الفصل الخامس

التحليل والتصميم الموجه لتقنية الكائنات

OBJECT-ORIENTED ANALYSIS AND DESIGN

أهداف الفصل:

- استكشاف تحليل النظم وتصميمها مستخدماً تقنية الكائنات.
- فهم لغة النمذجة الموحدة (UML).
- استخدام التصميم ثلاثي الطبقات في تطوير الكائنات.
- التعرف على نشاط شركة برادشو مارينا (الحالة الدراسية لهذا الكتاب).

تقدم لغة VB .NET بيئة برمجة سهلة ومتكاملة لاستكشاف المفاهيم الموجهة للكائنات والبرمجة المعتمدة على الكائنات (Object Oriented Programming). ولقد اكتشفت بعض وظائفها في الفصل الأول والثاني والثالث والرابع، وكما أوضحنا في الفصل الأول أن تطوير التطبيقات المعتمدة على الكائنات أكبر بكثير من معرفة البرمجة المعتمدة على الكائنات فقط ولكن الأمر يتطلب فهماً جيداً للمشكلة المطلوب حلها بالإضافة إلى وضع المخططات اللازمة لمكونات حل المشكلة. ولذلك يجب عليك تحديد الوظائف المطلوب من النظام القيام بها وتصميم هيكل النظام الجديد قبل البداية في برمجته واختباره. أي أنه باختصار يجب عليك الانتهاء من كل من مرحلة التحليل والتصميم للنظام.

يراجع هذا الفصل المفاهيم الأساسية للتحليل والتصميم وكذلك أنشطة تطوير النظم المعتمدة على تقنية الكائنات مستخدماً لغة VB .NET حيث سنبدأ بمناقشة تحليل النظم ثم تصميم النظم، مشتملاً على الأسلوب التكراري (Iterative Approach) وأسلوب العمل المتزايد (Incremental Approach) في تطوير النظم. كما يقدم هذا الفصل لغة النمذجة الموحدة (Unified Modeling Language) والتي تختصر بلغة UML وتستخدم لتمثيل مكونات النظام بأشكال رسومية مثل نموذج Use Case Diagram، ونموذج Class Diagram، ونموذج Sequence Diagram. ويتم التركيز أيضاً في هذا الفصل على أسلوب التصميم ثلاثي الطبقات (Three-Tier Design) وذلك لتوضيح كيفية

الانتقال من مرحلة التحليل المعتمد على الكائنات (OO Analysis) إلى مرحلة التصميم المعتمدة على الكائنات (OO Design). وأخيراً يقدم الفصل حالة شركة برادشو مارينا التي نفترض أنها تطلب أعداد نظام معلومات ومن ثم سنوضح جميع مفاهيم لغة VB .NET وجميع الأمثلة خلال بقية هذا الكتاب بناء على تلك الحالة.

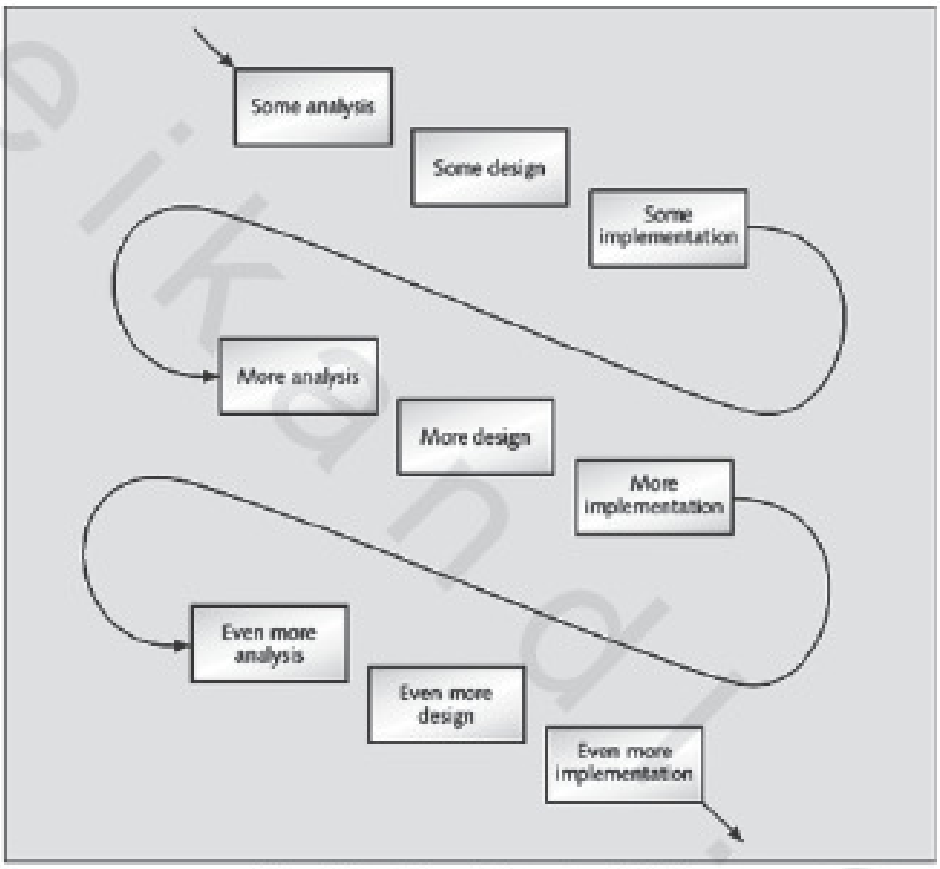
استكشاف كل من OOA و OOD
Exploring OOD and OOA

