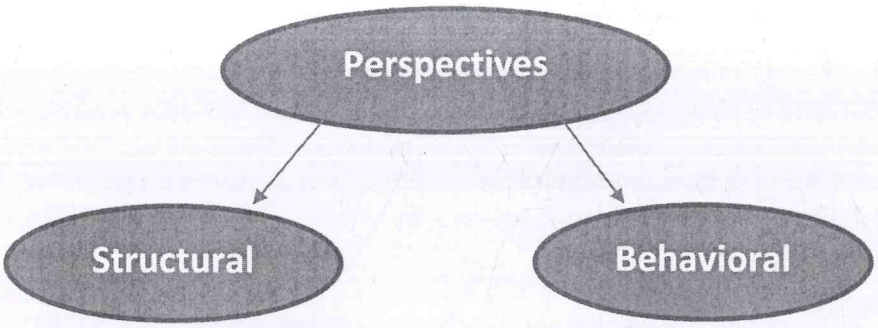
سنتكلم في هذه المحاضرة عن الـ System Modelling أي نمذجة النظام وكلمة نمذجة تعني رسم المخططات, وبدايةً سنستخدم هذه المخططات في نمذجة المتطلبات Requirements التي تعرفنا عليها في المحاضرات السابقة وذلك لهدفين هما **توثيقها** و **لتكون صلة الوصل بين الـ Analyst والـ Developer من جهة وبين الـ Analyst والـ Customer من جهة أخرى.**

ولذلك من المهم جداً توثيق هذه المتطلبات باستخدام المخططات بشكل دقيق و مفهوم بحيث تكون مرجع لكل من الزبون والمطوّرين في حال حدوث أمر ما, أي من الممكن أن تكون جزء من العقد.

إن عملية بناء المتطلبات بما فيها من بناء المخططات والوثاق تتم في مرحلة الـAnalysis ولقد تم تسميتها **بالتحليل** لأنه في هذه المرحلة يقوم المحللون Analysts بالتعبير عن هذه المتطلبات ليس كـكتلة واحدة و إنما من عدة وجهات نظر, حيث يمكن أن يكون للـ Customers وثيقة ونماذج مخصصة لهم ووثيقة للـDevelopers مخصصة لهم, ويمكن أن يتم جميع المخططات والنماذج في وثيقة واحدة وكل طرف يهتم بما يخصه وهو المنحى المنتشر...

سنلاحظ في هذه المحاضرة أن كل فكرة يمكن أن تتواجد **بأكثر من وجهة نظر** وقد تختلف من مرجع إلى آخر ويعود سبب الاختلاف إلى عدم تواجد صيغة موحدة لهذا العلم وأن كل الأفكار قد تم تطويرها من قبل شركات مختلفة فكل شركة تنظر للفكرة بالشكل الذي يناسبها ونحن علينا أن نستخدم ما نراه مناسباً وأن نطور ما أسماه الدكتور Common Behavior أي وجهة نظر مشتركة بين وجهات النظر هذه...

ولكن هنالك بعض وجهات النظر العامّة Perspectives والتي تصلح في أي مكان و منها:



❖ **Structural Perspective: منظور بنيوي**

وهو المنظور الذي يهتم بتحديد الكائنات **Entities** و العناصر **Components** التي يتألف منها النظام وبنية البيانات التي يعالجها, وسنستخدم في هذا المنظور **Class Diagrams**.

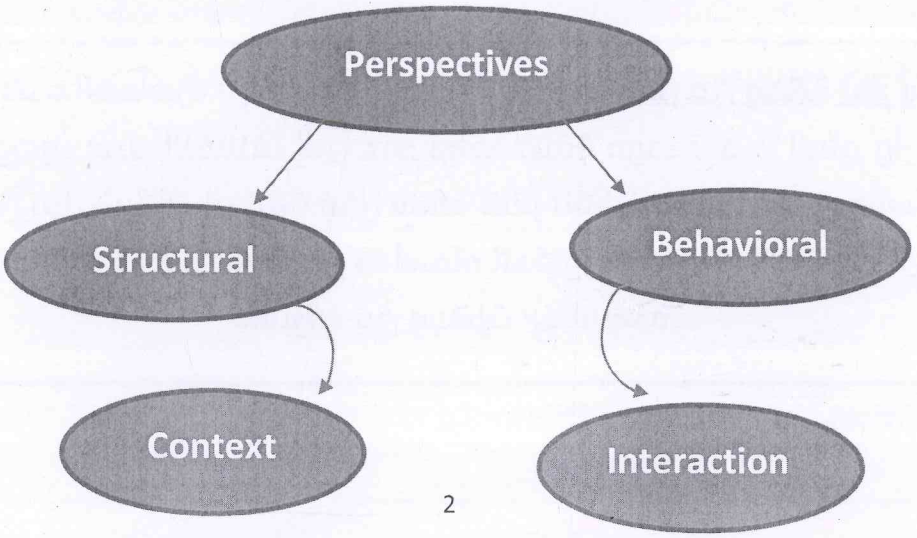
❖ **Behavioural Perspective: منظور سلوكي**

