هيا للننمذج Let's Model

الجزء الأول Part I

المتطلبات Requirements

فهرس الموضوعات

هيا للننمذج 1
تمهيد 2
مقدمة 2
المتطلبات 2
الخطوة الأولى 2
النظرة الأولى نظرة على العمليات 3
النظرة الثانية نظرة على التركيب 4
النظرة الثالثة نظرة على التفاعل أو التصرف 6

فهرس الصور

Figure 1 مثال على المكونات 3
Figure 2 مثال تطبيقي على النظام المعطى 4
Figure 3 ليتبين لنا كيف نستخدم هذا النموذج لننظر إلى المثال التالي 4
Figure 4 هذا مثال آخر يوضح كل شيء 5
Figure 5 مكونات State Transition Diagram 7
Figure 6 مثال على البريد STD 8

تمهيد

كما وسبق أن وعدت بتقديم مثال متكامل للنمذجة بلغة UML، فأنا أقدم بين يدي القارئ الكريم هذه المحاولة المتواضعة مني. عسى أن يجعلها ربنا سبحانه وتعالى ذات فائدة.

مقدمة

تبدأ هذه الورقة بشرح متطلبات المشروع (المثال) المراد شرحه ومن ثم تبين مراحل النمذجة ابتداء من المتطلبات حتى كتابة الكود (الشفرة) بأي لغة كانت (في هذا المثال سنتبع طريقة java أو C#) .

المتطلبات

لنفترض أن زبوننا أراد برنامجا لتنظيم البريد في مكتب الخدمات البريدية. وبعد إجراء اللازم من طرق جمع المتطلبات من مقابلة وقراءة للوثائق وغيرها من الطرق المعروفة لدينا.

ولنفترض أن مجموعة المتطلبات هي كالآتي:

- يستطيع المشترك أن يعمل ثلاث أشياء:
  - الإرسال
  - الاستقبال
  - يغير عنوانه
- يتكون الإرسال من أشياء معينة
  - إرسال سريع
  - إرسال عادي

هذا يكفي وإذا احتجنا متطلبات إضافية ، سوف نقوم بسؤال الزبون عنها.

الخطوة الأولى

الآن نحن في مرحلة جمع المتطلبات وتحليلها. لذا علينا أن ننمذج ما فهمنا حتى الآن من النظام لكي يتبين لنا مدى مقدار فهمنا للنظام الصغير. سنحتاج هنا إلى تطوير ثلاث نظرات للنظام وهي نظرة على العمليات و نظرة على التصرف ونظرة على التركيب. قد يتبادر لذهن القارئ في هذه اللحظة أني أتكلم عن شيء غريب جدا ولكن مع نهاية الورقة سيتبين هذه النظرات ويستطيع التفريق بينها.

إن النمذجة تحتاج إلى هذه الثلاث نظرات لتشمل جميع الجوانب الممكنة للبرنامج. أيضا نحتاج إلى طريقة للتأكد من أن هذه النظرات متجانسة وتتكلم عن نفس الشيء.

النظرة الأولى نظرة على العمليات

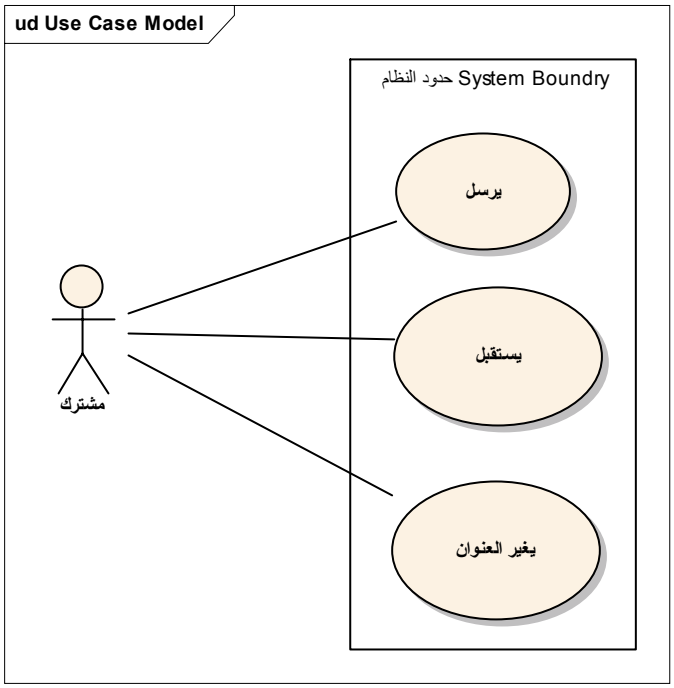
إن UML تقدم نوع من الرسم يدعى Use Case Diagram(UCD) أو رسمة استخدام الحالة وأنا أفضل أن أستخدم هنا المصطلح الإنجليزي لأن الترجمة الحرفية للرسم قد لا تكون دقيقة ولا تؤدي المعنى.