كما هو معروف أن تحليل النظم يعني دراسة مجموعة من المتطلبات التي يقوم بها النظام (System Requirements) وفهمها وتحديدها، والمتطلبات تعني احتياجات المستخدم من النظام حيث يتم التعبير عنها بلغة المستخدم، وعادة يتم تمثيل تلك المتطلبات مستخدماً أشكال (Diagrams) أو نماذج (Models) حيث يصف النموذج أحد أجزاء النظام (أي كل نموذج يصور ظاهرة ما للنظام). ومن ثم نحتاج إلى مجموعة من النماذج لتوصيف متطلبات النظام. ويطلق على مجموعة النماذج التي يتم إنشاءها أثناء مرحلة التحليل بالنماذج المنطقية (Logical Models) لأنها توضح فقط ما هو مطلوب من النظام بغض النظر عن التقنية التي ستستخدم لتطوير النظام. أما تصميم النظام فيعني إنشاء مجموعة أخرى من النماذج التي توضح مكونات النظام المختلفة والتي سيتم تطويرها مستخدماً تقنية معينة، ويطلق على هذه النماذج اسم النماذج المادية (Physical Models).

عملية إنشاء النماذج المنطقية أثناء مرحلة التحليل وإنشاء النماذج المادية أثناء مرحلة التصميم تسمى تطوير النظم المعتمد على النماذج (Model-Driven Development)، وكما ناقشنا في الفصل الأول أن طريقة التطوير الإجرائية التقليدية تنشئ أيضاً نماذج لمتطلبات النظام مثل شكل تدفق البيانات (Data Flow Diagram) الذي يوضح المدخلات والمخرجات والعمليات وكذلك شكل الكينونة والعلاقة (Entity-Relationship Diagram) الذي يوضح تفاصيل عن البيانات المخزنة. أما نماذج التصميم فتشمل أشكالاً تفصيلية عن مكونات النظام وعن قواعد البيانات التي ستحتوي على بيانات النظام.

وبشكل عام فإن النماذج التي يتم إنشاءها أثناء عملية تطوير النظم المعتمدة على الكائنات تختلف كثيراً عن تلك النماذج التي يتم إنشاءها أثناء تطوير النظم مستخدماً الطريقة التقليدية التي توصف البيانات والعمليات بشكل منفصل، فعلى سبيل المثال إن النظم المعتمدة على الكائنات تقوم بإنشاء نماذج تُعرِّف أصناف كائنات النظام كما تصف التفاعل الذي يحدث بين تلك الكائنات لإنجاز مهام النظام. وتعتمد هذه النماذج على لغة UML التي سيتم الحديث عنها بالتفصيل خلال هذا الفصل حيث تتنوع هذه النماذج مثل نموذج مهام النظام (Use Case Diagram)، ونموذج الأصناف (Class Diagram)، ونموذج تتابع الأحداث (Sequence Diagram)، ونماذج أخرى، ولأن تطوير النظم المعتمدة على الكائنات يركز على أصناف كائنات النظام خلال مرحلة التحليل ثم مرحلة التصميم وحتى

مرحلة البرمجة؛ لذلك فإنها تتفق تماماً مع الأسلوب التكراري لتطوير النظم التي تحدثنا عنه خلال الفصل الأول وهو الأسلوب الذي يتم فيه التحليل ثم التصميم ثم البرمجة التي تقود إلى المزيد من متطلبات النظام مما يستلزم إعادة نفس الكرَّة لتحديد المزيد من متطلبات النظام. وبمعنى آخر أنك تقوم بإتمام بعض التحليل قبل البداية في التصميم وإتمام بعض التصميم قبل البرمجة كما يوضحه الشكل رقم (٥,١).



الشكل رقم (٥,١). الأسلوب التكراري لتطوير النظم.

