-	-	(defer payment, edit order), ...	-
8	-	-	-	(review orders, login to system)	-
9	-	-	(1,unpaid order,1,order)	(call-back unpaid orders, login to system)	(call-back unpaid orders, enter cash-in), ...
10	-	(change/receipt,order)	-	(return change/receipt, enter cash-in), ...	(return change/receipt, send paid order to ASM station)

Design Construction Guidelines: Data View



Stage 3: Design Construction

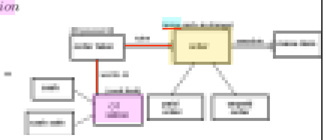
- **Step 1:** Extract all instances of *Actor*, *Working information*, and *Data dependency* classes from the object base and apply the following rules on them:
  1. Instances of *Actor* and *Working information* are candidate entities/attributes.
  2. Instances of *Is* dependency imply generalization and inheritance relationships, i.e., A *Is* B, means A is sub-entity of B, or B is super-entity of A.
  3. Candidate entities/attributes that appear on either side of a *Is*, *Is-associated-with*, or *Is-part-of* relationship are considered as entities.
  4. Instances of *Has* and *Belong-to* dependencies are used to identify the attributes of the entities, i.e., A *Has* B (or B *Belongs-to* A) means B is an attribute of entity A.
  5. Instances of *Is-associated-with* dependency imply candidate association relationships.
  6. Instances of *Is-part-of* dependency imply candidate decomposition relationships.
- **Step 2**: Depict every entity by a rectangle, every attribute of an entity as a bubble connected to it and label them by their names. Every relationship between two entities can be represented by a line connecting them. Label every relationship according to the type of dependency it came from, e.g., "is", "is-part-of", etc.

Generated E-R Diagram

Stage 3: Design Construction

$goal = taking\ order\ \&\ handling\ payment$
$actor|_{System} = order\ taking\ station$
$information = order,\ price$
$action|_{Internal} = compute\ price$
$action|_{Output} = report\ price$
$data\ dependency|_{Is\ associated\ with} = (1, OT\ station, n, order)$
$data\ dependency|_{Belong\ to} = (price, order)$
$action\ dependency|_{Precede} = (compute\ price, report\ price)$

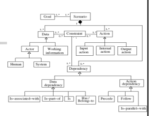
Decomposed scenario



Part of generated ER diagram for fast-food restaurant

14

Design Construction Guidelines: Function view



Stage 3: Design Construction

- **Step 1**: Extract all instances of *Action*, *Action dependency*, and *Constraint* classes from the object base and apply the following rules on them:
  1. Instances of *Action* class are the functions.
  2. Instances of the *Follow* and *Precede* dependencies determine the time-order of execution of the functions. To simplify the diagram generation, transform all the *Precede* dependencies to *Follow*, i.e., for all functions f1 and f2, change "f1*Precede* f2" to "f2 *Follow* f1"
  3. The participants of a *Is-parallel-with* dependency must be executed concurrently.
  4. The conditions for a function to follow another is determined by the *Constraints* related to the function, actor, and working information in the corresponding scenario that the "following" appears.
- **Step2**: Generate Follow+ relationship (the transitive-closure of the *Follow*).
- **Step 3**: Sort the functions in ascending order based on the number of the functions they follow, i.e., based on the number of times they appear on the left hand side of a *Follow* relationship.
- **Step 4**: Starting from the first of the list, depict the function (name A) with a square and label it by its name. List all the functions that *Follow* A. Use AND and OR connectors when necessary. Next, all arrows are labeled with the triggering conditions obtained in rule "4" above. Finally, remove A from the list and repeat *Step 4*, until the list is empty.

List of Actions in Order Taking Component and the "Follows" Relation

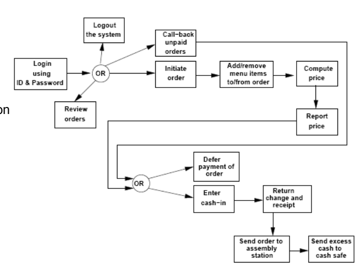
Index	Action	Follows+
1	Login using ID & password	-
2	Logout the system	1
3	Review orders	1
4	Initiate order	1
5	Call-back unpaid orders	1
6	Edit orders	1,5
7	Compute price	1,5,6
8	Report price	1,5,6,7
9	Defer order payment	1,5,6,7,8
10	Enter cash-in	1,4,5,6,7,8
11	Return change & receipt	1,4,5,6,7,8,10
12	Send order to assembly station	1,5,6,7,8,10,11
13	Send excess cash to cash safe	1,4,5,6,7,8,10,11,12

16

Generated Function Diagram

Stage 3: Design Construction

Part of generated Function diagram for fast-food restaurant



17

Conclusion

- Task scenarios: are used to generate the ingredients of the design diagrams.
- Scenario generation: generating a set of structured text-based scenarios that conform with a regular expression syntax.
- Scenario decomposition: mapping generated scenarios onto scenario schema which allows parsing the structured scenarios and generating instances of schema classes.
- Design construction: generating design diagrams in Data and Function views using the decomposed scenarios and based on a set of guidelines.

