USAGE-CENTERED SOFTWARE ENGINEERING: AN APPROACH TO DEVELOPING BUSINESS INFORMATION SYSTEM

Arogundade Oluwasefunmi 'Tale & Akinwale Adio Taofiki.

Department Of Computer Science,
University Of Agriculture, P.M.B 2240, Abeokuta , Ogun State.

ABSTRACT

This paper examines one aspect of usage-centered software development, modeling system requirement using USE_CASES. Usage-centered design is a systematic, model driven approach to visual and interaction design with an established record of effectiveness in a wide variety of settings and areas of application. To this end, a general approach to fairness conditions and timer concepts are developed with usage-centered engineering. As a case study, this modeling tool is applied to the well known generalized car hiring problem where a complete model is included. Designing and implementation of the tool allows monitoring of all components of car hiring process as a complete system since the modeling tool adequately convey the understanding of the system to the users. This will enhance the client experience and as in the case of software development can lead to lower costs for the organization.

Keywords: information system, usage-centered design, sequence diagram, conceptual model, relationship.

INTRODUCTION

The introduction of information technology in business management has brought about a change in the way businesses are carried out. Capturing and documenting system requirements have proved to be a critical factor in the outcome of a successful business information systems development projects. Documenting the requirement from the perspective of the users in a manner that they can understand promotes user involvement which greatly enhances the probability for the success of any project.

The use of information technology (IT) in business has generally been geared towards improving the business. Since maximizing profit is the major goal of any business organization then there is a need to ensure and put in place all that is needed to make the business successful. One of the basic factors is incorporating information technology into business processes.

The hardest single part of building information system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements to people, machine and other software systems. So designing a system is more than jumping into coding but knowing first what that system would do, rather than how the system would perform the operations[8]. A review of the literature reveals that poor specification of user requirements is a major reason for project failures (for example), [10]. Nonetheless system failure rates continue to be well above 50% and perhaps as high as 85% (for example), [6, 12].

Usage-centered design [2, 3] is a systematic, model-driven approach to visual and interaction design for user interfaces in software and web-based applications. On projects of widely varying scope and scale in a variety of application areas [1, 2, 9, 11, 13], usage-centered design has proved capable of delivering superior designs [14]. As the name suggests, usage-centered design differs from conventional user-centered approaches primarily in a shift of focus from users per se to usage, that is, to the tasks to be accomplished by users. This difference in emphasis is reflected in differing practices. Instead of trial-and-error design through repeated cycles of prototyping, user feedback, and usability testing, usage-centered design constructs robust and highly refined abstract models and more or less directly derives from these an initial design requiring only limited usability testing and minimal refinement.

Usage-centered design integrates readily with software engineering precisely because it is grounded in a strong engineering orientation and was developed from the outset to be compatible with object-oriented software engineering [7].

The very same extensions and refinements to well established models and techniques, such as actors and use cases [4,7], that drive the user interface design can also be employed directly for software engineering .

Dr Ivar Jacobson, the originator of Use-Case Model used use-case modeling as the frame work for his methodology which he successfully used for developing object-oriented information system [8]. Software is

intended to serve the needs and support the interests of its human users, yet software engineering is often weakest when it comes to addressing the critical areas of user requirements, usability, user interfaces, and interaction design. Often these concerns are either minimized, ignored, or relegated to "other" disciplines as a responsibility outside the scope of software engineering [11]. Usage-centered design has proved to be a valuable aid in meeting challenges of determining what a system is required to do from user and stakeholder perspective and it is now widely recognized as a best practice for defining, documenting and understanding of a information systems functional requirements [8].

USE CASE MODELING

The steps required to produce this model are the following:

(a) Identify business actors
(b) Identify business requirements use cases
(c) Construct use-case model diagram
(d) Document business requirements use-case narratives

CONCEPTUAL MODEL

Conceptual model illustrates abstract and meaningful concepts in the problem domain. The creation of concepts is the most essential object-oriented step in analysis or investigation of the problem domain for building genuinely extensible software with reuse [5]. This work decomposes car hiring system into individual concepts or objects. Figure 1 illustrates conceptual model of car hiring system.