يختلف الأسلوب التكراري عن الأسلوب السابق والمعروف باسم أسلوب الشلال (Waterfall Approach) والذي تنتهي فيه عملية التحليل بالكامل قبل الدخول في عملية التصميم، وكذلك الانتهاء من عملية التصميم بالكامل قبل البدء في البرمجة. والآن أصبح الأسلوب التكراري يستخدم بكثرة مع الأسلوب القديم في تطوير النظم ولكنه يتناسب أكثر مع تطوير النظم المعتمدة على الكائنات لأن كل دورة تحليل-تصميم-برمجة تؤدي إلى تحسين أصناف كائنات أخرى وزيادتها.

إذا تم تقسيم النظام إلى عدة نظم فرعية وتم تطوير كل نظام فرعي على حده ووضعه في طور التشغيل قبل الانتهاء من النظام بالكامل نطلق على هذا الأسلوب من التطوير اسم أسلوب التطوير المتزايد (Incremental Development Approach)، فعلى سبيل المثال يمكن تطوير النظم الأكثر أهمية أولاً ثم تطوير باقي النظم الفرعية الأخرى. وأسلوب التطوير المتزايد يستخدم أيضاً بكثرة مع تطوير النظم المعتمدة على الكائنات وذلك لأن استخدام كل من الأسلوب التكراري وأسلوب التطوير المتزايد معاً لتطوير نظم تعتمد على الكائنات يكون مناسباً جداً حيث يحدث العديد من دورات تحليل وتصميم وبرمجة لتطوير أول نظام فرعي لتسليمه وتشغيله ثم تنفيذ العديد من دورات التحليل–التصميم–البرمجة لتطوير نظام ثانٍ وتشغيله مع النظام الأول وهكذا.

يوجد أيضاً أسلوب مشهور آخر يسمى النموذج الحلزوني (Spiral Model) والموضح في الشكل رقم (٥,٢). بهذا النموذج يمكن تمثيل الأسلوب التكراري في تطوير النظم حيث يظهر المشروع في الشكل رقم (٥,٢) كحلزونة تبدأ من الوسط وتتجه إلى الخارج. كل مشروع لديه مجموعة من المشاكل والمخاطر؛ لذلك فإن مطوري النظم يجب عليهم أن يحددوا المخاطر الكبرى التي يمكن أن تؤثر على نجاح المشروع والتركيز عليها في الدورة الأولى ومجموعة التطوير عليها أن تنتهي من التحليل والتصميم وإنشاء النموذج الأولي للنظام مع تقييم المهام في كل دورة قبل البدء في الدورة التالية بنفس الأسلوب.

الشكل رقم (٥,٢). الأسلوب الحلزوني لتطوير النظم.

إن تطوير النظم معتمداً على الكائنات يتطلب أيضاً معظم المفاهيم والتقنيات والأدوات المستخدمة في الأسلوب المعتاد للتحليل والتصميم حيث مازال هناك احتياج لتخطيط المشروع (Project Planning)، وإدارة المشروع (Project Management)، وهكذا.

التعرف على لغة UML
Understanding the Unified Modeling Language

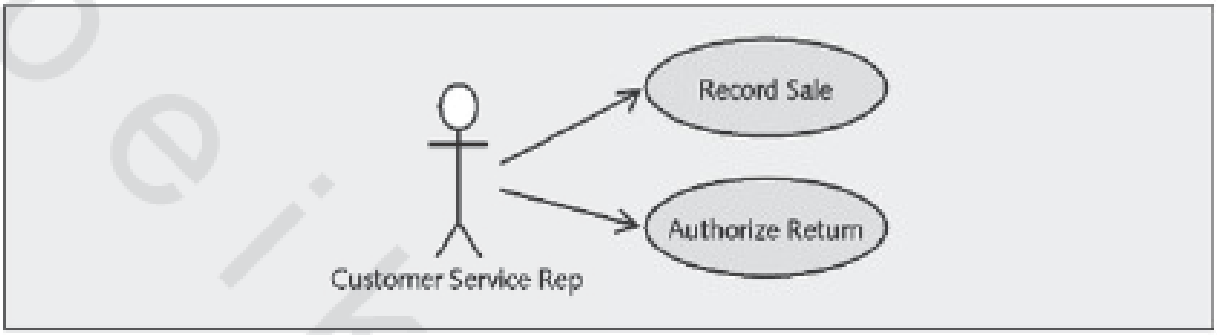
إن تطوير النظم معتمداً على الكائنات يتطلب مجموعة من النماذج والأشكال التي تصف تصميمات النظام ومتطلباته، وتقدم لغة UML مجموعة مثالية وموحدة من الأشكال والنماذج التي يمكن استخدامها لتوصيف النظم المعتمدة على الكائنات، حيث كان المطورون قبل عام ١٩٩٧م يستخدمون نماذج مختلفة في تطوير النظم المعتمدة على الكائنات مما أدى إلى صعوبة الاتصال بين فريق العمل الواحد؛ ولذلك كان هناك احتياج إلى أسلوب نمذجة موحد، ولهذا اجتمع ثلاثة علماء لوضع هذه اللغة وهم جرادي بوتش Grady Booch، وجيم رامبوتش James Rumbaugh، وإيفلر جاكوبسون Ivar Jacobson الذين ينتمون إلى شركة Rational Software وأنتج عملهم مجموعة من الأشكال والنماذج التي كونت لغة UML والتي تعتبر اللغة الموحدة والمثالية لتطوير النظم المعتمدة على الكائنات حيث اعتمدتها مجموعة إدارة الكائنات Object Management Group (OMG) وهي المؤسسة المسؤولة عن تحسين تقنية تطوير النظم المعتمدة على الكائنات وتطويرها.

