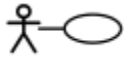


## Airport Check-In and Security Screening



### Airport check-in and security screening business model

**Purpose:** An example of a business use case diagram for airport check-in and security screening.

**Summary:** Business use cases are Individual Check-In, Group Check-In (for groups of tourists), Security Screening, etc. - representing business functions or processes taking place in an airport and serving needs of passengers.



### UML Use Case Diagram Example

This is an example of a **business use case** diagram which is usually created during **Business Modeling** and is rendered here in **Rational Unified Process**(RUP) notation.

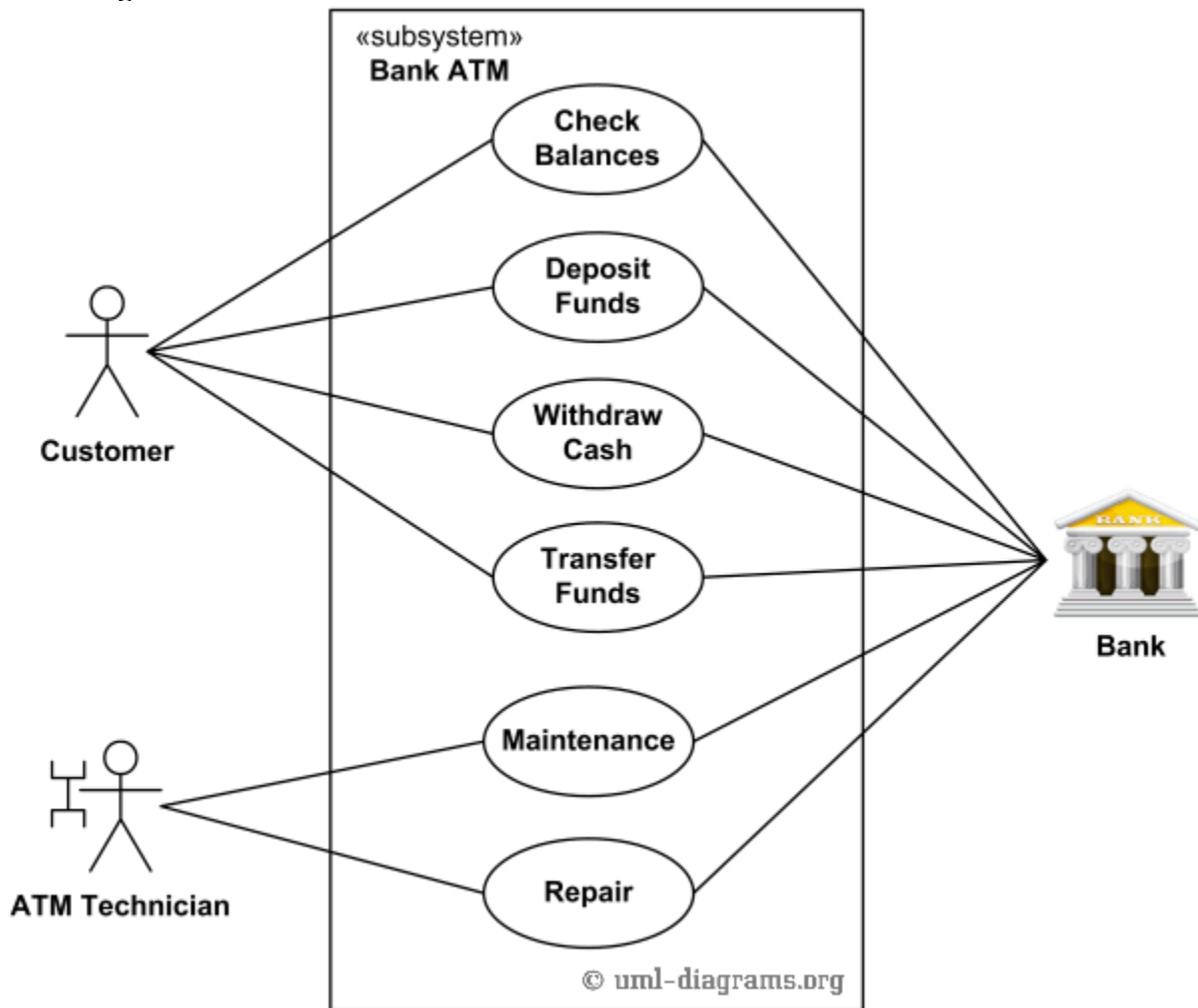
**Business actors** are Passenger, Tour Guide, Minor (Child), Passenger with Special Needs (e.g. with disabilities), all playing **external roles** in relation to airport business. **Business use cases** are Individual Check-In, Group Check-In (for groups of tourists), Security Screening, etc. - representing business functions or processes taking place in airport and serving the needs of passengers.

Business use cases Baggage Check-in and Baggage Handling extend Check-In use cases, because passenger might have no luggage, so baggage check-in and handling are optional.

## Bank ATM

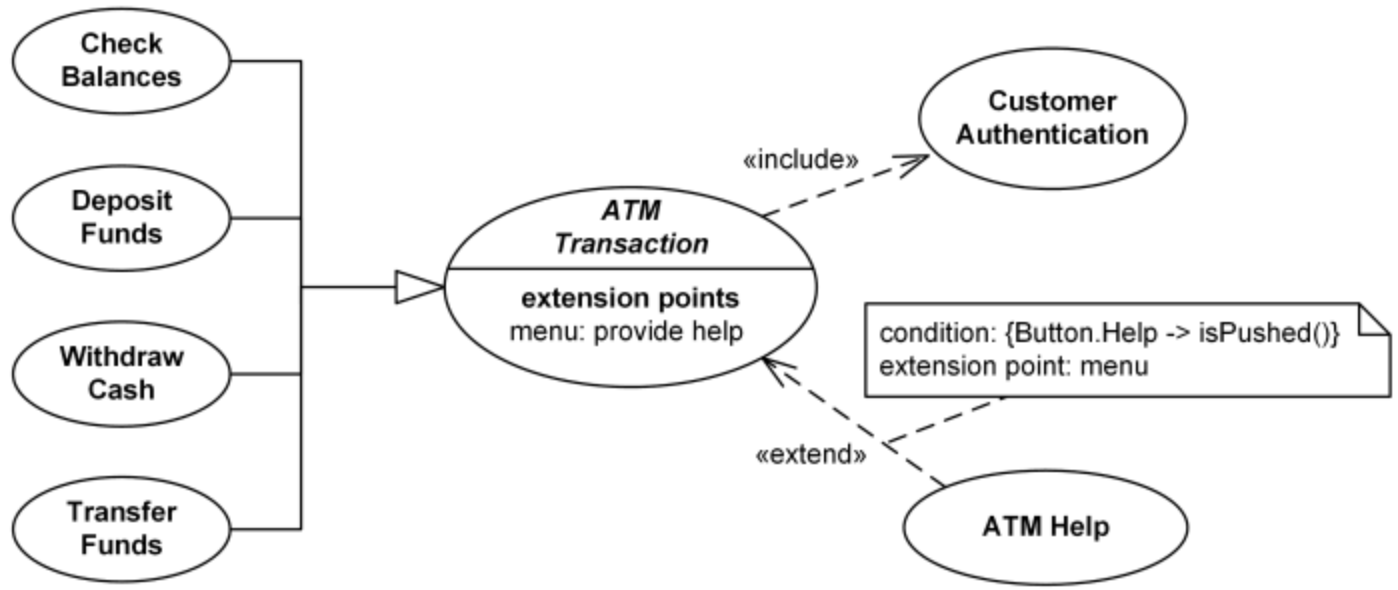
### UML Use Case Diagram Examples

An automated teller machine (ATM) or the automatic banking machine (ABM) is a banking subsystem (**subject**) that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk, or bank teller. *Customer* (**actor**) uses bank ATM to *Check Balances* of his/her bank accounts, *Deposit Funds*, *Withdraw Cash* and/or *Transfer Funds* (**use cases**). *ATM Technician* provides *Maintenance* and *Repairs*. All these use cases also involve *Bank* actor whether it is related to customer transactions or to the ATM servicing.



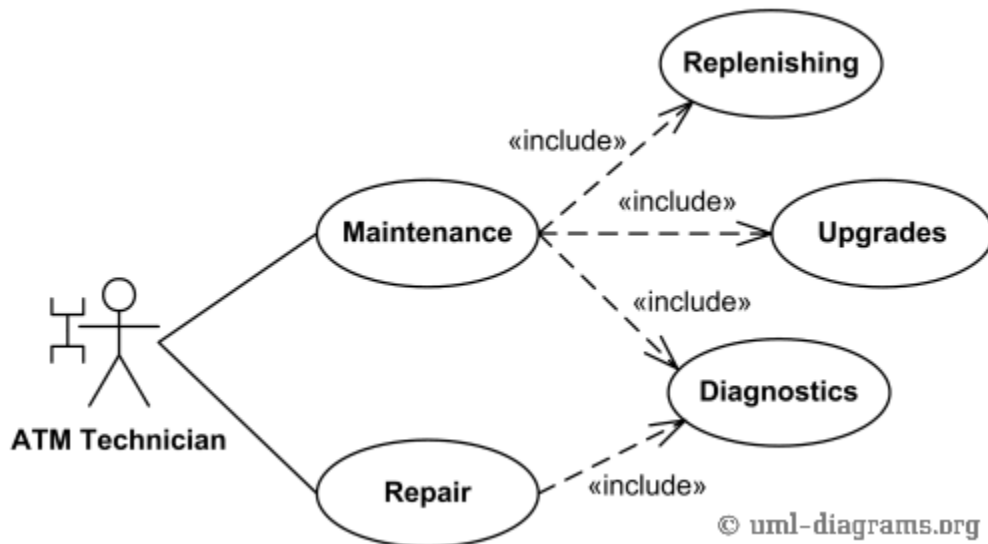
*An example of use case diagram for Bank ATM subsystem - top level use cases.*

On most bank ATMs, the customer is authenticated by inserting a plastic ATM card and entering a personal identification number (PIN). *Customer Authentication* use case is required for every ATM transaction so we show it as **include** relationship. Including this use case as well as transaction **generalizations** make the *ATM Transaction* an **abstract use case**.