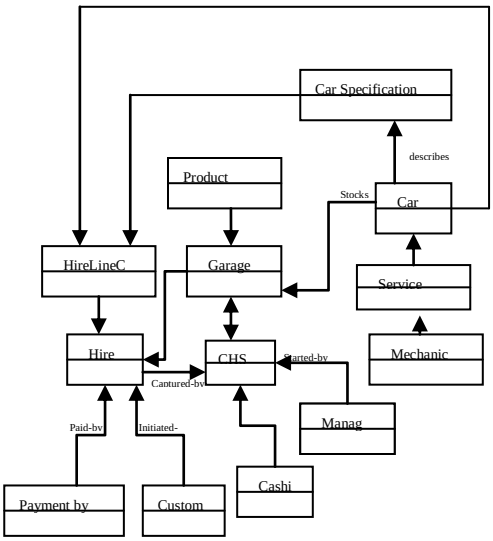


Fig. 1 A Conceptual Model for the Car Hiring

SYSTEM DESIGN

Designing this model consists of system functions, essential use cases, and sequence diagrams.

SYSTEM FUNCTION

Systems Functions are what the system suppose to do. It contains reference number, object information and category. Table 1 illustrates system functions while table 2 illustrates the payment functions of car hiring system.

ESSENTIAL USE CASES

An essential use case describes the sequence of events of major actor of car hiring system.

Ref #	Function	Category
R1.0	Customer supplies demographic data	Evident
R1.1	Record the underway (current) sales – the estimated mileage covered	Evident
R1.2	Calculate current Transaction	Evident
R1.2.5	Register a new mechanic	Evident
R1.2.6	Remove Mechanic	Evident
R1.2.7	Assign Car	Evident
R1.2.8	Reassign Car to another mechanic	Evident
R1.3	Capture Hired Car information from data supplied by the customer	Evident
R1.4	Store inventory when car is hired	Hidden
R1.4.5	Update Car Inventory	Hidden
R1.5	Log Completes Transaction	Hidden
R1.6	Cashier needs no login in order to use the system	Evident
R1.7	Provides a consistent and persistent storage mechanism	Hidden
R1.8	Display description and price of Car (s) hired	Evident
R1.9	Display Information when car is due for service	Evident
R2.0	Display the service history of a particular car	Evident
R2.1	Display the Log History of all the Hire Transactions	Evident
R2.2	Remove Customer	Evident
R2.3	Register New Car	Evident
R2.3.5	Remove Car	Evident
R2.3.6	Assign Mechanic to Car	Evident

Table 1. System functions

Payment Functions

Ref #	Function	Category
R2.4	Handle cash payments, capturing amount tendered and calculating balance due	Evident
R2.5	Store the log of cash payments into the appropriate database	Hidden
R2.6	Refund Customer	Evident

Table 2. Payment functions

Use Case: Register Customer
Actors: Customer (initiator), Cashier
Purpose: To gets and store Customer Information for future reference.
Overview: A Customer arrives at the booking office and supplies his/her name (personal Information), if it was not recorded earlier that is in the case of a new customer.
Cross References: functions R1.0

Use Case: Register Car
Actors: Company's Manager
Purpose: To register new cars and delete old ones that is no longer in use by the company
Overview: The Manager updates the system with the details of new cars (inventory) acquired by the company and

deletes the forgone cars and adds the mechanic responsible for the car.
Cross Reference: function R1.4.5, R2.3.6

Use Case: Hire Car
Actor: The customer (initiator), Cashier
Purpose: To get the type of car hired
Overview: Check the mileage covered and maybe there is any damages incurred by the customer.
Cross Reference: function R1.0, R1.1, R1.2, R1.3, R1.8, R1.9

Use Case: Service Car
Actors: Manager (initiator), Mechanic
Purpose: To make sure that car is in good condition
Overview: Depending on the distance covered – minor service at every 600 miles and major service at every 1200 miles.
Cross Reference: R1.3, R1.4, R1.4.5, R1.8, R1.9.