إن Case Use تتكون من ثلاث أشياء رئيسية هي:

1- Case Use وهو عبارة عن مجموعة من العمليات تنفذ عن طريق المستخدم تخدم في النهاية عمل واحد فقط. وتمثل في الرسم بشكل بيضاوي يكتب فيه اسم العملية التي تنفذها الـCase Use.
2- الممثل أو Actor وهو أي شيء يتعامل مع النظام من الخارج قد يكون المستخدم المباشر أو شخص متأثر بالنظام أو حتى نظام آخر يتعامل مع النظام. ويرمز للمثل في هذه الرسمة بشكل رسمة الرجل الأسود.
3- العلاقة أو Relation or Acociation. وهي ما يربط بين "اليوس كيس" وبين الممثل ويرمز لها بخط وله أنواع تجدها في وثائق UML.

مثال على المكونات Figure 1

مثال تطبيقي على النظام المعطى Figure 2

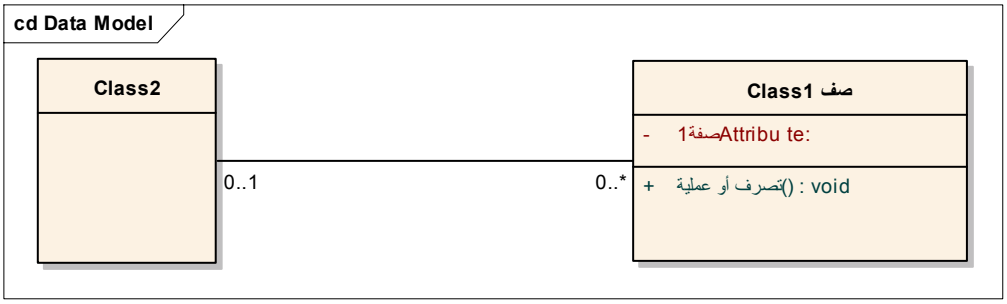


النظرة الثانية نظرة على التركيب

هنا تقدم UML طريقة رسمية في غاية الأهمية تدعى نموذج الصف المعنوي أو قل Conceptual Class Model (CCM). وهنا تأتي خبرة الشخص في الكائنة المنحى ، حيث يتكون النموذج من شيئين مهمين هما الصف Class و الرابط Association.

ويتكون الصف من صفات و تصرفات. وتكون فائدة الصفات في شرح صفات هذا الصف بيد أن التصرفات تشرح أعمال ومسؤوليات الصف.
يجب أن يكون الصف مترابطا عالي التجانس أي يجب أن يصف ويتكلم عن نفسه فقط ويصف شيئا واحدا فقط.

ليتبين لنا كيف نستخدم هذا النموذج لننظر إلى المثال التالي Figure 3



نلاحظ هنا أن هناك صفان ترتبطان مع بعضهما بعلاقة معينة تقرأ كالتالي: لـ2Class على الأقل صفر و على الأغلب علاقة واحدة مع 1Class. أيضا لـ Class1 على الأقل صفر و على الأغلب عدة علاقات مع Class2.

