Н.Н. Лесбек
магистрант,
Казахско-Британский университет
Алматы/Казахстан

Transforming use case diagrams into sequence diagrams using Kermeta *(Payment Server)*

What is Payment Server?

This payment server system is a simplified version of a transport service which is enabled to perform payments for different types of services such as apartment bills, phone bills, tax bills, etc. through the banking terminals.

Usually such systems involve:

- User – a client of a bank;
- Terminal – special device to perform requests transmission;
- Payment server – a centralized server which coordinates the providers and payment channels;
- Provider – a service provider who cares about fulfillment of the service request and provides the bill for operations.

Metamodel : Use Case diagram

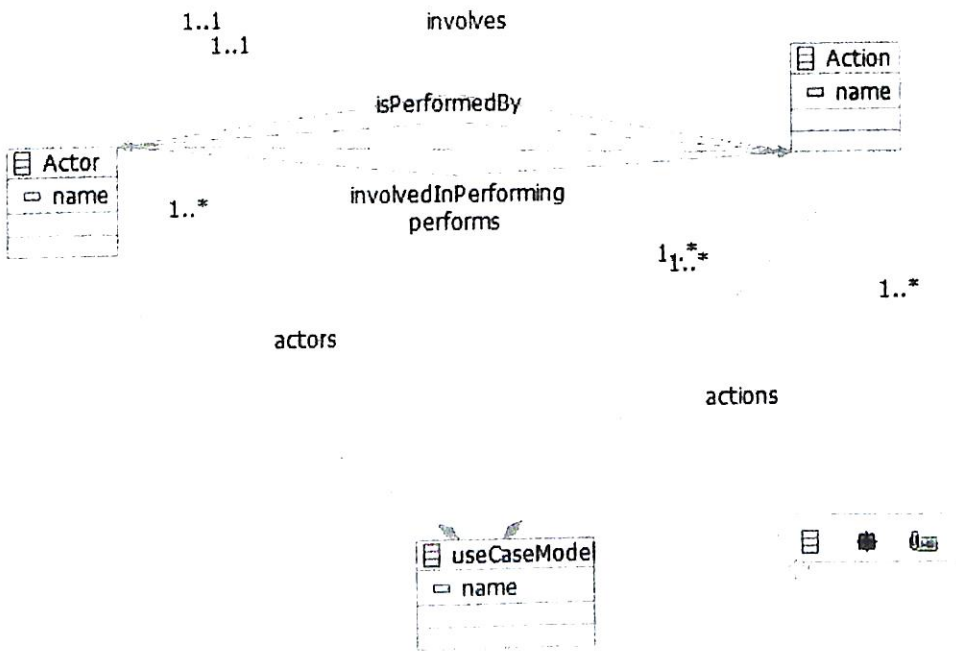
Use case is a sequence of some events that show how the system should interact with the user to achieve goals.

The main purpose of a use case diagram is to show what system functions are performed by each actor.

List of elements (Use case diagram)

1) Actor- Actors represent roles that users take on when they use the IT system, e.g., the role of a check-in employee.

2) Action



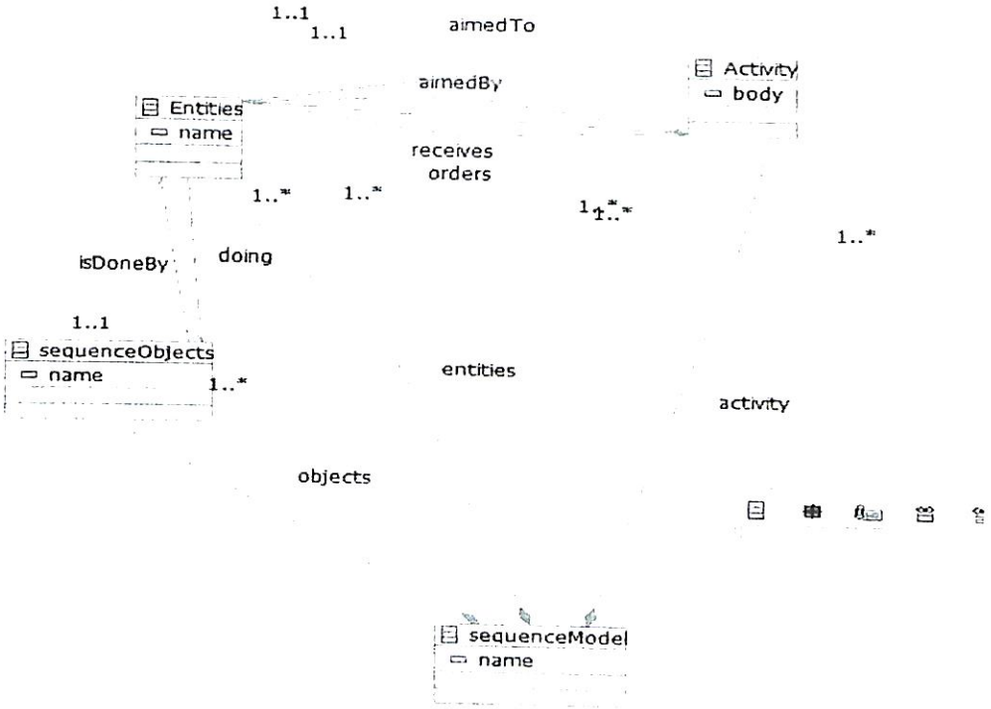
Metamodel: Sequence diagram

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another. It depicts the objects involved in the scenario and the sequence of activations with messages exchanged between the objects needed to carry out the functionality of the scenario. This allows the specification of simple runtime scenarios in a graphical manner.