ومن أهم نماذج لغة UML نموذج Use Case Diagram ونموذج Class Diagram ونموذج Sequence Diagram ويمكنك الحصول على معلومات كاملة عن لغة UML وجميع النماذج الخاصة بها بزيارة الموقع www.rational.com، كما يمكنك الاستعانة بالعديد من الكتب التي تعلم لغة UML وخصوصاً كتاب بعنوان "The Unified Modeling Language User Guide" (١٩٩٩م) والمؤلف بواسطة بوتش، ورامبوتش، وجاكوبسون.

إنشاء نموذج Use Case Diagram وتوضيحه
Creating and Interpreting the Use Case Diagram

إن أول خطوة في توصيف النظام هي تحديد الوظائف الرئيسة للنظام (ما يقدمه النظام للمستخدم)، ويطلق على كل وظيفة اسم (أحد حالات استخدام النظام). على سبيل المثال إن نظام معالجة طلبات العملاء لابد أن يؤدي وظيفة تسجيل عملية البيع (Record a sale) وبذلك تصبح هذه الوظيفة حالة استخدام (Use Case). وعملية تقسيم النظام إلى عدة حالات استخدام يسمح لمطوري النظام تقسيم العمل والتركيز على بعض وظائف النظام، ومن ثم إن كل دورة تحليل–تصميم–برمجة تركز على مجموعة من حالات الاستخدام. وهذه الطريقة في التطوير تشبه الأسلوب التقليدي القديم لتطوير النظم حيث كانت عملية التحليل تبدأ بتقسيم وظائف النظام.

يوضح نموذج Use Case Diagram مفهومان أساسيان، الأول حالة الاستخدام نفسها، والثاني هو المستخدم (يطلق عليه Actor)، والمقصود به مستخدم النظام أو كائن يستخدم النظام. ويعرض الشكل رقم (٥,٣) مثالاً لنموذج Use Case Diagram.



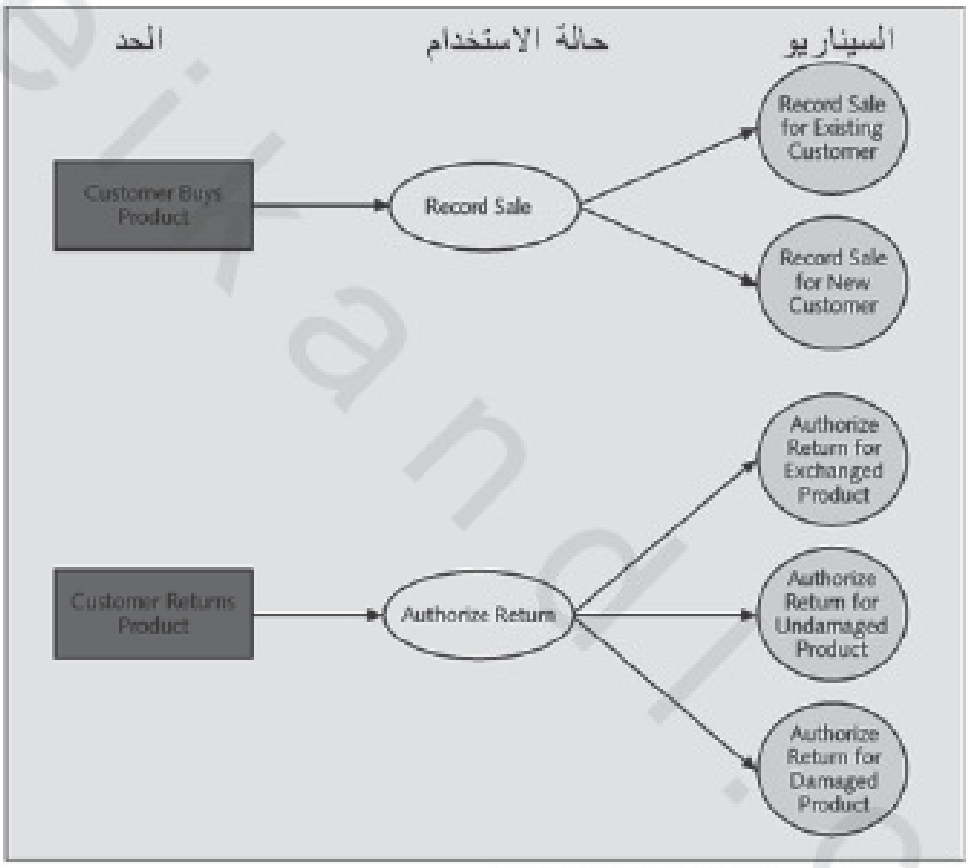
الشكل رقم (٥,٣). مثال لنموذج Use Case Diagram.