Use Case: Hire Details
Actors: Manager, System Admin
Purpose: To have knowledge of cars hired
Overview: date of the beginning and end of the hire are recorded
Cross Reference: function R1.0, R1.1, R1.2, R1.3, R1.4

Use Case: Pay By Cash Only
Actors: Customer (initiator), Cashier
Purpose: capture the cost of hired car
Overview: on return of the car(s), the date, mileage covered, damages incurred and the cost is calculate, the customer might end up with a refund if the estimated mileage is less than mileage covered.
Cross Reference: functions R2.4, R2.5, R2.6

Use Cases: Remove Customer
Actors: Customer, System Administrator (initiator)
Purpose: delete the file of un-regular customer (s)
Overview: When a customer have not patronized for a specific period of time, his / her file is deleted.
Cross Reference: R2.2
Use Case: Register Mechanic
Actors: Mechanic, System Administrator
Purpose: register and assign mechanic to a specific car
Overview: Mechanic is registered and assigned to a particular car, a particular mechanic is responsible for a particular car for servicing.
Cross Reference: R1.2.5

Use Case: Remove Mechanic
Actors: Mechanic, System Administrator
Purpose: To delete mechanic details
Overview: Mechanic (s) that is/are no longer with the company are deleted and their cars are transferred to another mechanic.
Cross Reference: R1.2.6

Car hiring system Use-Case Model Diagram

Use-Case model diagram of car hiring system describes the process of essential activities of car hiring system. It relates various actors of car hiring system together. Figure 2 shows part of car hiring system use-case diagram

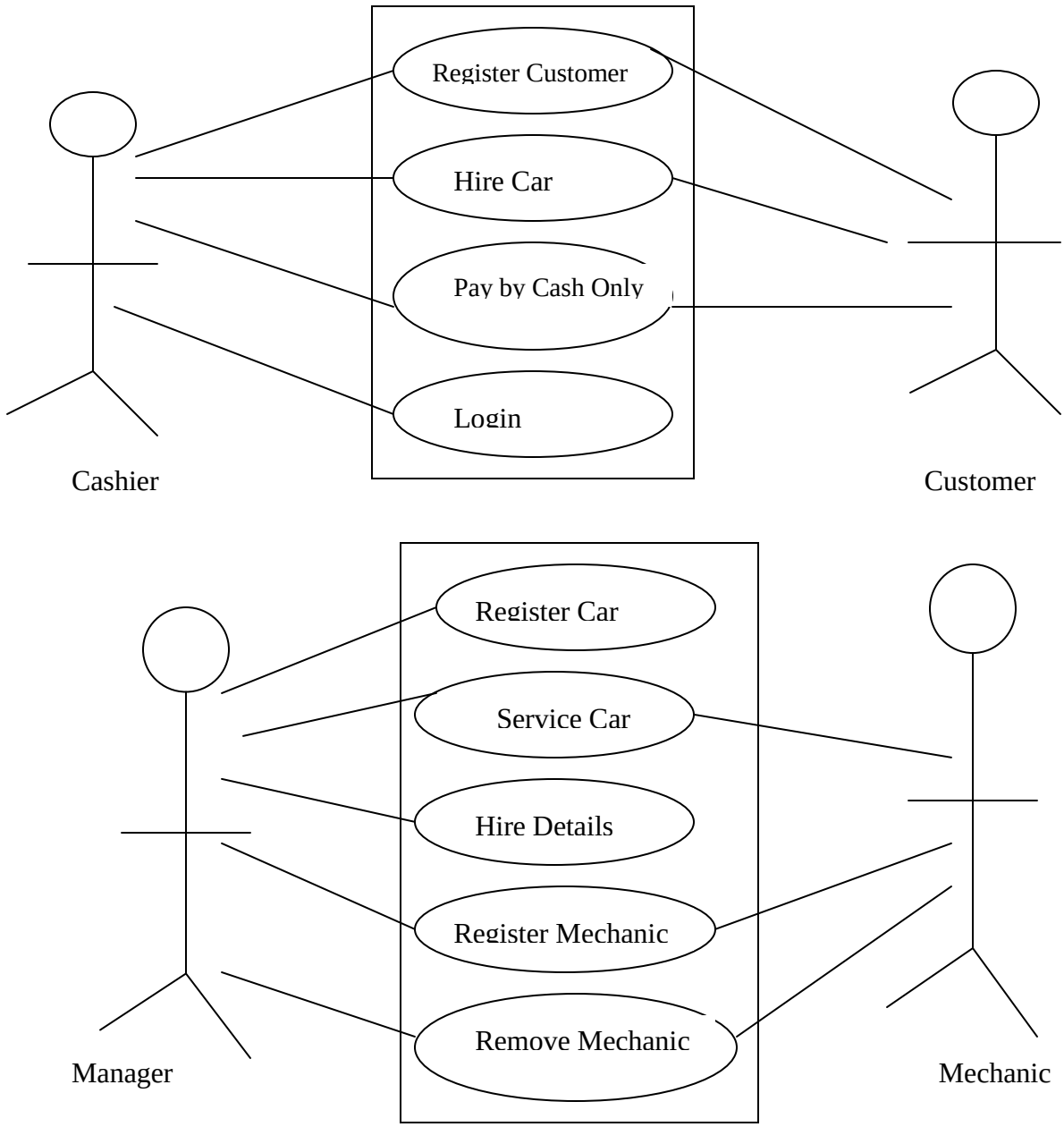
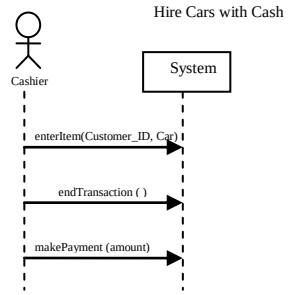