يعبّر عن عملية التفاعل **Interaction** بين النظام و الكائنات والأحداث **Events**.

ويتم بتحديد الكائنات المجردة **Abstract entities** التي تصف النظام وحركة الاتصال **Flow of The Data** داخل هذه الـ **Entities** .

ومن المخططات المستخدمة في هذا المنظور **State Diagram, Sequence Diagram**.

ويمكن إضافة وجهتي نظر بشكل أكثر تفصيلي على وجهات النظر العامّة السابقة وهي:



❖ Interaction Perspective: منظور التفاعل

يتم فيه نمذجة التفاعلات بين النظام والبية المحيطة به **Environment** أو بين عناصر النظام نفسه.

ومن المخططات المستخدمة هذا المنظور هو **Use Case Diagram**.

لكن ما الفرق بين الـ Interaction والـ Behavioural ؟

بعض وجهات النظر تعتبر الـ**Interaction** و الـ**Behavioral** وجهة نظر واحدة, وبعضها يميّز بينها كالآتي:

Behavioural: السلوكي

Set of interactions happens internally between set of objects or Entities to achieve certain functions.

Interaction: التفاعل

External interactions happens between the environment and the Software.

❖ Context Perspective: منظور السياق

هي الـ **External Projection** أو وجهة نظر المراقب الخارجي أي يحدد **فعلياً** أين تقع منظومة الـ **Software** الذي نطوّره ضمن البيئة الأكبر المحيطة به.

وهذا يعني أنه لا يوجد جزء مستقل وإنما دائماً سيكون جزء من منظومة أكبر وأعلى.

من بداية عملية وضع توصيف النظام **System Specification** يجب تحديد الـ **System Boundary** لمعرفة ماهي حدود الـ **Software** للعمل المنجز أي معرفة الـ **Environment** المحيطة به, ويتم ذلك بالعمل مع الـ**Stakeholders** أو باستخراجها من المتطلبات لتحديد الوظائف التي يجب أن يتم

تضمينها للنظام, فمثلاً يمكن أن يتقرر تضمين بعض المهام ليتم تنفيذها أو أن يتم تنفيذها يدوياً (موظف) أو أن تتم هذه المهمات بمساعدة نظام خارجي...

الـ Environment يمكن أن تتكون من Systems أو Humans **أي** كل ما هو خارج حدود العمل يعتبر من الـEnvironment.

DFD Diagrams:

من أحد النماذج أو المخططات في الـ **Context Perspective** هي مخططات Data Flow Diagrams (DFD) والتي من الوهلة الأولى لاسم هذه المخططات يمكن أن توحي بأن ليس لها علاقة بالسياق **Context** وإنما تتعلق بتدفق البيانات والمعلومات في النظام الذي يتم تطويره وهذا صحيح و**لكن** يمكن استخدام هذه المخططات في وصف البيئة المحيطة بالنظام وسياقه ونطلق عليها حينها Context DFD وستتألف من عدة مستويات level0,level1,level2.

وسنطلق على المستوى الذي يعبّر عن السياق بـ **Context DFD** أما المستويات الأخرى فتضيف قليلاً من التفاصيل على المخطط في كل مستوى...