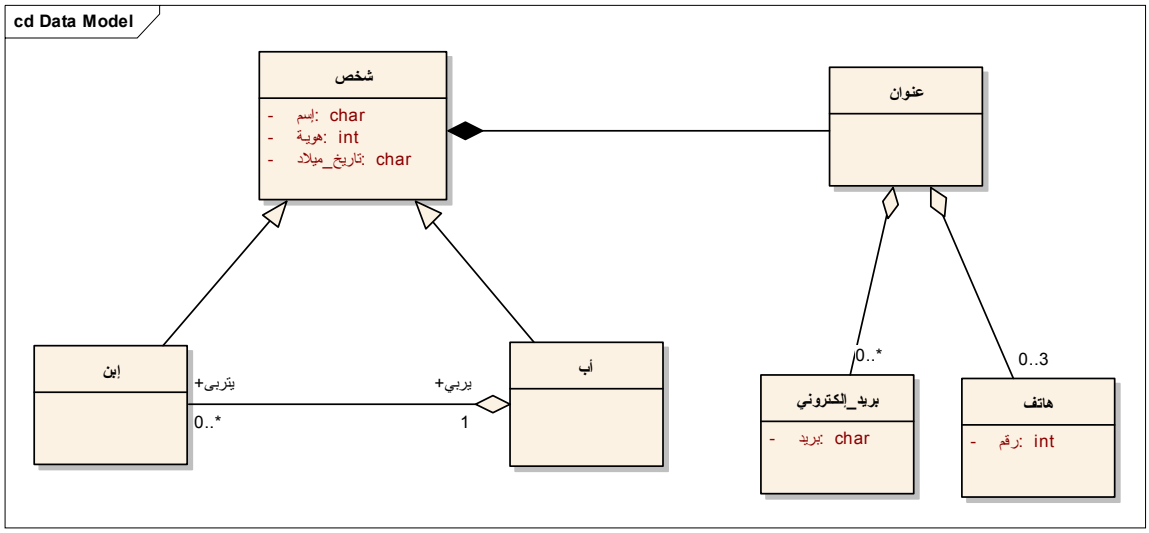
ما هي الخيارات المتاحة؟

إن لكل طرف في الرابط عدد يستخدمه ليصف عدد العلاقة مع الطرف الآخر. لذا فمن الممكن أن يكون الطرف (*) ويعني عديد أو (0) ويعني احتمالية عدم وجود العلاقة أو (1) وتسمى أيضا ( ) بدون أي رقم ويعني أن العلاقة من طرف واحد. أيضا لو وضعنا رقمين متجاورين فأحدهما يعني الحد الأقصى والآخر الحد الأعلى.

نستطيع أن نحدد شكل العلاقة هنا بأحد الأنواع التالية:

- علاقة "هو" وتعني علاقة الوراثة المعروفة لدى الكائنة المنحى وهي أن الابن هو الأب بالوراثة.
- علاقة "جزء من" وتعني أن الصف الأول هو "جزء من" الصف الثاني. أي أن الصف الأول هو في الأساس صفة موجودة في الصف الثاني.

هذا مثال آخر يوضح كل شيء Figure 4



نلاحظ هنا أن الشخص يحمل الصفات التالية (الاسم من نوع حرف و هوية من نوع رقم و تاريخ ميلاد من نوع حرف) لاحظ أن لكل صفة اسم ونوع. هذه الأنواع معروفة في لغات البرمجة مثل (الحرف ، الأعداد ، بوليان "صح أم خطأ"). طبعا الصفان "أب" و "ابن" يأخذان تلك الصفات بالوراثة بالإضافة إلى الصفات المعينة لكل من هما. وقد تم رسم علاقة الوراثة بالسهم المفرغ كما هو مبين في الرسمة. وتقرأ علاقة الوراثة كالتالي: (الأب هو شخص) و (الابن هو شخص).

هناك شيء آخر وهو علاقة "التجزؤ" أو سمها علاقة "البعض من كل". في الحقيقة للصف "شخص" يوجد 4 صفات ذكرت 3 منها سابقا أما الرابعة هي صفة "العنوان". ولأنها صفة معقدة – أي تكون صفا بذاته– لذا استخدمنا علاقة "الجزئية" و تقرأ كما سبق وشرحنا هكذا: ("العنوان" جزء من "الشخص" أو "الشخص" يحوي "عنوان"). أيضا لو نلاحظ أن "الهاتف" جزء من "العنوان" و "بريد_إلكتروني" جزء من "العنوان".