In simpler worlds, it shows the sequence of events that take place during some user interaction with the system. It contains objects, activations and messages.

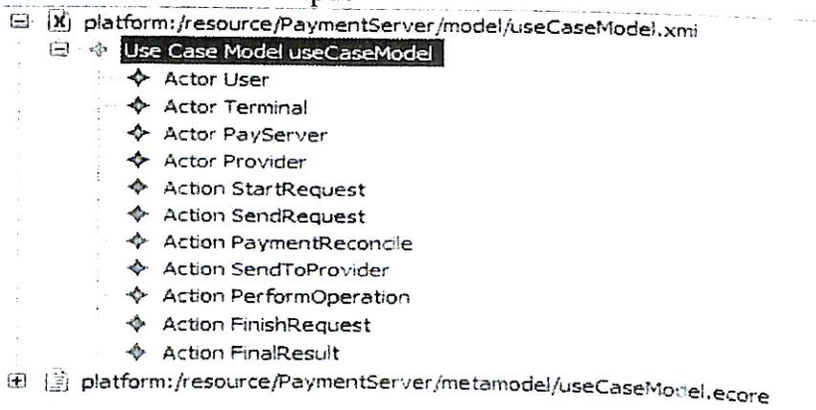
List of elements (Sequence diagram)

1. sequenceObject
2. Instances
3. Activity

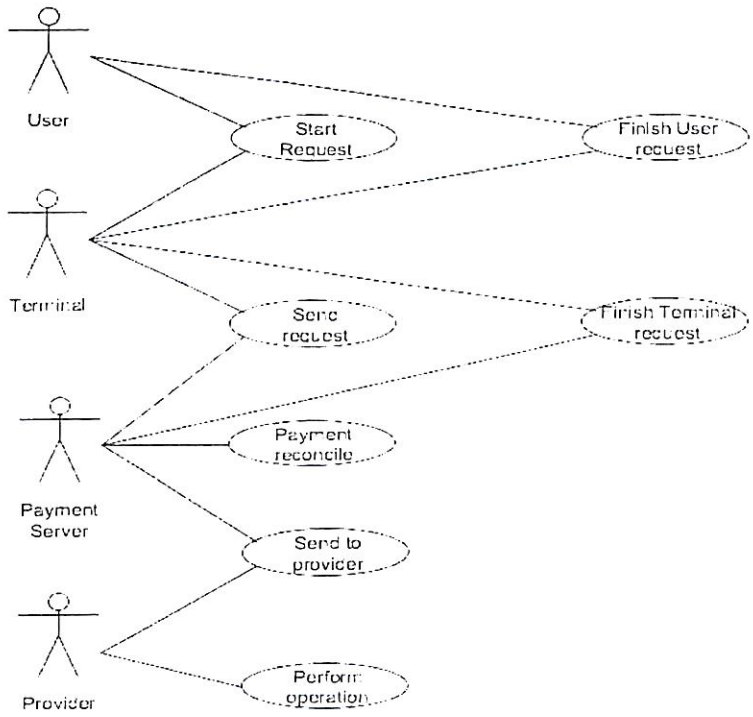


Input Model - Use-case Diagram

useCaseModel.xmi - output



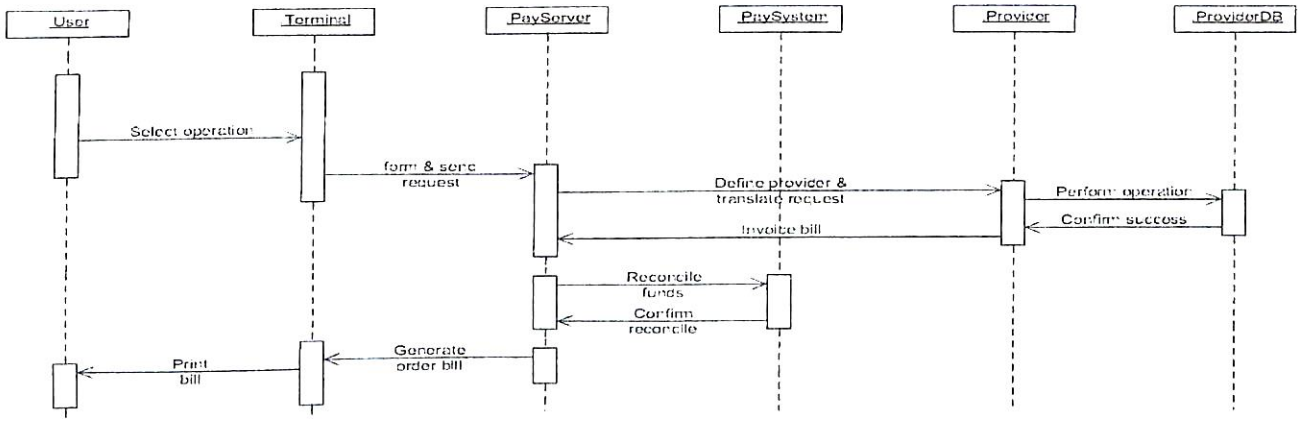
Graphical representation



In this diagram, User performs "Start request" action which involves terminal in this action. Then Terminal sends this request to Payment Sever in "send request action".

Furthermore, Payment server sends to provider this request which later is confirmed by provider and payments server reconciles funds from user account according to the operation fees.

Finally, Payment server reports a bill to terminal and user receives the print of this bill from the terminal.



3.2 Output Model - Sequence Diagram

In our sequence diagram we have 6 objects: User, Terminal, PayServer, PaySystem, Provider and ProviderDB.

Firstly, User initiates the process, and selects operation which forms a request inside terminal that includes all necessary data such as, provide, card holder name, bank account, service type, fees, etc. Then Terminal sends formed request to Pay server which deals the most significant role in overall system.

After receiving the request server reads the request, define provider, account and sends translated request to provider on especially dedicated payment channel to provider. At that time provider quickly reads all entire data deals with its own DB and performs the operations. All these doings must be paid and provider invoices the bill to the pay server.

According to this bill pay server reconciles the needed amount of money from users account from its payment system in the bank. After confirming successful reconcile pay server responses to terminal the bill.

Finally this bill is printed out to the client on the terminal printer.

5. Conclusion

Use cases helps to make easy comprehend about the functional requirements in the system and it easy to identify the various interactions between users and the systems within an environment.

Firstly we distinguished the PIM and PSM sides of the project where, PIM is a use case diagram which is more abstract, whereas PSM is sequence diagram that has more detailed description on functions.