الدكتور قد ميّز بين الـContext Level والـ **Level 0** ومن الممكن أن نجد بعض المراجع تعتبر الـContext Level هي نفسها الـ**Level 0**. (اختلاف وجهات نظر)

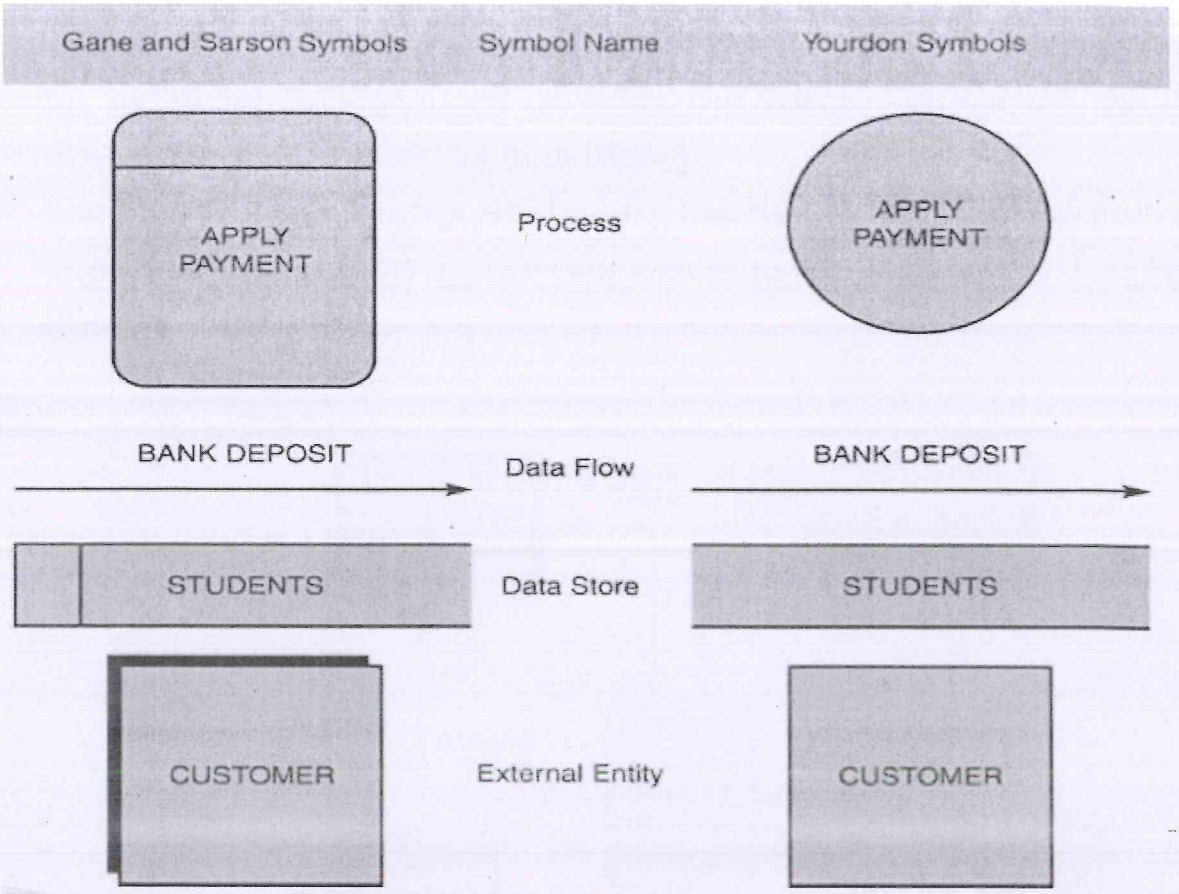
The DFD is presented in a hierarchical fashion. That is, the first data flow model (sometimes called a level 0 DFD or context diagram) represents the system as a whole. Subsequent data flow diagrams refine the context diagram, providing increasing detail with each subsequent level.

SE-Practitioner Approach 7th edition – Page 187

وسنتعرف على الرموز المستخدمة في هذه المخططات والتي هي:

Data Flow (Relation) – Process – External Entity

وهذه الرموز لها نوعين أو لنقل وجهتي نظر وهي **Yourdon Symbols** و **Gane and Sarson Symbols**:



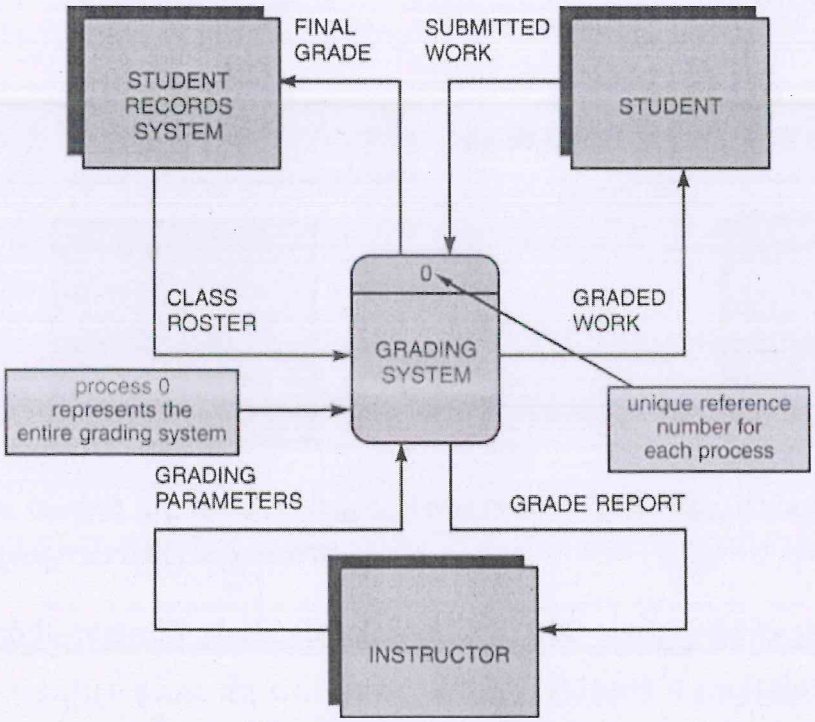
*الـ**Data Store** تستخدم في مستويات أدق من **Context Level** أو المستوى 0 وتستخدم للتوضيح في داخل الـ**Processes** وفي مخططات أخرى للـ**DFD**.

عندما نمذجة الـ **Context** باستخدام مخططات الـ **DFD** سيكون هناك **Process واحدة** وهي النظام الذي نقوم بتطويره وعدد غير محدود من **الكائنات** **Entities** و **Relations** أيضاً عدد غير محدود لكن ماهي انواع الـ **Relations** التي ممكن أن تتواجد في مخططات الـ**DFD**:

Data Flow That Connects	*Okay to Use?*	
	YES	NO
A process to another process	☑	☐
A process to an external entity	☑	☐
A process to a data store	☑	☐
An entity to another entity	☐	☑
An entity to a data store	☐	☑
A data store to another data store	☐	☑

- من غير المنطقي أن أقوم بربط الـ **Entities** ودراسة العلاقات المتبادلة بينها وبالتالي تكون علاقة **Entity to Entity** **<u>غير قابلة للتحقيق.</u>**
- الـ **Process** هو الوسيط الحصري لعملية تبادل المعطيات الداخلية والخارجية ومنه تكون علاقة **Entity to Data store** وعلاقة **Data store to another data store** أيضاً **<u>غير قابلة للتحقيق.</u>**

<u>مثال:</u>



كما قلنا سابقاً أن الـ **Context** يحوي **<u>Process واحدة</u>** وستكون هنا **Grading System** وسيكون لدينا **<u>عدد من الـ Entities</u>** وهي المدرس **Instructor** ونظام ثانوي لسجلات الطلاب **Students Records System** و الطالب **Student**.

يوجد مثال آخر في السلايد 19 من سلايدات System Modelling يوضح Levels of DFD Context Diagram .

Use Case Diagrams:

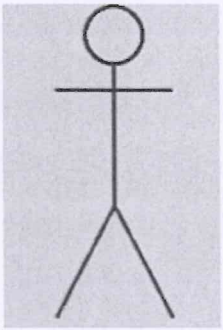
حالات الاستخدام Use Cases تعني كيفية تبادل الـInteraction بين النظام والبيئة والكائنات المحيطة به وبشكل عام هي مجموعة الخدمات والوظائف (المتطلبات) التي يقدمها النظام إلى Entity ما وتعد موجهة بشكل رئيسي إلى الـ User, أي يجب أن نبتعد عن التعقيد فيها.

فهي تعتبر قاعدة أساسية للتواصل مع الزبون وتأكيد المتطلبات ويمكننا من خلالها اشتقاق المخططات التحليلية والتصميمية للنظام في مراحل نمذجة متقدمة.

لايوجد في الـ Use Case Diagram داعي لتمثيل الوظاف والـObjects الداخلية في النظام فقط نمثل الـ Business Processes أو الوظائف المستقلة التي يمكن ان نحصل عليها نتيجة طلب ما, أو الوظائف التي تحقق متطلبات المستخدمين.