توضيح مهم: ما هو الفرق بين علاقة "العنوان" مع "شخص" وبين علاقة "الهاتف" مع "العنوان"؟ إن علاقة "الجزئية" الملونة باللون الأسود– أو مصمتة إذا أردت الدقة– تسمى علاقة "الجزئية القوية Composition " و الأخرى تسمى علاقة "الجزئية العادية Aggregation". وتسمى علاقة "الجزئية القوية" بهذا الاسم لأن طرفي العلاقة لا يتغنيان عن بعض و يتطلب وجود أحدهما الآخر. مثل

أننا لا نستطيع في النظام الموصوف لدينا في الرسمة أن نسجل شخصا بدون عنوان. أو بمعنى آخر لا نستطيع أن يكون لدينا عنوان ما بدون شخص مرتبط به.

الآن وقد وضحت صورة الرسمة ، ما بقي سهل جدا وسنقرئه سريعا. العنوان يتكون من الآتي : قد لا يحوي بريدا إليكترونيا وقد يحوي العديد منه للشخص الواحد. أيضا إن العنوان قد لا يحوي هاتفا وقد يحوي إلى ثلاثة هواتف للشخص الواحد.

وفي النهاية نقرأ العلاقة بين الأب والابن. إن الأب والابن هما في النهاية شخص. و قد يكون لكل أب ابن يربيه. ولكن لكل ابن أب وحد فقط.

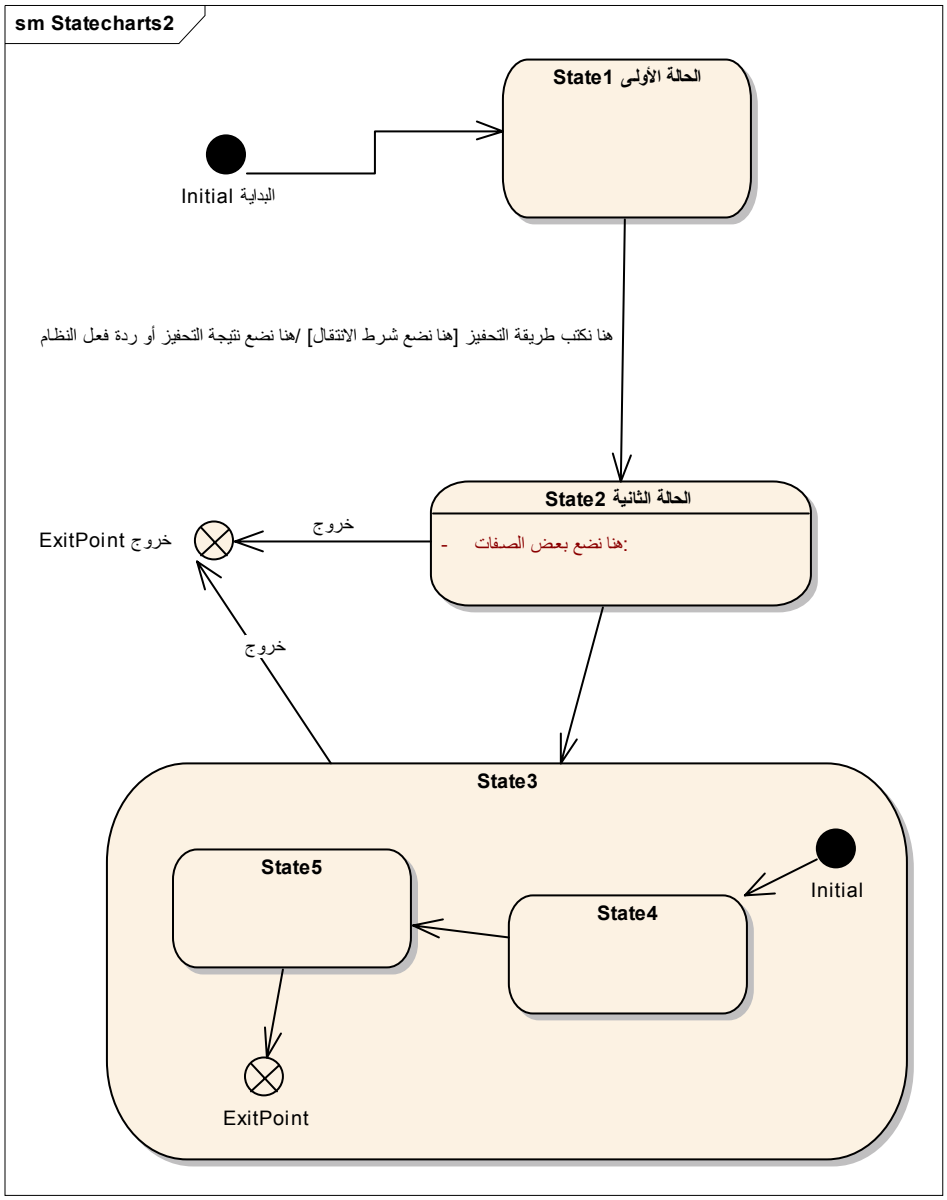
لكل أنهي هذا الفصل أو الرؤية الثانية للنظام أحب أن أنبه على أن المتعلم للـUML يجب أن يقرأ وثائق UML ليعرف باقي التفاصيل. لأن هناك أشياء كثيرة لم أذكرها مثل أنواع الصفوف "الكلاسات" و طرق بناء العلاقات. وقد اعتقدت أن ما تم تبيانه هنا كاف للبدء للنمذجة. وشيء آخر هو أن ما كتبته كان باللغة العربية وهذا مخالف لما يجب أن يكون عليه نموذج UML.