كما هو واضح في الشكل رقم (٥,٣) أنه يوجد مستخدم واحد فقط وهو مندوب المبيعات (Customer Service Representative) والذي يحتاج من النظام القيام بوظيفتين (حالتي استخدام)، الأولى هي حالة تسجيل مبيعات (Record Sale) والثانية هي حالة استقبال مرتجعات (Authorize Return). يمثل المستخدم (Actor) على شكل عصا (Stick Figure)، ويتصل المستخدم بسهم موجه إلى كل Use Case الذي يمثل على هيئة شكل بيضاوي (Oval). ولا يتطلب أن يكون المستخدم شخصاً فيمكن أن يكون المستخدم نظاماً آخر أو جهازاً لديه القدرة على إرسال بيانات واستقبال مخرجات.

والآن نسأل أنفسنا كيف يتم تحديد حالات استخدام النظام؟ وللإجابة على هذا السؤال، يمكن اتباع أحد الأساليب التي تحصر الأحداث التي يجب أن يستجيب لها النظام حيث توجد ثلاثة أنواع من الأحداث: أحداث خارجية (External Events) وأحداث زمنية (Temporal Events) وأحداث تغيير حالة (State Events). الحدث الخارجي هو حدث يحدث خارج النظام ويتطلب الأمر من النظام الاستجابة لهذا الحدث مثل حدث شراء منتجات أو استقبال مرتجعات، ولذلك عندما يشتري العميل منتجات يستجيب النظام من خلال Use Case اسمه "Record Sale"، وعندما يريد المستخدم أن يعيد منتجات يستجيب النظام من خلال Use Case اسمه "Authorize Return".

يمكن توثيق كل Use Case بواسطة مجموعة متتالية من الخطوات التي يتبعها المستخدم عندما يتفاعل مع النظام لإنهاء مهمة ما، وأحياناً يوجد أكثر من تسلسل للخطوات الرئيسة على مستوى Use Case واحد؛ ولذلك ربما يوجد أكثر من سيناريو لتفاعل المستخدم مع النظام، فعلى سبيل المثال إن خطوات تسجيل مبيعات لمستخدم

جديد ربما تختلف عن تسجيل مبيعات لمستخدم معرف مسبقاً، ويوضح الشكل رقم (٥,٤) أمثلة مختلفة على كل من الإجراءات الخارجية والسيناريوهات الخاصة بها. ويحدث الحدث الزمني عند نقطة زمنية محددة مثل نهاية اليوم أو نهاية الشهر، وعندئذ يمكن إصدار فواتير شهرية مثلاً أو إصدار تقارير دورية طبقاً لجدول محدد. ويجب أن تقود تلك الأحداث أيضاً إلى حالات استخدام والتي ربما تكون ذات سيناريوهات متعددة. ولأن حالات الاستخدام تعتمد على أحداث زمنية فإن التفاعل بين المستخدم والنظام يقل ولكن تنتج هذه الحالات مخرجات هامة.



الشكل رقم (٥,٤). الأحداث وحالات الاستخدام والسيناريوهات.

أما أحداث تغيير الحالة فتحدث عندما تتغير قيم صفات كائن ما والذي يتطلب معالجة من النظام، فعلى سبيل المثال يحدث هذا الحدث عندما تقل كمية المخزون من منتج ما عن الكمية التي يجب عندها إعادة طلب هذا المنتج من المورد، عندئذ لابد من إضافة حالة استخدام لهذا الحدث. ويحدث هذا الحدث أيضاً عندما تقل قيمة GPA لطالب ما في أحد الكليات عن مستوى معين، عندئذ يجب إضافة حالة استخدام لإنذار الطالب بضرورة تحسين مستواه.

إن عملية تحديد حالات الاستخدام وتوثيقها تتطلب عقد اجتماعات عديدة مع المستخدم حيث عادة ما يجد المستخدم أن الأحداث الخارجية تعتبر طريقة مفيدة للتفكير في النظام المراد تطويره. وأحياناً يستطيع بعض المستخدمين تسمية حالات الاستخدام أو وصفها بأنفسهم، عندئذ يجب عليك توثيق الأحداث المسببة لتلك الحالات. وأحياناً يعرض بعض المستخدمين سيناريوهات الحالة بشكل منفصل، عندئذ يجب عليك ربطها داخل حالة استخدام واحدة مع توثيق الحدث المسبب لها. وحاول أن تجعل المستخدمين يركزون دوماً على الأحداث لأن الأحداث الزمنية وأحداث تغير الحالة قد تكون غير واضحة والأحداث الخارجية ربما تخص بعض المستخدمين دون آخرين.