رموز مخططات حالات الاستخدام : Use Case Diagram Symbols

a. **Actor:** هو كينونة Entity تُعتبر المتفاعل (الممثل) الوحيد في البيئة المحيطة النظام ويمكن أن يمثل شخص أو منظمة أو أي أجهزة أو أنظمة خارجية ويتم رسمه على شكل Stick Figure :



وله عدة أنواع ويكون الرئيسية منها:

- Primary business actor:

هو الكينونة التي تطلب خدمة من النظام.

- **Primary system actor:**

هو الكينونة التي تتواصل بشكل مباشر مع النظام.

على سبيل المثال في الـ **Queuing System** لبعض الشركات يمكن أن يتواجد موظف خاص لإعطاء ورقة الدور والتسجيل في الرتل ويكون هو الـ**Primary System Actor** أما الزبون الذي بمجيئه وطلبه دور قد فعّل الحدث يكون الـ**Primary Business Actor**.

بعض وجهات النظر تقوم بنمذجة فقط الـ **Primary System Actors** أما الـ **Primary Business Actors** يتوضحون بحقل الـ **Trigger** في التمثيل الجدولي أو النصي لحالات الاستخدام.

هل من الممكن أن يكون الـ **Business Actor** و الـ **System Actor** هو نفس الشخص؟

بالطبع ممكن ومثال على ذلك عندما يريد شخص أن يسحب مبلغ من الصراف الآلي **ATM** عن طريق بطاقته الأئتمانية فهو عندما يقوم بإدخال معلومات البطاقة يكون **System Actor** وعندما يحصل على المبلغ (أي استفاد) يكون **Business Actor** .

أما الحالات الأخرى لـ **Actors** فتكون ثانوية ويطلق عليه **Secondary Actor** أو **Server Actor**:

- **External Server Actor:**

هو من يتواصل مع النظام ولكن بمرتبة أقل من الـ **Primary** حيث يستجيب لخدمة تم طلبها من قبل **Primary Actor** ويقوم بالرد و**مثال على ذلك** مدير منتدى عندما يتلقى طلب انضمام إلى هذا المنتدى و تكون مهمته ويرد قبول أو رفض الطلب.

- **External Receiver Actor:**

هو أيضاً يتواصل مع النظام ولكن لا يقوم بأي عملية رد أي أنه يستقبل التعليمات أو المعلومات من النظام فقط و**مثال على ذلك** أمين مستودع يتلقى طلبات من النظام ويقوم بشحنها إلى وجهاتها.

الفرق أن الـ **Server Actor** يقوم بعملية تفاعل مع النظام أي أنه يقوم باستقبال طلب ما من النظام و بعد ذلك يرد هذا الطلب بينما الـ **Receiver Actor** يقوم فقط باستقبال طلب من النظام.

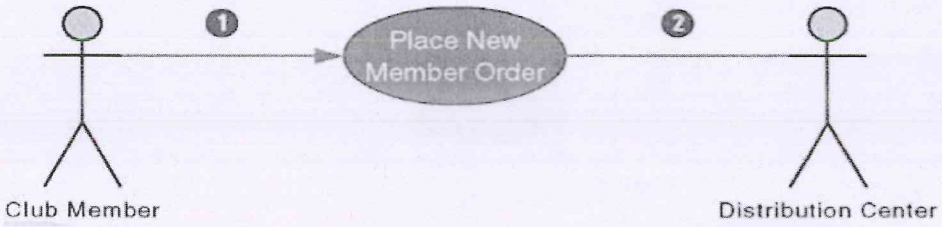
b. **Use case:** شكل أهليلي نضع في وسطه اسم هذه الحالة, يعبر عن خدمة أو وظيفة يقدمها النظام.

c. **Relations:**

وهي التي تربط طالب الخدمة **Actor** مع الخدمة **Use case** التي يطلبها أو بين خدمة و أخرى ولها عدة أنواع:

- **Association**:

هي العلاقة التي تربط بين **Actor** و **Use Case**.



وعادةً مايكون المهم في المخطط هو فقط الـ **Primary Actors** أما الـ **Secondary Actors** فيتم إسقاطهم خاصة عندما تكون التفاصيل كثيرة وكبيرة في المخطط لأنها ستعقد أكثر من أن تفيد.