النظرة الثالثة نظرة على التفاعل أو التصرف

هذه النظرة على النظام تكمل لنا الجزء الباقي من النظام وهو بعدما عرفنا تركيب معلومات النظام و ماذا يعمل النظام بقي لنا أن نعرف كيف تتصرف أجزاء النظام مع بعضها البعض وكيف ينتقل من حالة إلى أخرى.

لكل نظام حالات يكون عليها ويتغير فيها من حالة إلى أخرى حسب نوع التحفيز الذي يؤثر عليه من عناصر خارجية. لنأخذ مثالا واضحا وهو نظام جهاز بيع المشروبات. هذا النظام يكون دائما في وضع الاستعداد ويتحفز بوضع قطع معدنية داخله حتى يصل لمرحلة الاكتفاء و من ثم يحفز بضغط المستخدم على زر اختيار المشروب المراد ومن ثم يقوم الجهاز بإخراج المشروب والعودة إلى وضع الاستعداد مرة أخرى. تعرف هذه العملية بجهاز الحالات النهائي أو Final State Machine.
توفر UML نموذجا يشرح فيه هذه التصرفات للنظام والتي تقوم على مبدأ تحفيز المستخدم للجهاز و تفاعل النظام مع هذا التحفيز. هذا النموذج يدعى نموذج الحالات الانتقالية أو"State Transition Diagram(STD)". يتكون النموذج من شيئين رئيسيين الحالة والانتقال. تمثل الحالة على شكل مستطيل مستدير الجوانب و يمثل الانتقال على شكل خط يصل بين حالة وأخرى. يكتب على كل خط الفعل الذي يفعله المؤثر الخارجي وردة فعل النظام. ولتوضيح ما ذكرت نأخذ مثالا مثل العادة يفصل ما شرحنا.