أثناء مرحلة تحديد حالات الاستخدام وتوثيقها يجب على فريق العمل إنشاء نماذج Use Cases حيث من الممكن أن يوضع نموذج واحد لجميع حالات استخدام النظام وجميع المستخدمين. وإذا كان حجم النظام كبيراً، فيجوز إنشاء نموذج لكل نظام فرعي على حده. وأحياناً أخرى يمكن تقسيم إنشاء نماذج Use Cases طبقاً للمستخدمين وذلك مفيد جداً عند مراجعة متطلبات النظام مع المستخدمين.

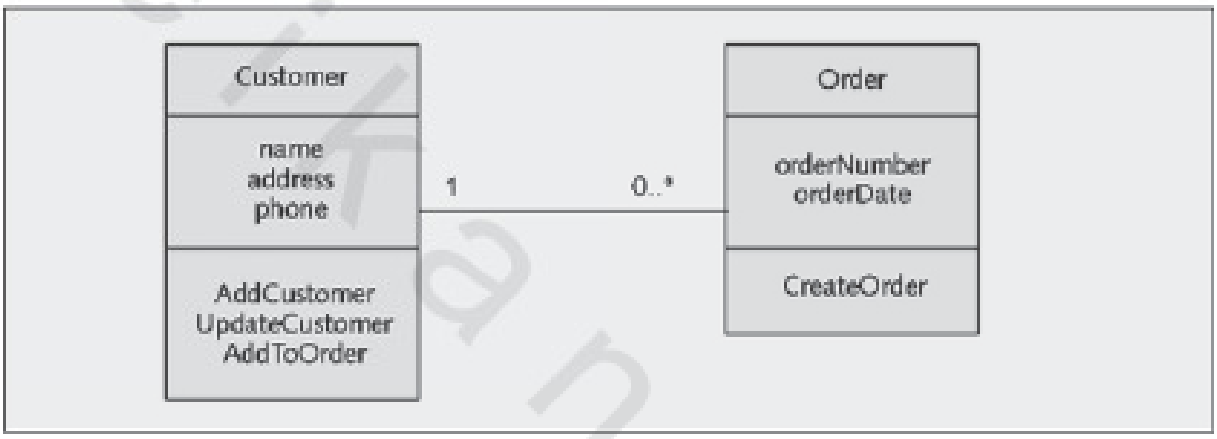
بعد ذلك يجب توثيق كل حالة استخدام بالتفصيل، وكما ذكرنا سابقاً أن أحد الطرق المستخدمة لتوثيق حالات الاستخدام هو سرد الخطوات التي يجب أن يتبعها المستخدم لكي يستخدم النظام. هذه الخطوات تتضمن تفاعلاً بين المستخدم والنظام على هيئة حوار. ولقد عرّفت لغة UML نموذجاً إضافياً لتوثيق حالات الاستخدام أطلقت عليه اسم نموذج التفاعل (Activity Diagram)، وأحياناً ننشئ هذا النموذج مع كل سيناريو خاص بحالة استخدام معينة كما يجب أن نعلم أن كمية التفاصيل وعدد النماذج يعتمد على مدى تعقيد حالات استخدام النظام.

إنشاء نموذج الأصناف Class Diagram وتوضيحه
Creating Interpreting the Class Diagram

كما يشير اسم تقنية التطوير المستخدمة في هذا الكتاب (تطوير النظم المعتمدة على الكائنات)، فإن كل شيء في النظام ما هو إلا كائن (Object) وكل مجموعة كائنات ما هي إلا أمثلة من صنف (Class) محدد؛ ولذلك فإن نموذج توصيف أصناف النظام يعتبر أهم نماذج توصيف النظام والذي نطلق عليه Class Diagram حيث يتم تمثيل الصنف بمستطيل مقسم إلى ثلاث أجزاء (انظر إلى الشكل رقم ٥.٥). الجزء العلوي يحتوي على اسم الصنف، والجزء الأوسط يحتوي على صفات الصنف، بينما الجزء السفلي يحتوي على إجراءات الصنف. يوضح الشكل رقم (٥.٥) الصنف Customer (عميل) الذي يحتوي على ثلاث صفات (name و address و phone)، ويحتوي أيضاً على ثلاثة إجراءات (AddCustomer و UpdateCustomer و AddToOrder).