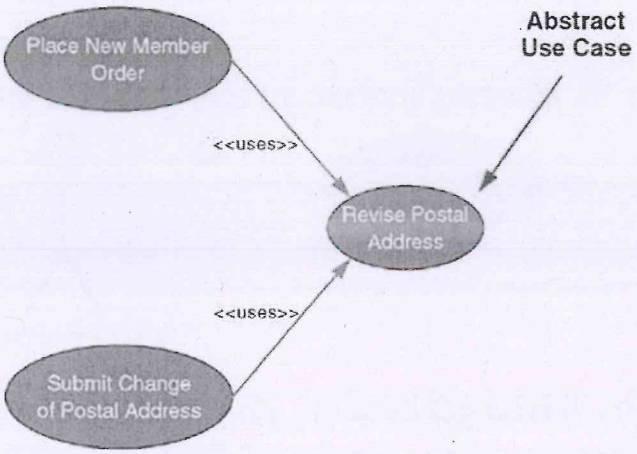
وفي نسخة الـ **UML الثانية** سنلاحظ في كثير من الأدوات أنه قد تم إسقاط إشارة السهم من الخط الواصل بين الـ**Actor** و الـ **Use case** وقد كانت تستخدم إشارة السهم للتميز بين المفعّل للـ **Use case** أي طالب الخدمة وبين المستفيد منها وقد لاحظوا أن **90%** من طالبي الخدمة هم المستفيدين منها لذلك تم اسقاطها.

➢ **Abstraction**:

تستخدم لصياغة وظيفة أو حالة استخدام في النظام والتي يمكن أن يتكرر استخدامها في حالات استخدام أخرى.

مثال: حالتي استخدام للتسجيل طلب في نظام ما, لدينا الحالتين **Place New Member Order** أي تسجيل طلب والحالة الثانية **Submit Change of Postal Address** وهي حالة تغيير عنوان بريدي.

سنلاحظ أن كلا الحالتين تشتركان في عملية **تأكيد العنوان البريدي**, أي يمكن فصل هذه الحالة عنهما ولنسميها **Revise Postal Address** وتكون هذه الحالة حالة تجريدية **Abstract Use Case** اي لايمكن تفعيلها لوحدها (لا يستطيع المستخدم تفعيل حالة **تأكيد العنوان البريدي**) وإنما يتم تضمينها **Include** في حالات أخرى, وتكون **إجبارية الحدوث** أي في حالة **تسجيل العضوية** من المؤكد أنه سيتم تفعيل حالة **تأكيد العنوان البريدي.**

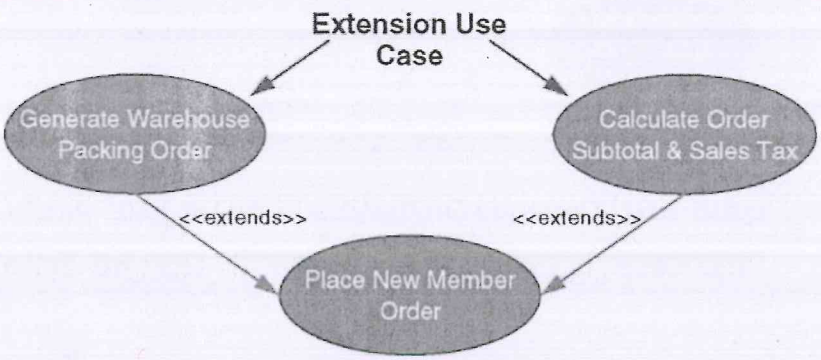


UML notation: a directed arrow labeled *<<uses>>* or *<<includes>>* and it is read as the example: Place new Member Order *Uses* Revise Postal Address.

➢ **Extension**:

تستخدم لصياغة وظيفة أو حالة تستكمل حالة أساسية أخرى, أي حالة **اختيارية** يمكن أن يتم تفعيلها من حالات أخرى عند تحقق شرط معين من قبل الـ **Actor**.

مثال: عملية تسجيل طلب **Place new member order** يمكن أن يختار الـ **actor** ان يتم تسجيل الضراب المدفوعة مع الفاتورة **Calculate Order Subtotal** أو توليد فاتورة من المستودع مع الفاتورة العادية **Generate Warehouse Packaging Order** والحالات الأخيرة هي **Extension Use Cases**.