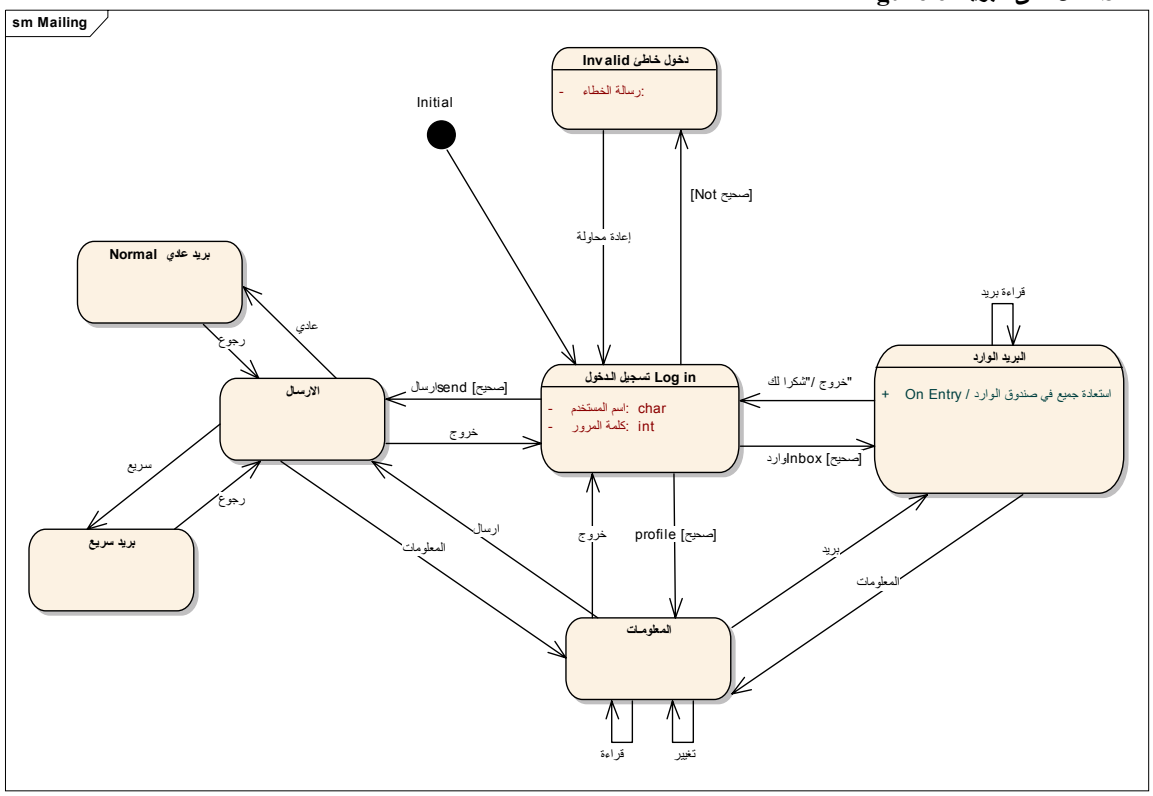
Figure 5 مكونات State Transition Diagram



لاحظ طريقة كتابة الحالة وكيفية الانتقال من حالة إلى أخرى وكيفية وضع شروط على الانتقال وبيان نتيجة الانتقال.و تستطيع أن ترى أن هناك بعض الحالات نضع لها صفات ونستطيع أن نضع لها مواصفات تتصرف على أساسها وبقي لنا أن نعرف أن لكل نموذج بداية إجبارية ونهاية اختيارية. لأن هناك بعض الأنظمة لا تتوقف أبدا مثل نظام جهاز المشروبات الذي تكلمنا عنه.أيضا نستطيع أن نضع نموذج داخل حالة من الحالات مثل الحالة رقم 3. ولو أمعنت في الرسمة لوجدت أن المستخدم لا يخرج إلا إذا حفز النظام على الخروج بالضغط على زر الخروج. بالإضافة إلى ذلك لو أراد المستخدم الخروج وهو في الحالة 1 فلا يستطيع ذلك لأن ليس هناك انتقال معرف بذلك ولكن المستخدم يستطيع الخروج لو كان في الحالة رقم 4 لأن انتقال الخروج معرف للحالة الأم رقم 3.