### Bank ATM Transactions and Customer Authentication Use Cases Example.

Customer may need some help from the ATM. *ATM Transaction* use case is **extended** via **extension point** called *menu* by the *ATM Help* use case whenever *ATM Transaction* is at the location specified by the *menu* and the bank customer requests help, e.g. by selecting Help menu item.



### Bank ATM Maintenance, Repair, Diagnostics Use Cases Example.

*ATM Technician* maintains or repairs Bank ATM. *Maintenance* use case includes *Replenishing* ATM with cash, ink or printer paper, *Upgrades* of hardware, firmware or software, and remote or on-site *Diagnostics*. *Diagnostics* is also **included** in (shared with) *Repair* use case.

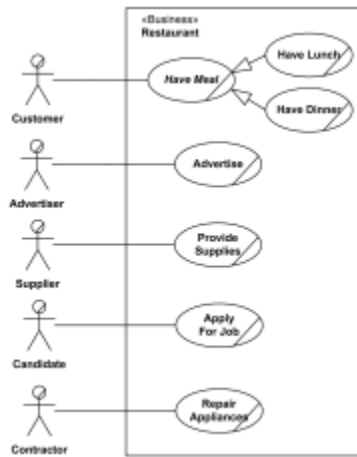
## Restaurant



### Restaurant business model

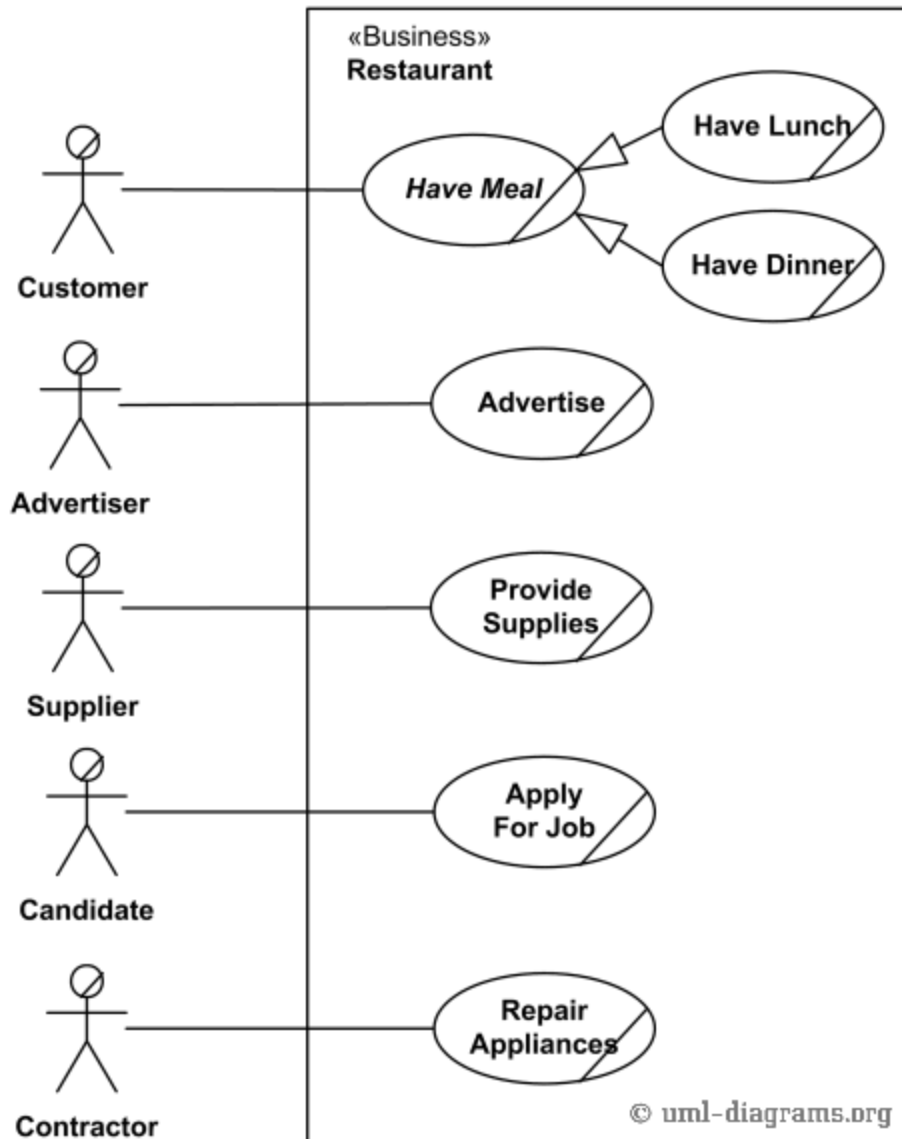
**Purpose:** Two alternative examples of business use case diagram for a Restaurant - external and internal business views of a restaurant.

**Summary:** Several business actors having some needs and goals as related to the restaurant and business use cases expressing expectations of the actors from the business.



### UML Use Case Diagram Example

Here we provide two alternative examples of a **business use case** diagram for a Restaurant, rendered in a notation used by **Rational Unified Process(RUP)**. First example shows **external business view** of a restaurant. We can see several **business actors** having some needs and goals as related to the restaurant and **business use cases** expressing expectations of the actors from the business.



### *Business use case diagram for Restaurant - External view*

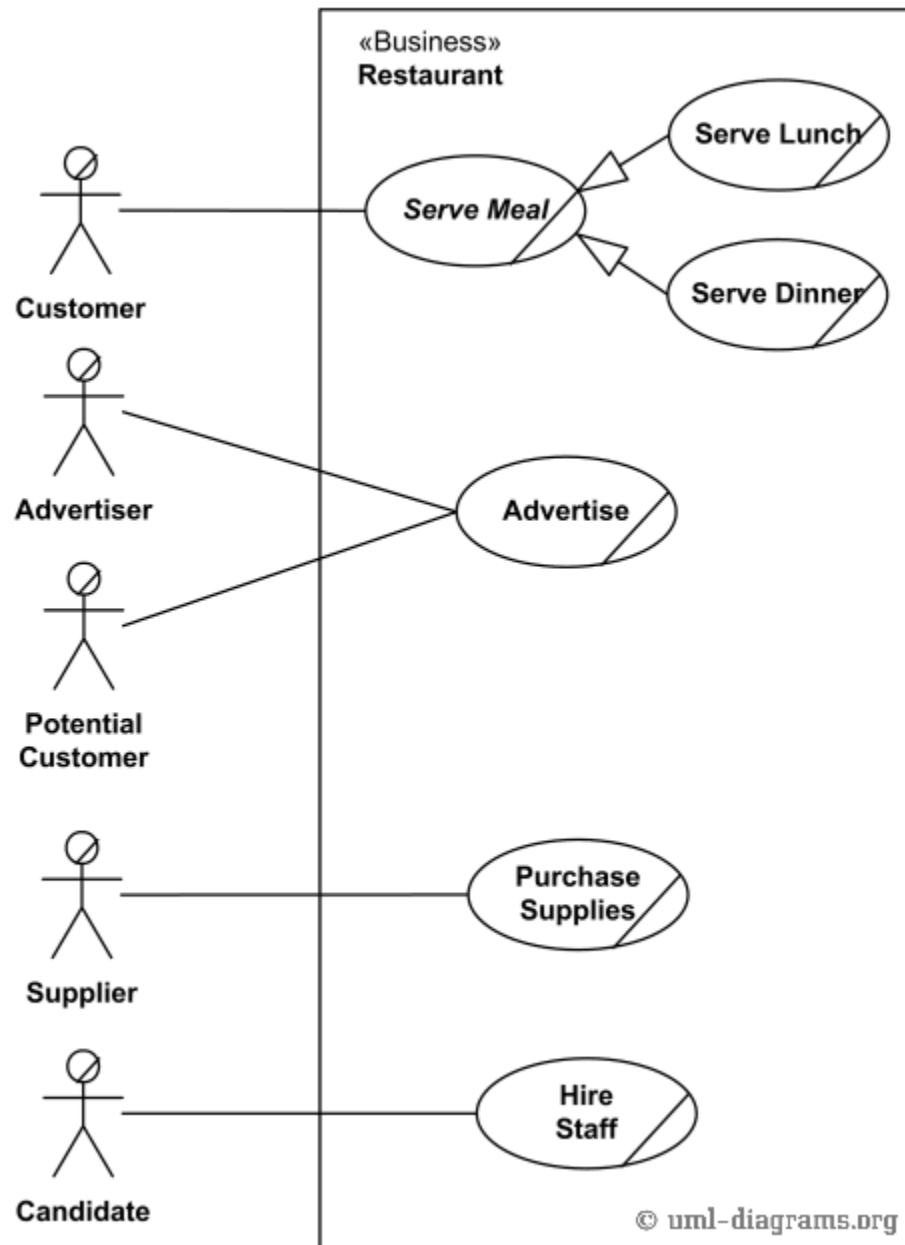
For example, **Customer** wants to **Have Meal**, **Candidate** - to **Apply for Job**, and **Contractor** - to fix some appliances. Note, that we don't have such actors as **Chef** or **Waiter**. They are not external roles but part of the business we model - the Restaurant, thus - they are not actors. In terms of RUP Chef and Waiter are **business workers**.

Second example shows **internal business view** of a restaurant. In this case we can see that restaurant has several business processes represented by **business use cases** which provide some services to external **business actors**. As in the previous example, actors have some needs and goals as related to the restaurant.

This approach could be more useful to model services that the business provides to different types of customers, but reading this kind of business use case diagrams could be confusing.

For example, **Customer** is now connected to **Serve Meal** use case, **Supplier** - to **Purchase Supplies**. We have now new actor **Potential Customer** participating

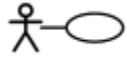
in **Advertise** use case by reading ads and getting some information about restaurant. At the same time, **Contractor** actor is gone because **Repair Appliances** is not a service usually provided by restaurants.