In order to achieve the goal of transforming from PIM to PSM we used a KerMeta framework, that has its own language and tools in order perform such actions.

They clarify system requirements very categorically and systematically making it easier to understand the system and its interactions with the users.

We can conclude that model transformation is very effective method as, it allows document transformations on the fly, and therein facilitates the process of software development, since it is highly prone to errors, inconsistencies and misunderstandings and very often requires the change of system functionality.

Түйін

«Use case» диаграммасы жүйенің функцианалдық талаптарын оңайлыкпен түсінуге , сонымен қатар пайдаланушы мен жүйенің өзара әрекеттесуінің әр түрлерін айқындауға мүмкіндік береді. Диаграмма толығымен талаптарды айқындайды. Үзілді және жүйелі түрде түсіндіреді, сонымен қатар пайдаланушымен өзара әрекеттесу жүйесін жылдамдатып, жүйені түсінуді жеңілдетеді. Үлгіні түрлендіру өте тиімді тәсіл болып табылады, себебі ол құжаттарды бірден өзгертуге мүмкіндік береді, және бағдарламалық қамтамасыз ету барысын талдауды жеңілдетеді, себебі олар қате жіберуге, сәйкессіздікке және үйлесімсіздікке бейімді, және функциональдық жүйенің жиі өзгертуін талап етеді.

Резюме

«Use case» диаграммы позволяют с легкостью понять функциональные требования системы, а так же выявить различные виды взаимодействия между пользователем и системой.

Диаграмма полностью разъясняет требования. Категорично и систематически поясняет, а так же ускоряет систему взаимодействие с пользователями, что делает понимание системы проще.

Трансформация модели является очень эффективным методом, так как он позволяет документы преобразовывать на лету, и в нем облегчает процесс разработки программного обеспечения, так как они весьма склонны к ошибкам, несоответствиям и противоречиям и очень часто требует изменения функциональности системы.

Özet

«Use case» diyagramları kolay sistem işlevsel gereksinimlerini anlamak için, hem de kullanıcı ve sistem arasındaki etkileşimin çeşitli belirlemek mümkün kılmaktadır. Grafik tam gereksinimlerini açıklar Kategorik ve sistematik olarak açıklanmaktadır, hem de sistem anlaşılması kolay hale getirir kullanıcı, sistemi ile etkileşim hızlandırır. Bu sinek belgesine dönüştürmek için izin verir bu yana Dönüşüm modeli, çok etkili bir yöntem olduğunu ve bunların hataları, tutarsızlıkları ve çatışmalar çok eğilimli oldukları gibi, yazılım geliştirme sürecini kolaylaştıran ve genelde tüm sisteme bir değişiklik gerektirir