المثال القادم هو المثال الذي استخدمناه في أول المقال، نظام البريد. و أريد أن يلاحظ القارئ الكريم أن كل حالة في نظامنا تقريبا تمثل شاشة حقيقية عند تطبيق النظام. وقد وجدنا أن للنظام حالات يتغير بينها حسب تحفيز المستخدم له (حالة الدخول، حالة الدخول الخاطئ،حالة الإرسال ومنها العادي والسريع ، حالة استقبال البريد و حالة عرض العنوان).

Figure 6 مثال على البريد STD



الآن هذا المثال قد يكون طويلا بعض الشيء ، ولكنه مفيد جدا حيث أنه يمثل نظاما كاملا تقريبا من حيث التصرف. ينبغي على قارئ المثال أن ينتبه إلى الحالات التي ينتقل بينها النظام وكيف يؤثر تحفز المستخدم له. فمثلا إذا بدأ النظام فهو يكون في حالة "تسجيل الدخول" وهي معرفة لدى الجميع. فإذا ضغط المستخدم على "إرسال" بعدما وفر المعلومات المطلوبة – اسم المستخدم وكلمة المرور– وكانت هذه المعلومات صحيحة ، فينتقل النظام من الحالة الأولى إلى حالة الإرسال.

لن أطيل في شرح النموذج لأن يجب أن يكون لدى القارئ في هذه المرحلة القدرة على قراءة نماذج UML ، لكني سأبين بعض الأشياء التي قد تحتاج إلى توضيح. لاحظ أي انتقال بين حالة وأخرى تتحدد بتحفز المستخدم ويكتب فوق خط الانتقال هذا التحفيز (فعل المستخدم). فإذا كان للانتقال من حالة إلى أخرى شرط فيجب وضع هذا الشرط بين [قوسين] كما في حالة الانتقال من "تسجيل الدخول" إلى "إرسال" اشترطنا أن تكون عملية الدخول صحيحة. طبعا كلمة "صحيح" التي كتبت تعني وجود متغير ثنائي القيمة Boolean ليكون قيمة ما بين القوسين إلى "صح True" عندها ينتقل النظام من هذه الحالة للثانية. فإذا كان للنظام خارجا أو عملا يؤديه مع الانتقال فيكتب بعد هذه العلامة"/". مثل حالة الانتقال من "الوارد" إلى "تسجيل الدخول" بعدما أُمِرَ النظام بالخروج عندها يصدر النظام رسالة "شكرا لك" وينتقل إلى حالة "تسجيل الدخول".

بقي شيء واحد في المثال وهو ما يكون داخل الحالة ، وعادة يكون بعض الصفاتAttribute و الأهم من ذلك عمل الحالة. فلو نظرنا إلى مثالنا لوجدنا حالة "البريد الوارد" تعمل عند الدخول "On Entry" باستعادة كل ما في صندوق الوارد للمستخدم. توجد عدة خيارات أخرى مثل "On Exit" وهي ما يعمل به عند الخروج من الحالة. يوجد خيارات أخرى تجدها في وثائق UML قد لا تحتاج إليها الآن.

قبل أنهي النظرة الثالثة على النظام أحب أن أقول أنك قد لا تجد بعض الكتب تنمذج الحالات بهذه الطريقة ولكني اخترتها لأنها أكثر عملية ، فبعض كتاب UML لا يضع للحالات صفات. ولكن في الحياة العملية يستخدمونها أكثر على الرغم أنها ليست في كتب بل طورت مع الخبرة. وبذلك أكون الآن قد أنهيت النظرة الثالثة وبالتالي الخطوة الأولى في صناعة النظام وهي خطوات المتطلبات. طبعا يجب علي أن أنوه مرة أخرى أن UML هي أداة لتساعدنا في هندسة البرمجيات ، وبالتالي إذا لم تعرف ما هي هندسة البرمجيات وكيف تجمع المتطلبات عندها كل نمذجتك تعتبرها مضيعة للوقت. لذا على أي شخص يتعلم أن يعرف الثلاثة أشياء التي تكلمنا عنها في موضع غير هذا وهي "ماذا نريد؟" و "لماذا نريد؟" ومن ثم "كيف نعملها؟".

وشكرا...