*Business use case diagram for Restaurant - Internal view*

Still, in this example we don't have actors as **Chef** or **Waiter** for the same reasons as before - they both are not external roles but part of the business we model.

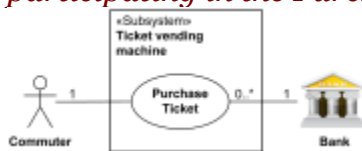
## Ticket Vending Machine



### Ticket vending machine

**Purpose:** Show that ticket vending machine allows commuters to buy tickets.

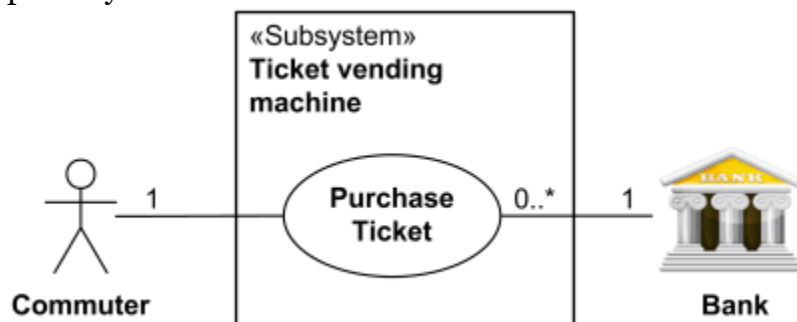
**Summary:** The ultimate goal of a Commuter in relation to our ticket vending machine is to buy a ticket. We have a single Purchase Ticket use case, as this vending machine is not providing any other services. Ticket vending machine is a **subject** of the example use case diagram. Commuter and Bank are our **actors**, both participating in the Purchase Ticket **use case**.



### UML Use Case Diagram Example

**Ticket vending machine**, i.e. vending machine that sells and produces tickets to commuters, is a **subject** of the example use case diagram. This kind of a machine is a combination of both hardware and software, and it is only a part of the whole system selling tickets to the customers. So we will use **«Subsystem»** stereotype.

Ticket vending machine allows commuters to buy tickets. So **Commuter** is our primary **actor**.

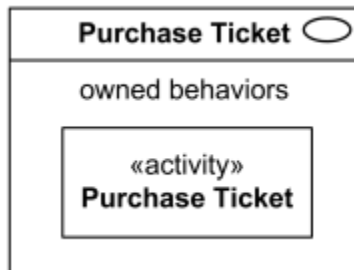


*Ticket vending machine provides Purchase Ticket use case for the Commuter and Bank actors.*

The ultimate goal of the Commuter in relation to our ticket vending machine is to buy a ticket. So we have **Purchase Ticket use case**. Purchasing ticket might involve a bank, if payment is to be made using a debit or credit card. So we are also adding another actor - **Bank**. Both actors participating in the use case are connected to the use case by association.

Use case **behaviors** may be described in a natural language text (opaque behavior), which is current common practice, or by using UML **behavior diagrams**. UML tools

should allow binding behaviors to the described use cases. Example of such binding of the Purchase Ticket use case to the behavior represented by activity is shown below using UML 2.5 notation.



*Purchase Ticket use case owns behavior represented by Purchase Ticket activity.*



[Ticket vending machine activity diagram](#)

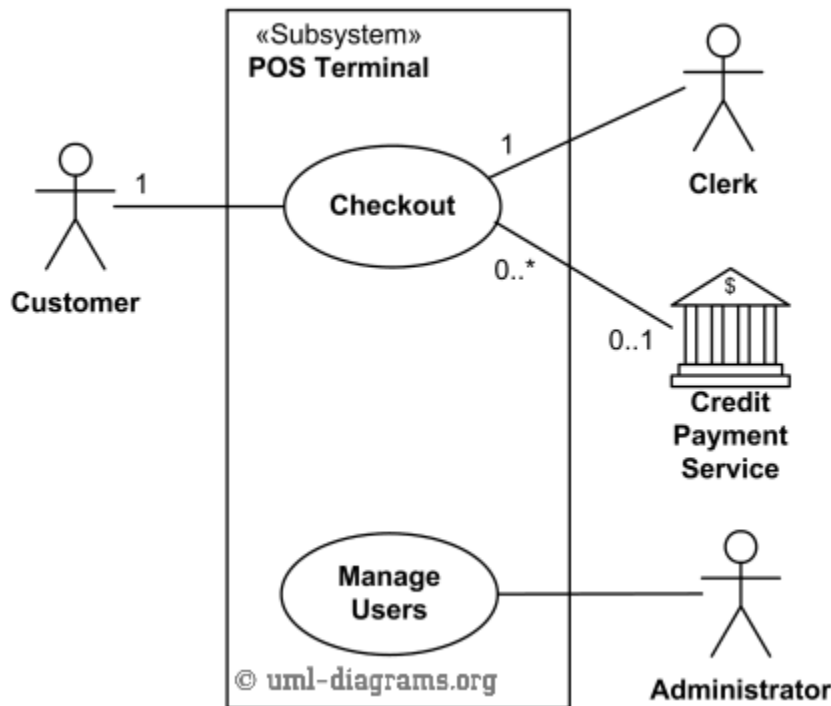


## Point of Sales Terminal

### UML Use Case Diagram Example

An example of UML **use case diagram** for **Point of Sale (POS) Terminal** or Checkout. A retail POS system typically includes a computer, monitor, keyboard, barcode scanners, weight scale, receipt printer, credit card processing system, etc. and POS terminal software.

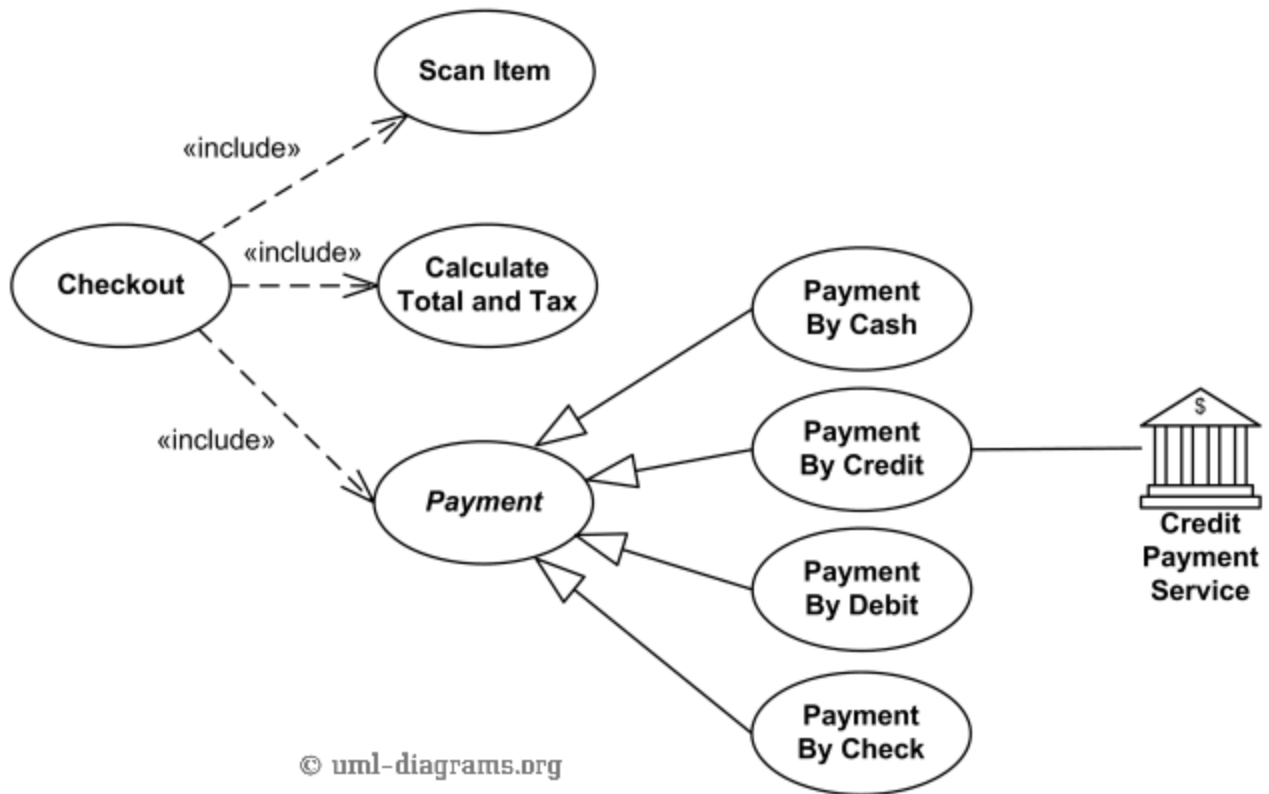
Checkout **use case** involves Customer, Clerk and Credit Payment Service **actors** and **includes** scanning items, calculating total and taxes, payment use cases.



#### *Top level UML use cases for Point of Sales Terminal (POS).*

Checkout use case requires Customer actor, hence the 1 multiplicity of Customer. Clerk can only participate in a single Checkout use case. Credit Payment Service can participate with many Checkout use cases at the same time. Checkout use case may not need Credit Payment Service (for example, if payment is in cash), thus the 0..1 multiplicity.

Checkout use case is an example of a large and complex use case split into several use cases each describing some logical unit of behavior. Note, that including use case becomes incomplete by itself and requires the included use cases to be complete.