18

Knowledge Transformation from Task Scenarios to View-based Design Diagrams

Nima Dezhkam
Kamran Sartipi

{dezhkan, sartipi}@mcmaster.ca

Department of Computing and Software
McMaster University
CANADA

SEKE' 08
July 1, 2008

19

Request For Proposal (RFP)

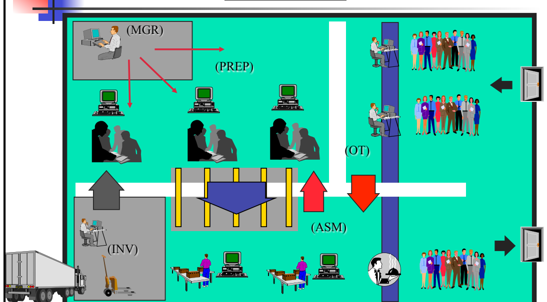
MacFood restaurant system

MacFood is a new restaurant chain which offers fast food to the customers. It uses an in-store computer system to assist order-taking and payment, food preparation, delivery, and inventory.
Orders and payments are taken by staff using "touch-screen" displays.
Kitchen and delivery staff view orders on displays, and register the status of orders by pressing buttons of the keypads.
Inventory of the food and supplies is tracked by the computer system.
The restaurant manager is able to configure the system to set menu items , ingredients, prices, inventory levels, and store setup.
The following section briefly introduces the various units of the **MacFood** System.

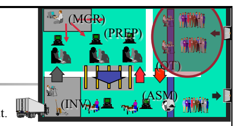
The following slides discuss the produced SRS after requirement analysis phase

Fast Food Restaurant System

Physical view

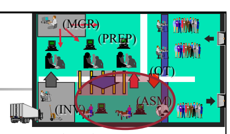


Order-Taking Unit



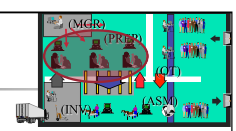
- This unit sets up customer orders and handles payment.
- Menu items are selected from the restaurant-menu by touching buttons on the touch-screen.
- Selection of an item causes it to be added to the current order (which is displayed in a scrollable window on the screen), and the subtotals / tax of the order are displayed.
- An order can be paid anytime between its set-up and delivery to the customer.
- The system keeps the cash balance of each order-taking station and has facilities for supporting "cash float" (i.e., a specified amount of cash in the order-taking station at the beginning) and "skim" (i.e., a threshold amount of cash, which once exceeded, must be transferred to the cash balance) of each station.
- Each order is handled by only one order-taker; however, the orders could be stored in a list and each order-taker in the system can access this list to service the stored orders.

Assembly Unit

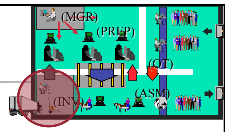


- When an order is set up, the kitchen should be informed to prepare the order-items.
- When the computer system determines that all items of an order are available in the chutes, the order can be assembled.
- Each available assembly-station picks the order and displays it on its screen.
- The assembly-stations use screen and keypad for interaction with the staff.
- The staff assemble the orders, and using keypads inform the system. If the order is paid, the system allows the delivery of the order to the customer, otherwise, the delivery will be postponed to the time that the order is paid.
- If the system indicates that an order can be filled, but the chutes do not contain a sufficient quantity of some order's item, the staff report the shortage to the system to be prepared.

Food Preparation Unit

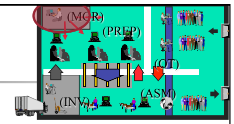


- In order to prepare an order, the system distributes order-items among preparation stations, equipped to prepare certain items of the restaurant-menu.
- In general, more than one station is capable of making a particular item. Each station has a screen and a keypad. Similar items of different orders are grouped together.
- Considering the number of items assigned to each station and its current load of work, the system decides whether to send the items to that station or not.
- The screen of the preparation-station displays a list of items and their quantities.
- Kitchen staff prepare the required quantity of an item, put them in the "chute", and using the keypad inform the system.
- There is one chute for each menu item.
- Menu items are prepared in response to real and anticipatory demands. Anticipatory demands are set up by the manager to shorten the average time of waiting for food.



Inventory Unit

- The inventory unit in the system keeps track of the consumption of all materials used for preparation and packaging of the order-items.
- We refer to these materials as "raw-materials". This unit has a very close interaction with the preparation unit.
- The system keeps stock, and the inventory of raw materials is updated dynamically.
- The arrival of new materials into storage is entered into the system by the staff, and the consumption of the materials is dictated by the recipes of food-items.
- To preserve stock integrity, the system assumes a minimum threshold for usage of each menu-item in the system. If the number of a certain menu-item drops below this threshold, it is considered unavailable and the inventory unit alerts the order-taking unit to inhibit taking that item.



Management Unit

The management-unit of the restaurant system is responsible for setting up:

- Active stations in order-taking, preparation, and assembly units.
- System tables such as restaurant-menu, recipes, anticipated demands, minimum number of menu-items, and raw-materials in stock.
- List of menu-items to be prepared by each preparation station.
- Cash skim and float.
- Different applicable taxes.
- System time and date.

E-R diagram of the Restaurant System