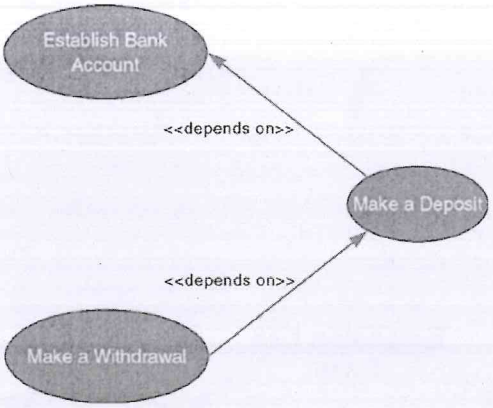
UML notation: a directed arrow labeled *<<extends>>* from the extension UC to the main UC and it is read as: Generate Warehouse Packing Order *extends* Place new member Order.

ملاحظة: اتجاه الأسهم في الـ **UML** لا يستدل به عن الحالة الأساسية من الحالة المشتقة, ويستخدم فقط في القراءة.

- **Depends On**:

علاقة تنشأ بين حالتي استخدام عندما يشترط حدوث الحالة الأولى حدوث الحالة الثانية قبلها.

مثل عملية سحب مبلغ من البنك تعتمد أولاً على عملية إيداع مبلغ في البنك وعملية الإيداع تعتمد على عملية فتح حساب وهكذا...

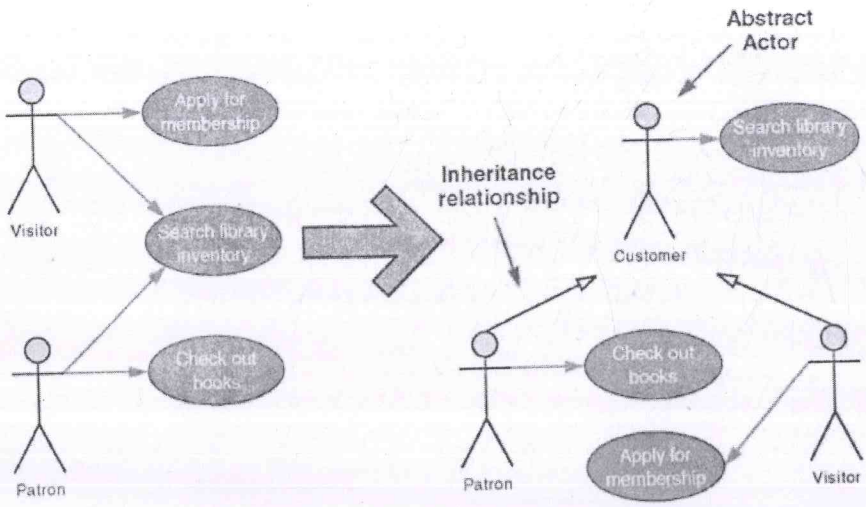


بعض وجهات النظر تعتبر علاقة الاعتمادية **Depends On** زيادة تعقيد للمخطط ويستغنون عنها بتوضيح الاعتمادية في حقل الـ **Precondition** في جدول **Use Case**.

UML notation: a directed arrow labeled *<<depends on>>* from the UC that depends on another UC.

- **Inheritance:**

علاقة نستخدمها عند وجود تصرفات مشتركة بين أكثر من **Actor**, فنقوم بجعل **Actor** واحد ويسمى **Abstract Actor** بتفعيل هذه الحالات المشتركة ثم نقوم بجعل الـ **Actors** الآخرين يرثون منه هذه التصرفات (التفعيلات) التي يقوم بها.



UML notation: a directed arrow beginning at one actor and pointing to the abstract actor whose interactions the first actor inherits.

ملاحظة:

يجب أن يكون هناك دائما حدود للنظام بحيث يكون الـ **Actors** ينتمي لـ **environment** والـ **Use Cases** تنتمي لـ **System Boundary** أي يجب عدم وضع الـ **Actor** ضمن الـ **System Boundary** ولكن كلاً منهم في جهة لوحده.

Primary actor أهم من **Secondary actor** أي عند الرسم الأولوية لـ **Primary** وفي حال كان الرسم يحتاج إلى تفصيل أكثر نضع الـ **Secondary**.

في اخر سلايدين من **System Modelling** يوجد مثالين لمخططين **Use Cases** مع نمذجة نصية (جدولية) لـ**Use Case Diagram**.

-النهاية-