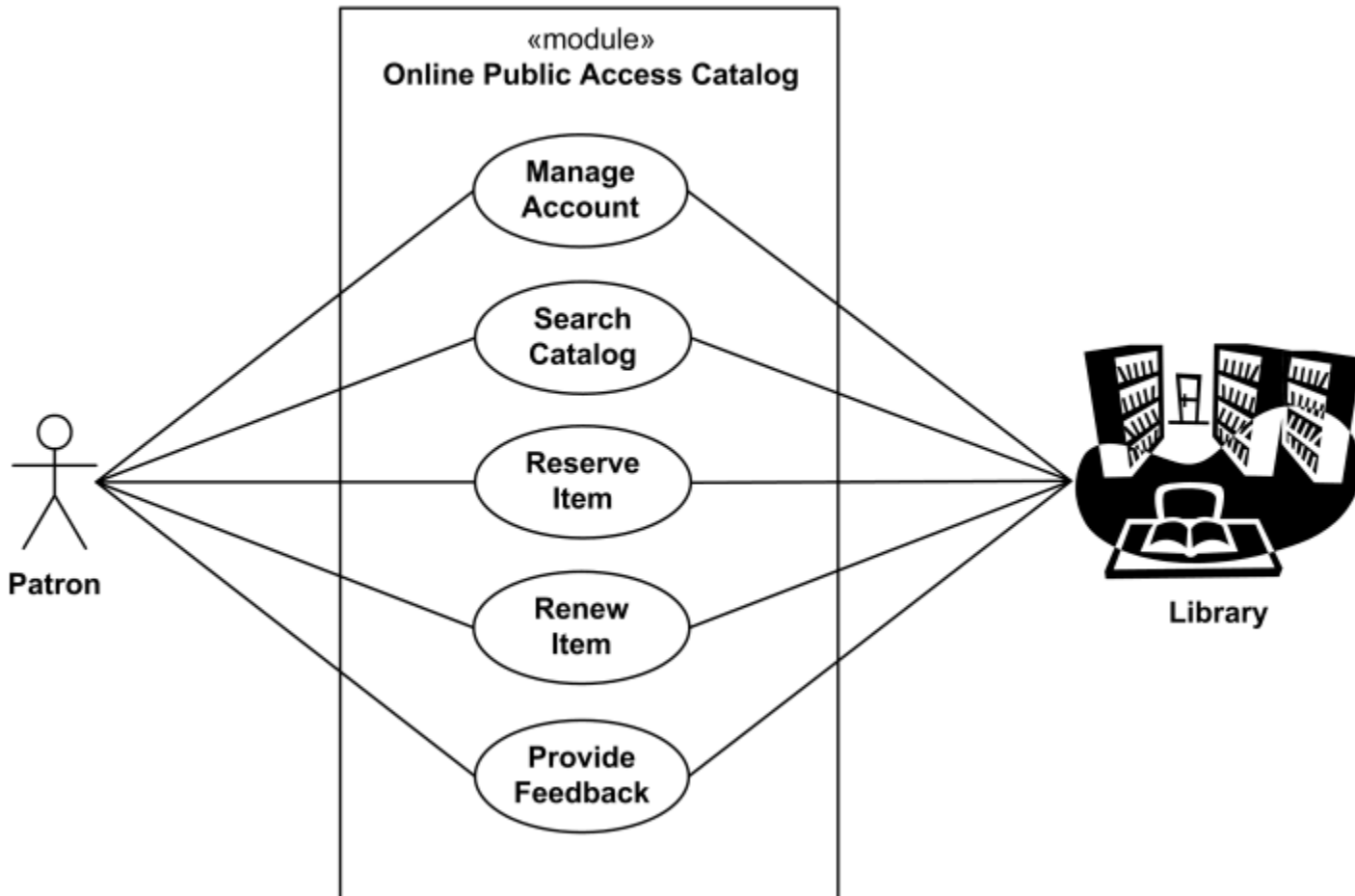
*Checkout use case includes Scan Item, Calculate Total and Tax, and Payment use cases.*

Payment use case is represented using **generalization** relationship. It means that only one specific type of payment is accepted - either by cash, or by credit, debit, or with check. An alternative to such representation could be to use **include** relationship so that not just single but several forms of payment could be accepted from the same client during checkout.

## Online Library Public Access Catalog

### *UML Use Case Diagram Examples*

An **Online Public Access Catalog (OPAC)** is an e-Library website which is part of **Integrated Library System (ILS)**, also known as a **Library Management System (LMS)**, and managed by a library or group of libraries. Patrons of the library can search library catalog online to locate various resources - books, periodicals, audio and visual materials, or other items under control of the library. Patrons may reserve or renew item, provide feedback, and manage their account.

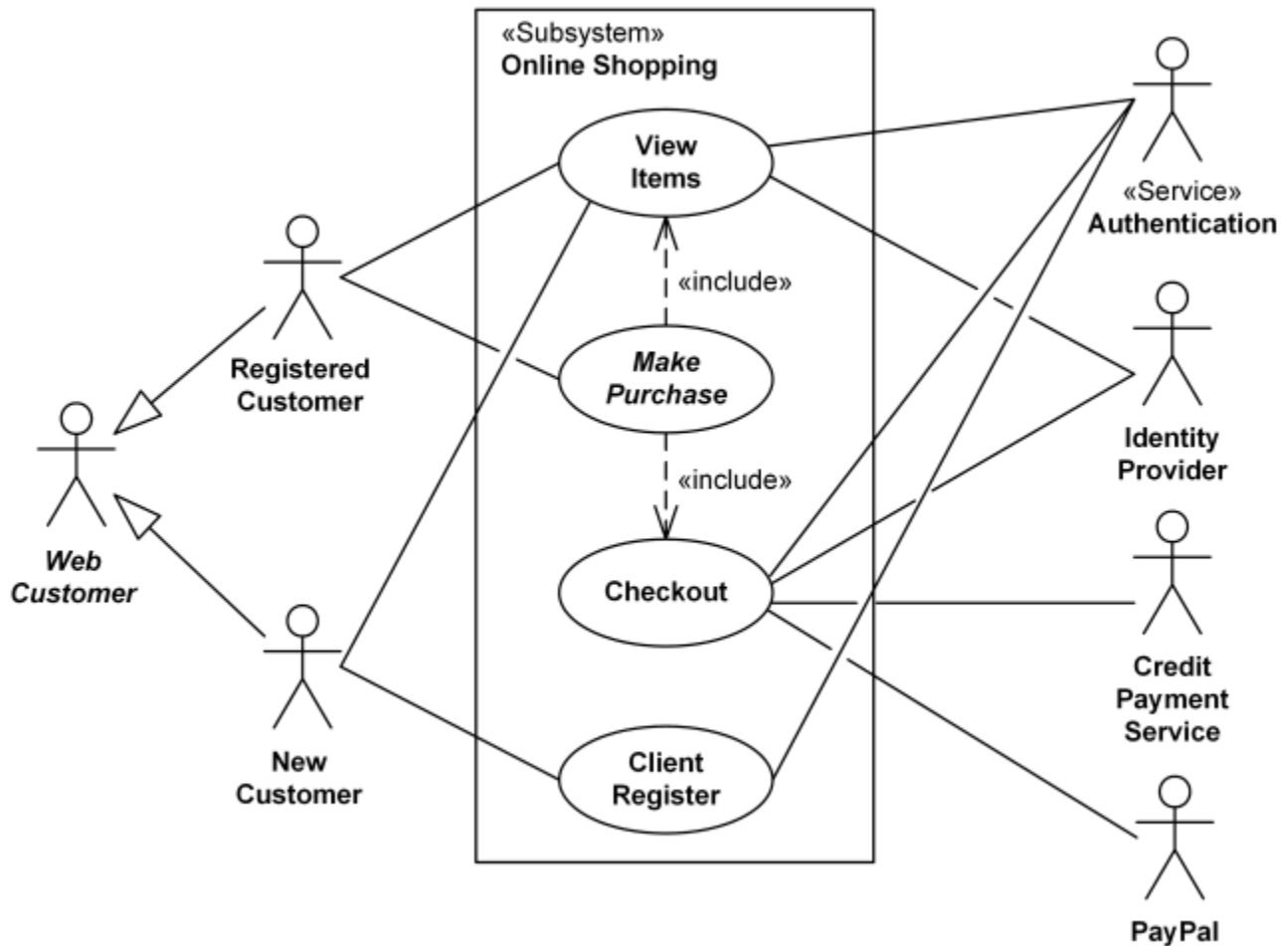


*An example of UML use case diagram for e-Library Online Public Access Catalog.*

## Online Shopping

### UML Use Case Diagram Example

**Web Customer actor** uses some web site to make purchases online. Top level **use cases** are **View Items**, **Make Purchase** and **Client Register**. View Items use case could be used by customer as top level use case if customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customer to register on the web site, for example to get some coupons or be invited to private sales. Note, that **Checkout** use case is **included use case** not available by itself - checkout is part of making purchase. Except for the **Web Customer** actor there are several other actors which will be described below with detailed use cases.



*Online shopping UML use case diagram example - top level use cases.*

**View Items** use case is **extended** by several optional use cases - customer may search for items, browse catalog, view items recommended for him/her, add items to shopping cart or wish list. All these use cases are extending use cases because they provide some optional functions allowing customer to find item.