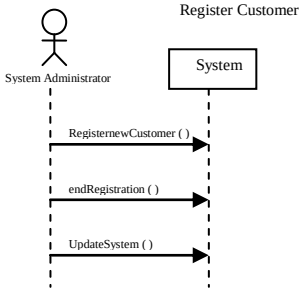
Figure 2: Car Hiring System Use-Case Model Diagram

SYSTEM SEQUENCE DIAGRAMS FOR THE CAR HIRING SYSTEM (CHS)

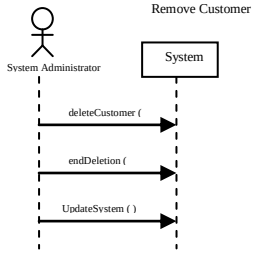
System sequence diagram for car hiring system identifies the operations of car hiring actors and the order in which they carry out their operations. Also it details the effects of such operations in the system as shown.



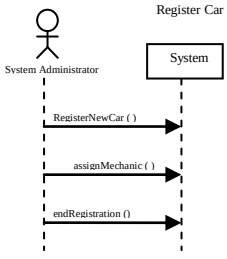
CHS Sequence diagram for Hire Cars with Cash Use Case



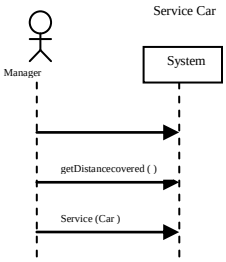
CHS Sequence diagram for Register Customer Use Case



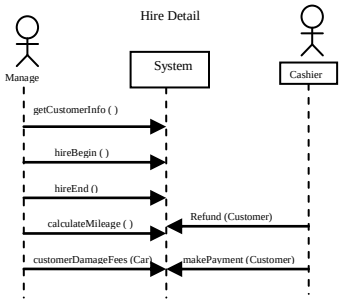
CHS Sequence diagram for Remove Customer Use Case