كما يوضح الشكل أيضاً الصنف Order (طلب)، ويوضح علاقة الترابط (Association Relationship) التي تربط الصنف Customer بالصنف Order حيث يتم تمثيل العلاقة بخط واصل بين الصنفين. ولأن العميل يمكن أن يطلب

أكثر من طلب بينما الطلب الواحد يتم طلبه بواسطة عميل واحد فقط ؛ لذلك فإن العدد المدون أعلى الخط عند النهايتين يوضح ذلك حيث تشير لغة UML إلى هذه الأعداد باسم Multiplicity (نفس هذا المفهوم موجود في قواعد البيانات باسم Cardinality). ولاحظ أن النجمة (*) تعني العديد ولذلك فإن *..٠ تعني أن العميل يستطيع أن يطلب صفر أو أكثر من الطلبات وأحياناً يتم تدوين هذا المعنى بنجمة فقط (*). أما العلاقة الإجبارية (Mandatory) فنرمز لها بالرمز (*..١) أي واحد أو أكثر وذلك يعني أنه لا يضاف العميل إلى النظام حتى يطلب على الأقل طلباً واحداً. وللفهم الصحيح للعلاقة ، تأكد من قراءة العلاقة في كلتا الاتجاهين (من اليمين إلى اليسار ومن اليسار إلى اليمين).

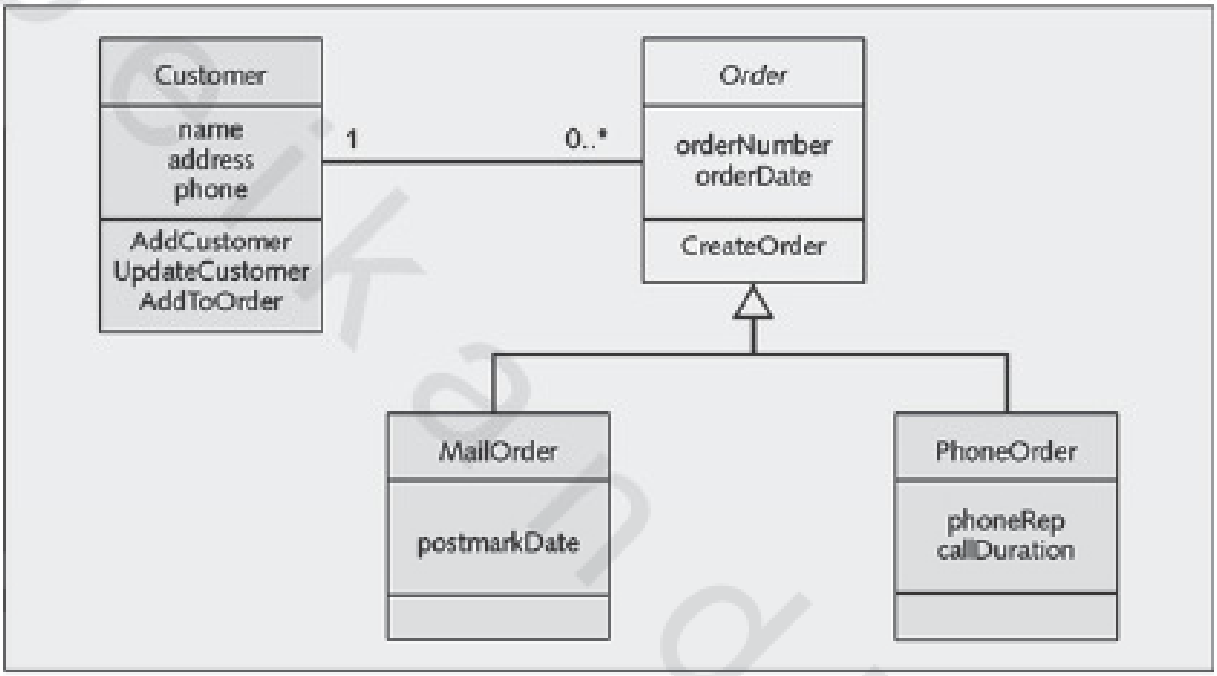


الشكل رقم (٥,٥). مثال لنموذج Class Diagram.

كما يوضح نموذج Class Diagram علاقة التوارث (Inheritance) بين الأصناف والتي تسمى أيضاً هرمية التعميم والتخصيص (Generalization/Specialization Hierarchy). يوضح الشكل رقم (٥,٦) نموذج Class Diagram بعد إضافة أصناف فرعية للصنف Order : الصنف MailOrder والصنف PhoneOrder حيث يرمز المثلث أسفل الصنف Order إلى وجود أصناف فرعية ترث صفات الصنف Order وإجراءاته، كما يمكن أن تشمل الأصناف الفرعية صفات وإجراءات خاصة بها، فعلى سبيل المثال الصنف PhoneOrder يرث صفات وإجراءات من الصنف Order، كما يحتوي أيضاً على صفات خاصة به (phoneRep و callDuration) والصنف MailOrder يرث صفات وإجراءات الصنف Order ويحتوي على الصفة (postmarkDate). وبذلك فإن كائن الصنف MailOrder يحتوي على قيم لثلاث صفات (order number و order date و postmark date)، بينما كائن الصنف PhoneOrder سيحتوي على قيم أربع صفات (order number و order date و phone order clerk و call duration).