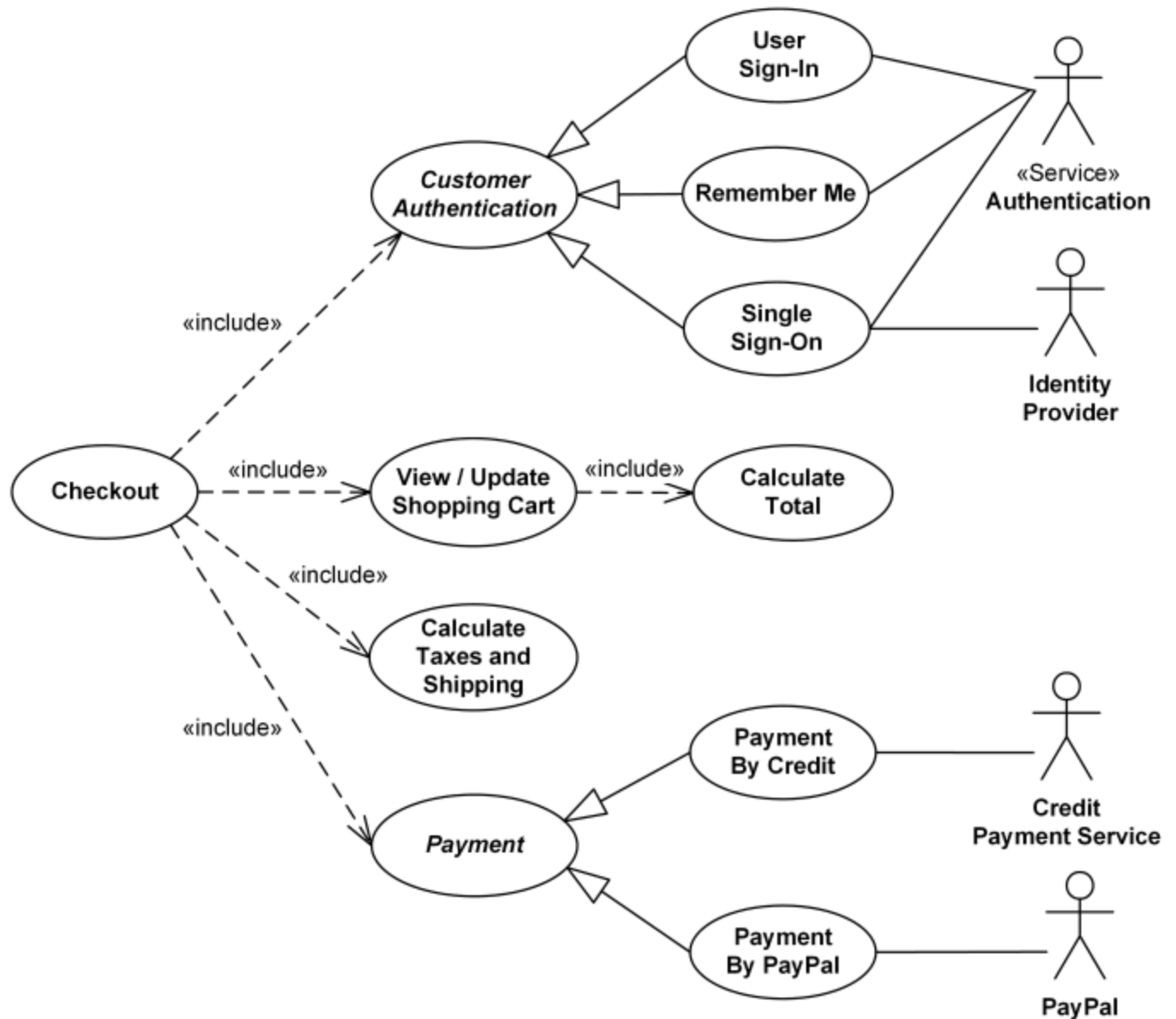
**Customer Authentication** use case is **included** in **View Recommended Items** and **Add to Wish List** because both require the customer to be authenticated. At the same time, item could be added to the shopping cart without user authentication.



*Online shopping UML use case diagram example - view items use case.*

**Checkout** use case includes several required uses cases. Web customer should be authenticated. It could be done through user login page, user authentication cookie ("Remember me") or Single Sign-On (SSO). Web site authentication service is used in all these use cases, while SSO also requires participation of external identity provider.

**Checkout** use case also includes **Payment** use case which could be done either by using credit card and external credit payment service or with PayPal.



*Online shopping UML use case diagram example - checkout, authentication and payment use cases.*

## Online Shopping - Credit Cards Processing

### *UML Use Case Diagram Example*

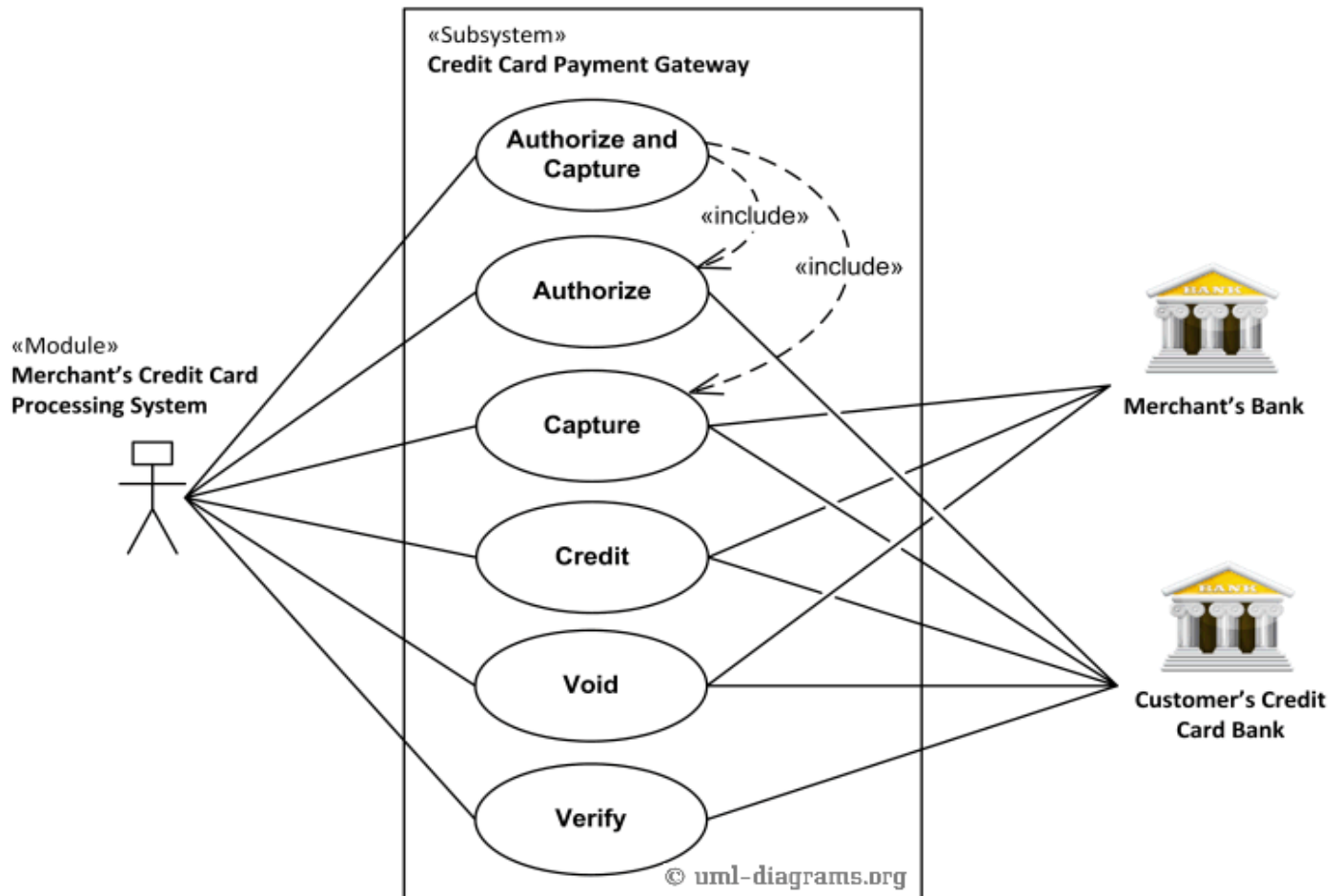
This UML use case diagram example shows some use cases for a system which processes credit cards.

**Credit Card Processing System** (aka Credit Card Payment Gateway) is a **subject**, i.e. system under design or consideration. Primary **actor** for the system is a **Merchant's Credit Card Processing System**. The merchant submits some credit card transaction request to the credit card payment gateway on behalf of a customer. Bank which issued customer's credit card is actor which could approve or reject the transaction. If transaction is approved, funds will be transferred to merchant's bank account.

**Authorize and Capture** use case is the most common type of credit card transaction. The requested amount of money should be first authorized by **Customer's Credit Card Bank**, and if approved, is further submitted for settlement. During the settlement funds approved for the credit card transaction are deposited into the **Merchant's Bank** account.

In some cases, only **authorization** is requested and the transaction will not be sent for settlement. In this case, usually if no further action is taken within some number of days, the authorization expires. Merchants can submit this request if they want to verify the availability of funds on the customer's credit card, if item is not currently in stock, or if merchant wants to review orders before shipping.

**Capture** (request to capture funds that were previously authorized) use case describes several scenarios when merchant needs to complete some previously authorized transaction - either submitted through the payment gateway or requested without using the system, e.g. using voice authorization.



*UML use case diagram example for a credit cards processing system.*

**Credit** use case describes situations when customer should receive a refund for a transaction that was either successfully processed and settled through the system or for some transaction that was not originally submitted through the payment gateway.

**Void** use case describes cases when it is needed to cancel one or several related transactions that were not yet settled. If possible, the transactions will not be sent for settlement. If the Void transaction fails, the original transaction is likely already settled.

**Verify** use case describes zero or small amount verification transactions which could also include verification of some client's data such as address.

You can find excellent resources, documentation, white papers, guides, etc. related to the credit card processing at [Authorize.Net - Payment Gateway to Accept Online Payments](#).



### *UML Use Case Diagram Example*

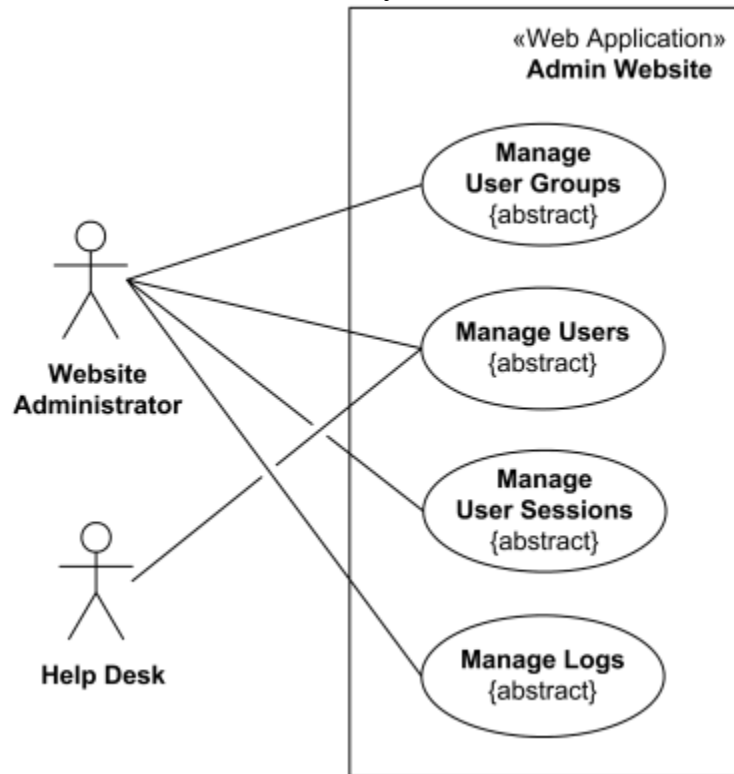
Website security requirements mandate separation of administrative interfaces from common functions provided to users. This segregation, for example, is required by the Sarbanes Oxley (SOX) in US and is strongly recommended by ISO 17799.