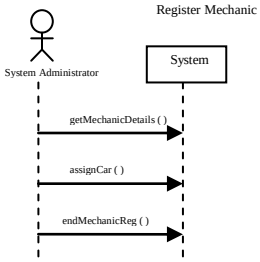
CHS Sequence diagram for Register Car Use Case



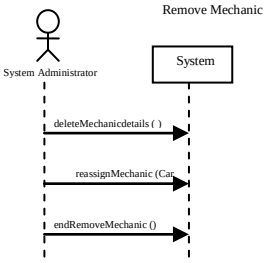
CHS Sequence diagram for Service Car Use Case



CHS Sequence diagram for Register Car Use Case



CHS Sequence diagram for Register Mechanic Use Case



CHS Sequence diagram for Remove Mechanic

IMPLEMENTATION

Use-Cases of car hiring system were implemented with visual Basic programming language. The implementation generates Multiple Document Interfaces (MDI) which allows to navigate from one form to another.

MAIN MENU

It starts up with Log on form which confirms whether a user is registered or not. A non – register user is not allowed to transact business without registering.

SEARCH CUSTOMER

This searches the Database for the Customer's ID, if found a Car Detail Form is loaded to screen if not gives the User the opportunity to retype the Customer's ID, because only registered Customers are allowed on the system. A sample of search customer information form is depicted in figure 3.

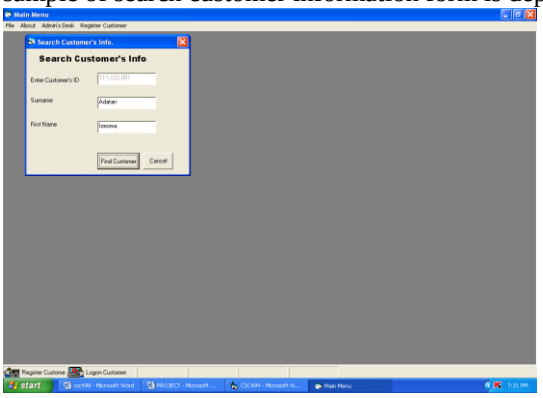


Figure 3: Search Customer Form

REGISTER CUSTOMER

This Form is used to register the user with the user's name, address, telephone number, drivers' license number as pictured in figure 4.

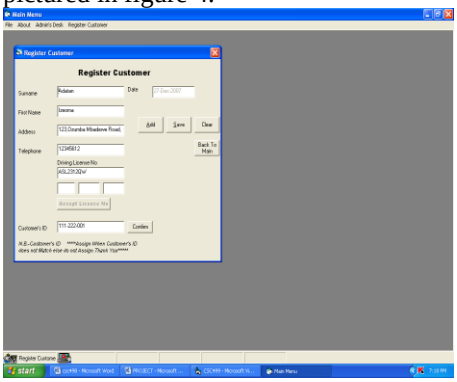


Figure 4: Register Customer Form

CAR DETAILS FORM

It enlists the number of available cars to the user and allows the user to make a choice of cars as illustrated in figure 5.

ADMIN DESK

The Admin Desk Comprises of a menu for Administrative issues such as:

Edit Customer Information:

Here the customer can be deleted or modified.

Edit Mechanic

Mechanic is added, deleted or assigned to Car.

Set Daily Hire Rate:
The Daily Hire Rate is inputted by the administrator which is what the cashier use for calculation.

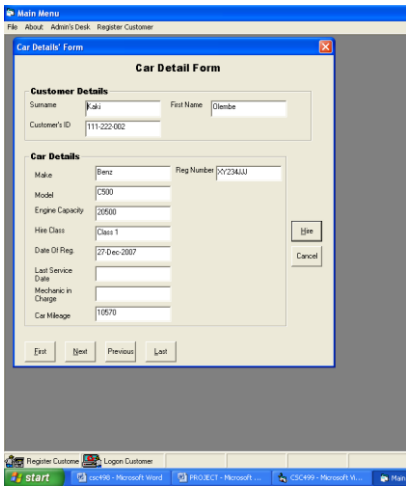


Figure 5: Car Details Form

Finish Transaction
After a Customer returns a car, the transaction is completed by recording the mileage covered and the car's mileage is update.