System should have separate applications for administrators and for common users. It is recommended by **OWASP Guide 2.0** that website administration applications should not be accessible from the internet without going through some management networks, e.g. via a strongly authenticated VPN or from a trusted network operations center.

Except for administrators, some part of the administrative interfaces should be also available to the Help desk staff, as they need to be able to assist customers having issues while using the customer oriented website.

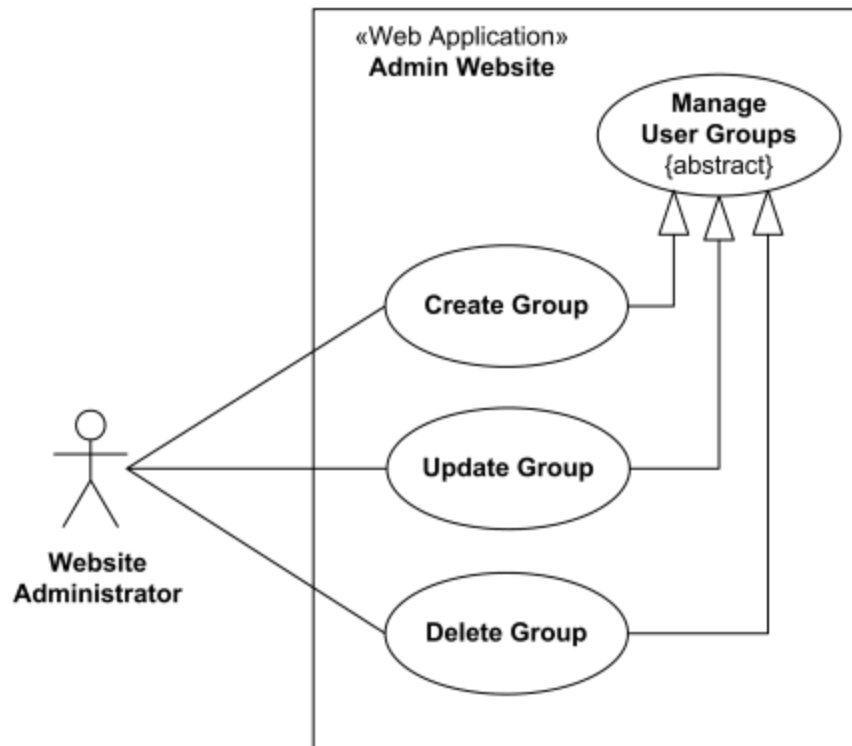
Top level use case diagram below shows some administrative functions that administration website could provide.

Two **actors** using administrative interfaces are **Website Administrator** and **Help Desk**. Help Desk uses a subset of functions available to the Website Administrator. All top level **use cases** shown are abstract as each represents some group or "package" of administrative functionality.



*Top level use case diagram for the administration website.*

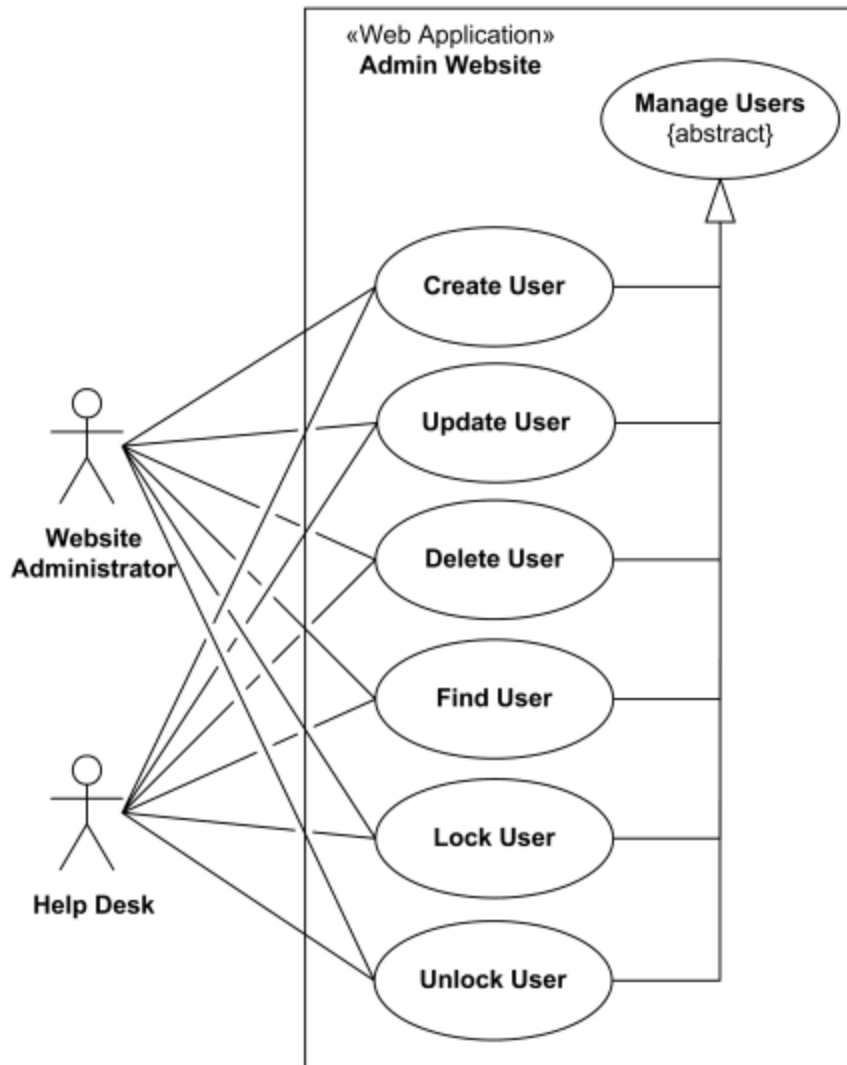
**Manage User Groups** abstract use case is specialized by **Create Group**, **Update Group**, and **Delete Group** use cases. The idea is that website administrator could create different user groups, for example, having different privileges or options, and later some user groups could be modified or even deleted.



*User group management use case diagram for the administration website.*

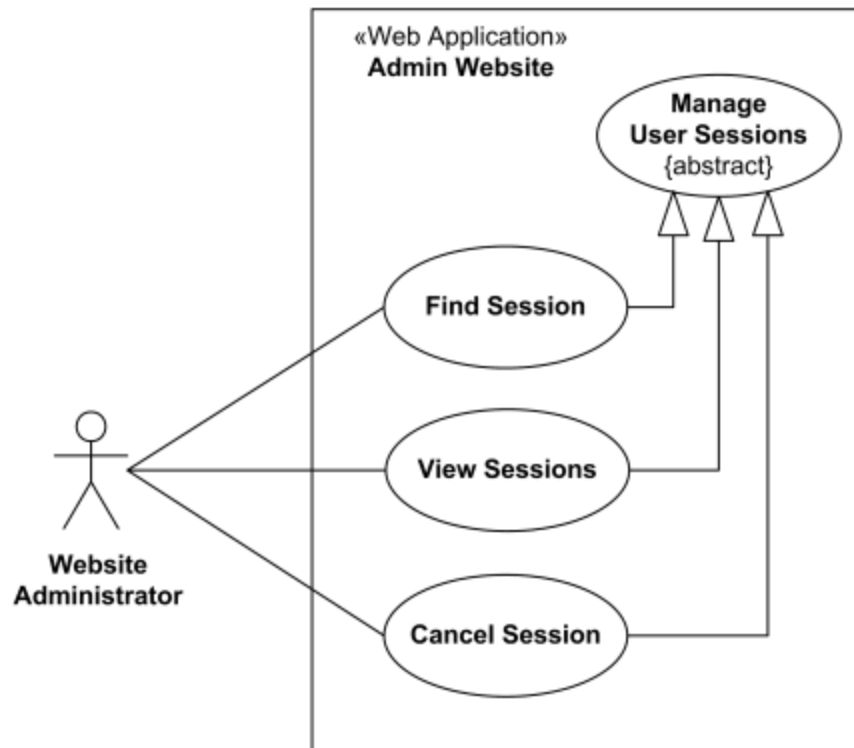
User management use cases are available both to the **Website Administrator** and to the **Help Desk**. There is standard user CRUD (Create, Retrieve/Find, Update, Delete) functionality set.

Two other use cases, **Lock User** and **Unlock User** are specific to website security. For example, if during some predefined period of time there were several unsuccessful login attempts using wrong user password, user account should be locked for some predefined time to prevent possible brute force password guessing attack. This locking and unlocking is usually done automatically by intrusion detection or website authentication subsystem, but this functionality needs to be available in the manual mode too, just in case. For example, some user might call and ask to lock his or her account.



*User management use case diagram for the administration website.*

User session is created either for each new incoming request that is not yet part of a session, or/and after user was authenticated. Website administrator should have ability to see how many sessions were created, including some statistics about sessions, to find some specific session and see status of that session, and to cancel (delete) some session, if required.