Get Transaction History
All the Transactions ever made on the system can be viewed at any point in time.

Service Details
The Service History of a particular car can be searched for through the system.

Edit Car Details
This is where a car is added to the fleet of cars already owned by the company; old cars that are no longer needed by the company can also be deleted through this service.

CONCLUSION

In this paper, an attempt was made to use Usage-centered engineering to ease and facilitate system designs, making sure that the system is not done half - way but completed. A Conceptual Model is employed to help know the relationships between all the classes of the system, letting us know how they relate with one another. The Business transaction can be done in a more friendly way, with accuracy, speed and precision.

One of the main contribution of this paper is the development of a system that provides the end – user with high level and users – friendly interfaces for Car Hiring. The System was design with the Usage – Centered approach that is having the work users are doing and the tasks they are trying to accomplish in mind. The End – user would definitely perform better using this system since the tools that were used better support the work being done. Secondly, since the models employed in usage-centered design are simple and easy to understand and develop, alternative designs and major revisions to architecture can be explored more rapidly in future through this model.

REFERENCES

1. Anderson, J., Fleek, F., Garrity, K., and Drake, F. "Integrating Usability Techniques into Software Development. IEEE Software, 18(1), January/February,2001.
2. Constantine, L. L., and Lockwood, L. A. D. "Usage-Centered Engineering for Web Applications. IEEE Software, 19(2), March/April,2002.
3. Constantine, L. L., and Lockwood, L. A. D. Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley, Reading, MA, 1999.
4. Constantine, L. L., and Lockwood, L.A.D. " Structure and Style in Use Cases for user Interface Design." In M. van Harmelan (Ed.), Object Modelling and User Interface Design. Addison-Wesley, Boston, 2001.

5. Constantine, L. L., and Lockword, L. A. D.(2003) Usage-centered software engineering: An Agile Approach to Integrating Users, Users Interfaces, and Usability into Software Engineering Practice. IEEE Software.
6. Dalcher, D., and Drevin, L. Learning from information systems failures by using narratives and ante-narrative mehods. In proceedings for the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologist on Enablement Through Technology. South African Institute for Computer Scientists aqnd Information Technologists.
7. Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. Object-Oriented software Engineering: A Use Case Driven Approach. Addison-Wesley, Reading, MA, 1992.
8. Jeffey L. Whitten, Lonnie D. Bentley, Kevin C. Dittman, (2004). *System Analysis And Design Method New York* Mc Graw Hills Publications.
9. Patton, J. "Extreme design; Usage-Centered Design in XP and Agile Development." Contantine (Ed.), forUSE 2002: Proceedings of the First International Conference on Usage-Centered, Task-Centered, and Performance-Centered Design. Ampersand Press, Rowley, MA, 2002.
10. Ross, J.K. (2004). Project and requirements management- Driving Software project success. Journal of Validation Technology 10, (3), 192.
11. Strope, J. "Putting Usage-Centered Design to Work: Clinical Applications." In L. Constantine (Ed.), forUSE 2002: Proceedings of the first International Conference on Usage-centered, and performance-centered Design. Ampersand Press, Rowley, MA, 2002.
12. The Standish Group International, Inc.(1999) "Chaos: A Recipe for Success" (electronic version). www.pm2go.com/sample-research/chaos1998.pdf
13. Windl, H. ' Designing a Winner: Creating STEP 7 Lite with Usage-Centered Design. "In L. Contantine (Ed.), forUSE 2002: Proceedings of the first International Conference on Usage-centered, and performance-centered Design. Ampersand Press, Rowley, MA, 2002.
14. Windll, H., and Constantine, L., ' Performance-centered Design: STEP 7 Lite. " Winning submission, Performance-Centered Design 2001, http://foruse.com/pcd/
15. Zhiming Liu (2002). Object – Oriented Software Development with Unified Modeling Language, United Nations Universities /International Institute for Software Technology Report No 259.