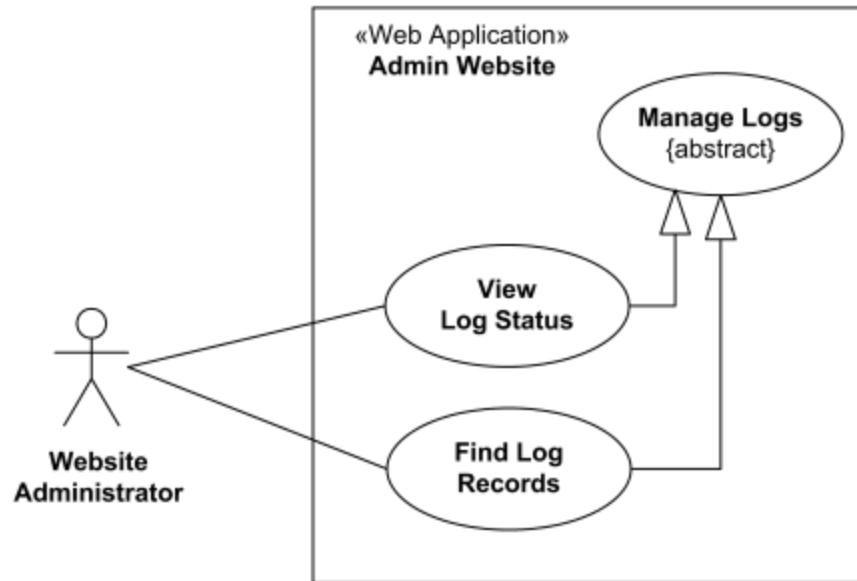


*User sessions management use case diagram for the administration website.*

List of administrative functions included in the **log management** depend on the security requirements supported and implemented by the website.

It is a standard security requirement (e.g., see [OWASP Guide 2.0](#)) for the logs that new records can be only appended while older log records should not be rewritten or deleted. It could be implemented e.g. by writing logs to a write once / read many (WORM) device such as a CD-R.

Website administrator should be able to see status of logs. The status could include verification that logging is still functional (there is enough space on disk and/or connection to database is not stale), and that older log files are on schedule being moved to a permanent storage for archiving.



*Logs management use case diagram for the administration website.*

It is also common requirement to allow website administrator to find and see some log records related to a specific user or an exceptional situation.

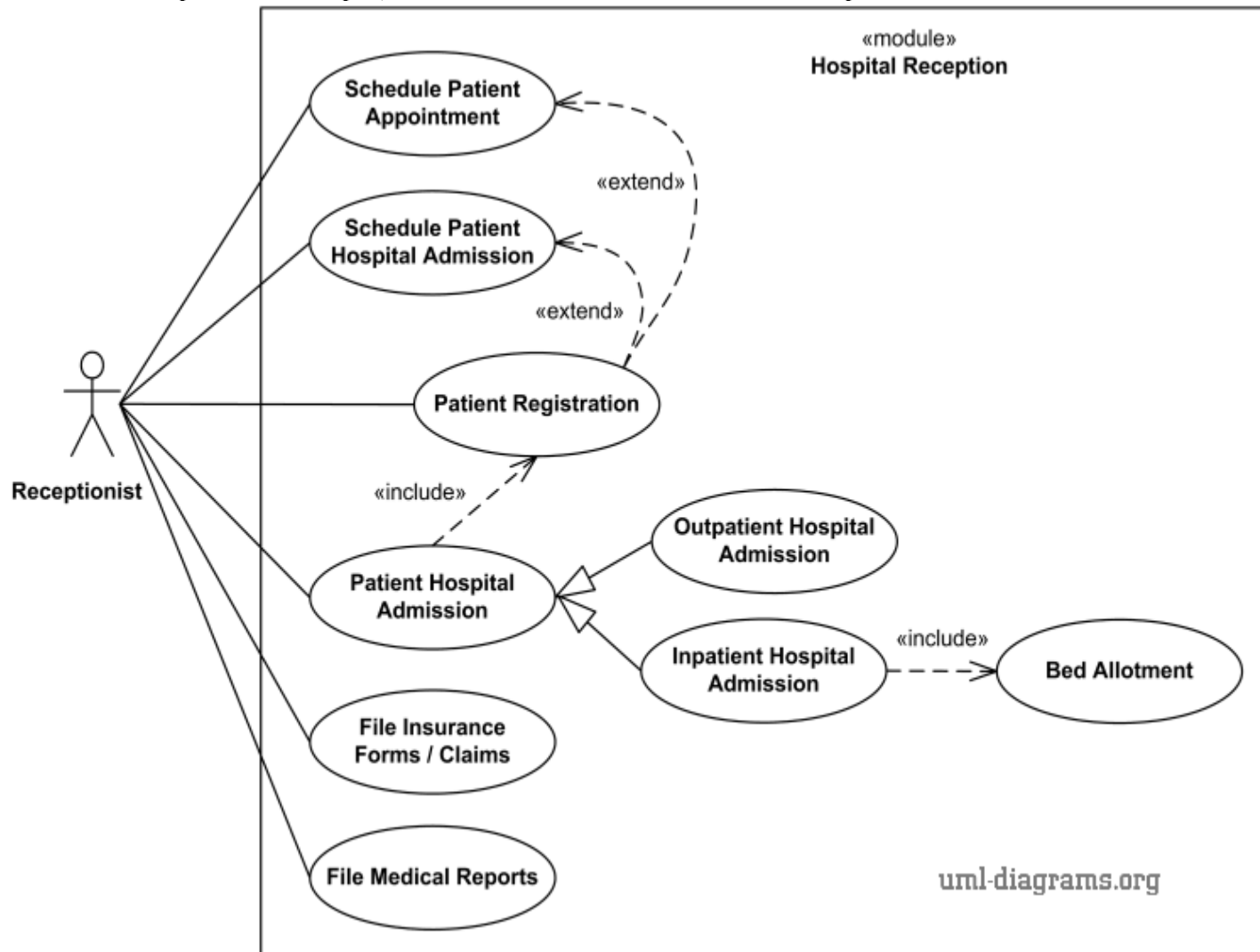
## Hospital Management

### UML Use Case Diagram Example

**Hospital Management System** is a large system including several subsystems or modules providing variety of functions. UML use case diagram example below shows actor and use cases for a hospital's reception.

**Purpose:** *Describe major services (functionality) provided by a hospital's reception.*

**Hospital Reception** subsystem or module supports some of the many job duties of hospital receptionist. Receptionist schedules patient's appointments and admission to the hospital, collects information from patient upon patient's arrival and/or by phone. For the patient that will stay in the hospital ("inpatient") she or he should have a bed allotted in a ward. Receptionists might also receive patient's payments, record them in a database and provide receipts, file insurance claims and medical reports.



*An example of use case diagram for Hospital Reception.*

## **Radiology Diagnostic Reporting**

### ***UML Use Case Diagram Example***

**Integrating the Healthcare Enterprise (IHE)** is an initiative promoting the use of standards to achieve interoperability of **Health Information Technology (HIT)** systems and effective use of **Electronic Health Records (EHRs)**.

IHE publishes the implementation guides called **IHE Profiles** incorporated in the appropriate **IHE Technical Framework** after successful testing and deployment in the real-world healthcare settings.

The **Simple Image and Numeric Report (SINR)** [[IHE Radiology Integration Profile, IHE RAD TF Vol. 1, Rev. 11.0](#)] facilitates the growing use of digital dictation, voice recognition, and specialized reporting packages, by separating the functions of diagnostic reporting into discrete actors for creation, management, storage and report viewing. Separating these functions while defining transactions to exchange the reports between them enables a vendor to include one or more of these functions in an actual system.

The **IHE Technical Framework (TF)** identifies **IHE Actors** - functional components of a healthcare enterprise from the point of view of their interactions in distributed healthcare environment. On the example diagram below we show **IHE Actors** as **UML actors** notated as **classifiers** marked with radiology icon.

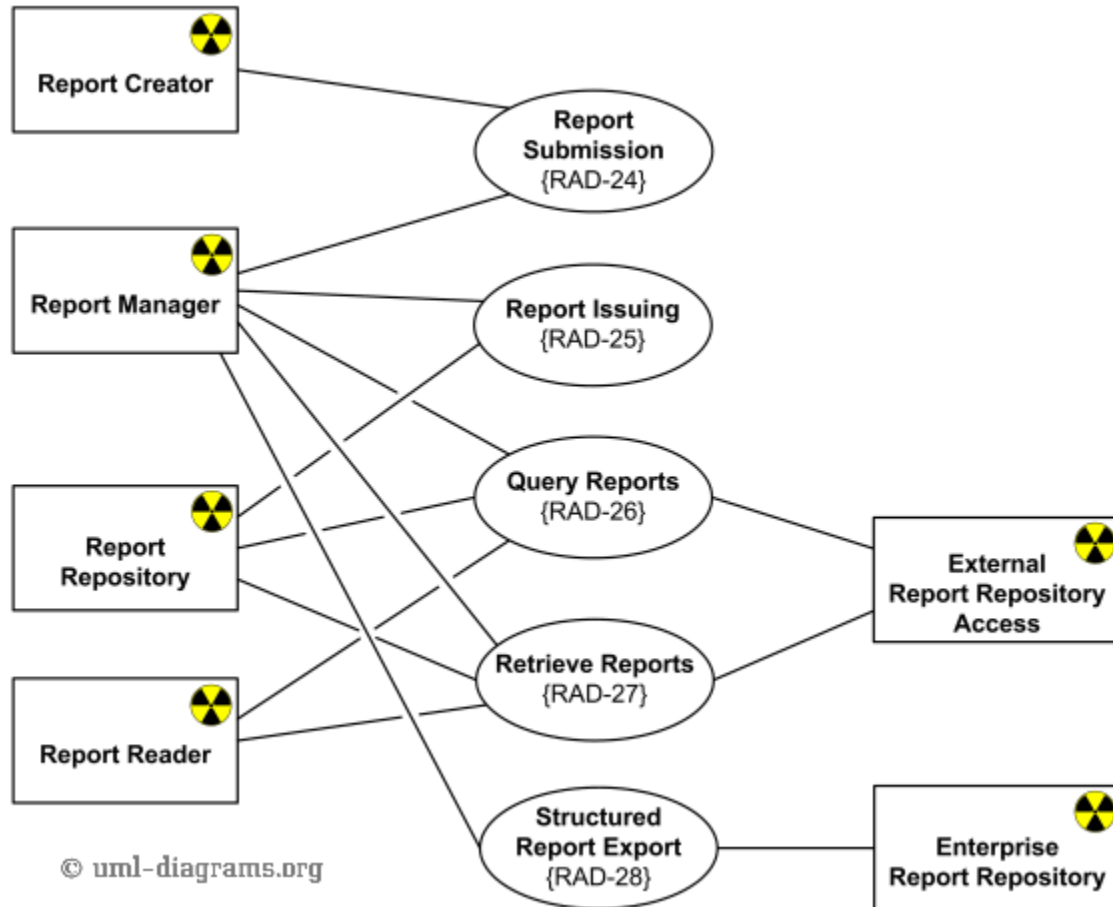
IHE TF also specifies a set of coordinated, standards-based interactions between IHE Actors, called **IHE Transactions**. Transactions transfer required information through standards-based messages. On the example diagram below we show **IHE Transactions** as **UML use cases**.

**IHE Diagnostic Report Process Flow** describes the typical process flow related to the diagnostic reporting. The IHE transactions of this flow are RAD-24 through RAD-27.

In the initial stage of diagnostic reporting, a **reading physician** records a diagnosis by generating a draft **DICOM Structured Report (SR)** object. In the **Report Submission** transaction, **Report Creator** actor transmits that DICOM SR object in an initial draft or final state to the **Report Manager**. The Structured Report object is required minimally to conform to the template TID 2000. DICOM Structured Reports offer the capability to encode arbitrarily structured diagnostic report data.

The **Simple Image Report** allows documents with multiple sections (with headings) containing report text and references to relevant images. Some text items of these documents may also be related to specific images. This allows a reading physician to identify one or more images from which their conclusions were inferred. This kind of reports (aka "content pattern") should use the DICOM Basic Text SR Information Object Definition (IOD) and Basic Image Diagnostic Report Template (TID 2000 in DICOM 2011 PS3.16).

The **Simple Image and Numeric Report** is similar to the Simple Image Report but allows the addition of numeric values. This enables a diagnosis to include measurements and other numeric values. This kind of reports should use the DICOM Enhanced SR IOD and Basic Image Diagnostic Report Template (TID 2000 in DICOM 2011 PS3.16).



*Radiology diagnostic reporting UML use case diagram example for Simple Image and Numeric Report (SINR) IHE Radiology Integration Profile.*

Reports are processed and modified by the **Report Manager** IHE actor. This involves adding and changing report data as well as verifying draft reports. In all cases, any change in the report content by the Report Manager leads to the creation of a new DICOM SR object. At any time, the Report Manager can transmit reports to the **Report Repository** for external access, but at a minimum the final report must be sent to the Report Repository.

In the **Report Issuing** transaction, the Report Manager transmits either an unchanged draft DICOM SR or a new modified DICOM SR to the Report Repository. The Report Manager handles all state and content changes to DICOM Structured Reports, and with each change new DICOM Structured Report objects are created and may be stored in the Report Repository.

The **Report Repository** provides permanent storage of DICOM Structured Reports. It also allows reports to be queried and retrieved throughout the enterprise by **Report Readers**.



The **Report Reader** provides a user interface to view DICOM Structured Reports. DICOM SRs are queried and retrieved by the Report Reader from the **Report Repository** or the **External Report Repository Access**.

The **External Report Repository Access** actor is a gateway to obtain other enterprise department reports, such as Laboratory and Pathology, from within the Imaging department.

In the Structured Report Export [RAD-28] transaction, the Report Manager transmits verified Structured Reports as unsolicited HL7 observations to the **Enterprise Report Repository**. The Enterprise Report Repository receives diagnostic reports in HL7 format. The Report Manager is responsible for mapping DICOM SR to HL7.

## **Software Protection and Licensing**

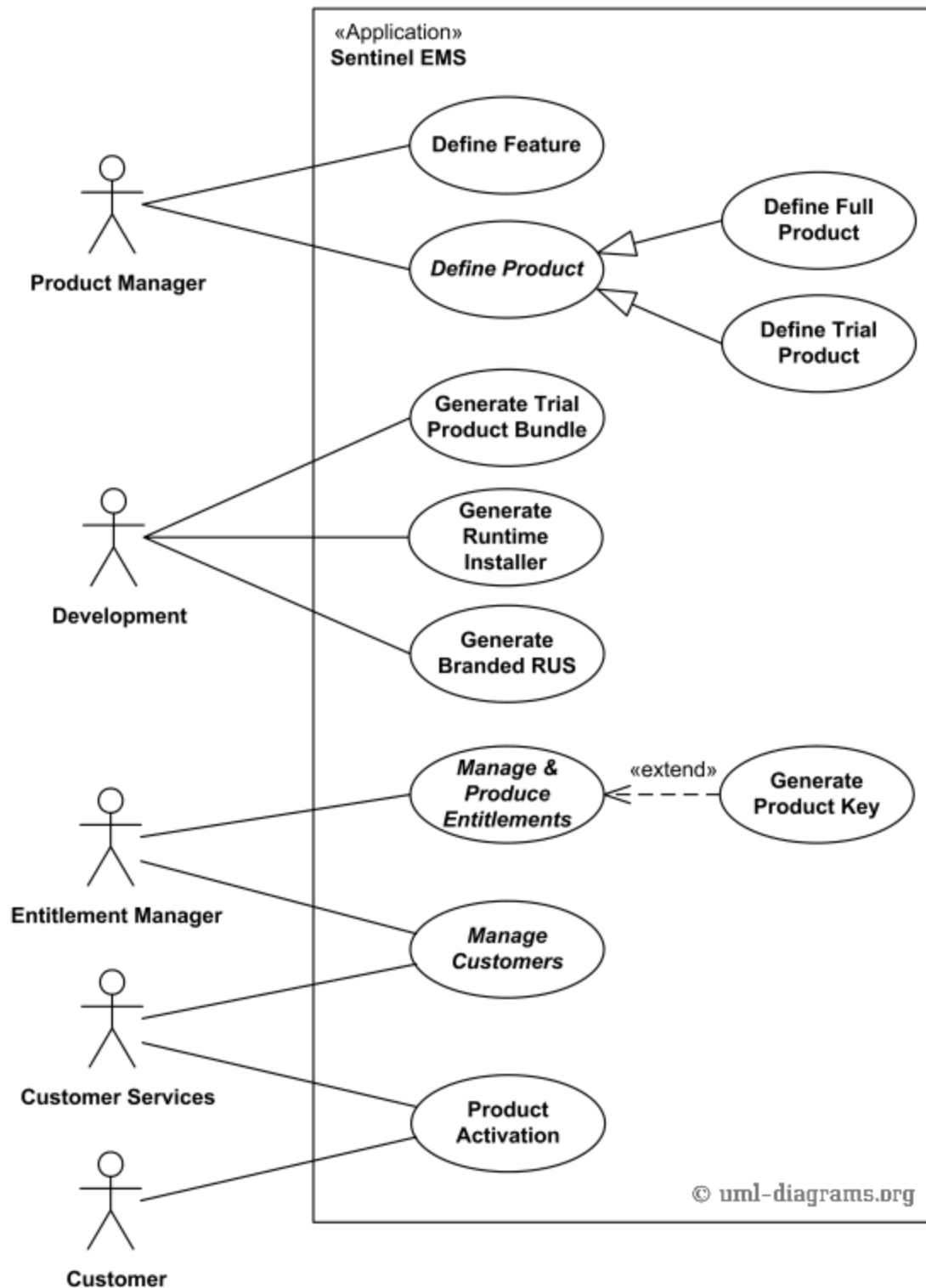
### *UML Use Case Diagram Example*

Sentinel License Development Kit (Sentinel LDK) is a Software Digital Rights Management (DRM) solution by SafeNet Inc. that delivers strong copy protection, protection for Intellectual Property (IP), and secure and flexible licensing. Sentinel LDK separates licensing and production processes (implemented with Sentinel EMS) from the software protection process (implemented with Sentinel Licensing API or Sentinel LDK Envelope).

Sentinel EMS is a web-based graphical application provided as part of Sentinel LDK that is used to perform a range of functions required to manage the licensing, production, distribution, customer support, and maintenance of protected applications. This application is a role-based application designed to manage the business activities required to implement and maintain Sentinel LDK in the organization which needs to protect its software. Sentinel EMS Server maintains a database containing a wide range of information, including data related to product features, licenses, sales, orders, and customers.

Use case diagram below shows some simplified view of software licensing use cases supported by Sentinel EMS Application (shown as «Application» stereotyped subject). The Sentinel EMS handles three major workflows:

- license planning,
- order processing and production, and
- activation of trial software.



*Software licensing with Sentinel EMS application UML use case diagram example.*  
 Product Manager defines Features and Products. Each Product has one or more Features. After Features and Products have been defined in Sentinel EMS, entitlements can be processed and produced using the Production group of functions.

Users assigned the Development role can fulfil one of the following development-related activities:

- Generate bundles of Provisional (Trial) Products
- Generate a customized Sentinel LDK Run-time Environment (RTE) installer file
- Customize the Sentinel Remote Update System utility (RUS utility)

Entitlement Manager defines and manages customers, and also enters and manages entitlements. An entitlement is the execution of a customer order for Sentinel LDK items, and can be either an order for Products to be supplied with one or more Sentinel protection keys, or a Protection Key Update that specifies changes to be made to the license terms and/or data stored in Sentinel protection keys that have already been deployed.

Customer Services role can manage customers the same way as Entitlement Manager does, and can also manage Product activation.

For entitlements that generate Product Keys, the customer receives an email from Sentinel EMS that contains the keys. The customer is able to log in to the EMS Customer Portal using the Product Key in order to activate the Product.