لاحظ أن اسم الصنف Order مدون بخط مائل بما يشير إلى أنه صنف تجريدي (Abstract Class) والذي لا يمكن أن ننشئ كائنات منه ولكن يستخدم فقط لتجميع الصفات والإجراءات المشتركة للأصناف الفرعية له. وبذلك

فإن جميع الطلبات (الكائنات) التي يمكن أن يطلبها العميل ستكون من النوع MailOrder أو من النوع PhoneOrder، ولاحظ أيضاً أن جميع الطلبات لابد أن تكون مرتبطة بكائن من النوع Customer. إن جميع كائنات كل من الصنف MailOrder والصنف PhoneOrder ترث صفات الصنف Order وإجراءاته، كما ترث أيضاً علاقات هذا الصنف (علاقته مع الصنف Customer).



الشكل رقم (٥,٦). نموذج Class Diagram الموسع ليعرض التوارث.

سوف تطلع على العديد من نماذج Class Diagram خلال هذا الكتاب. عادة يتم إنشاء هذه النماذج أثناء تحديد حالات استخدام النظام وتوثيقها حيث يتم استخدام بعض أصناف النظام التي يتفاعل معها المستخدم في توثيق حالات الاستخدام ويتم استكشاف أصناف جديدة خلال التحدث مع المستخدم ومن ثم تحديد حالات استخدام جديدة وتوثيقها وهكذا (تذكر أن هذه العملية تكرارية).

يجب أن تعلم عزيزي القارئ أن نماذج Class Diagram يتم تطويرها أثناء كل من مرحلة التحليل ومرحلة التصميم وتصبح أكثر تفصيلاً مع الوقت؛ ولذلك يوضح نموذج Class Diagram المبدئي عادة الصفات والإجراءات الأساسية فقط، وأثناء تطور النموذج يضاف إليه المزيد من الصفات والإجراءات. وكما تعلمنا سابقاً أن تطوير النظم المعتمد على الكائنات يعمل جيداً مع أسلوب التطوير التكراري (Iterative Approach)؛ وذلك لأن نموذج Class Diagram يتطور أثناء تطوير المشروع.

ولقد غطينا في هذه الفقرة الرموز والتركيبات الأساسية لنموذج Class Diagram مثل الأصناف وعلاقات الارتباط وعلاقة التوارث، ولكن يبقى العديد من الرموز والاصطلاحات التي لم تناقش.

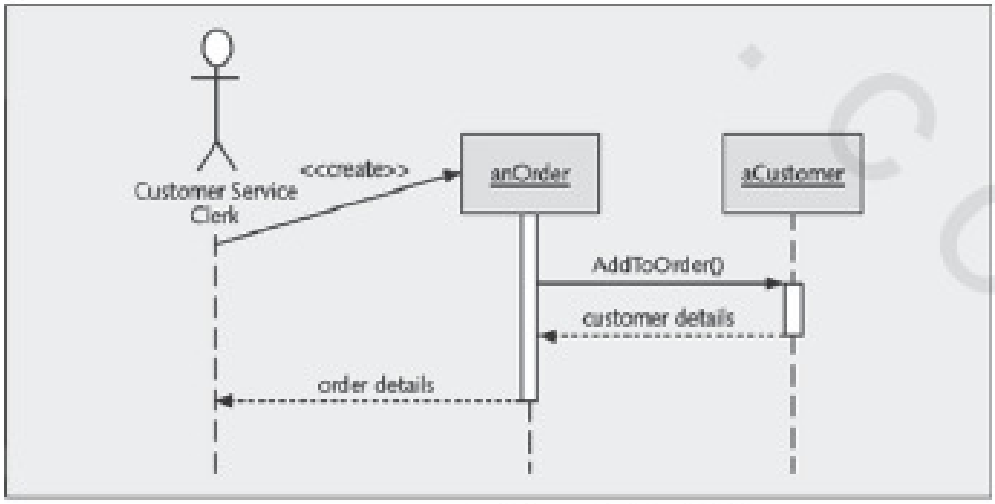
إنشاء نموذج Sequence Diagram وتوضيحه

Creating and Interpreting a Sequence Diagram

يوضح نموذج Sequence Diagram التفاعلات التي تحدث بين كائنات النظام لتنفيذ حالة استخدام (Use Case) أو لسيناريو؛ ولذلك يعتبر نموذج Sequence Diagram طريقة أخرى لوصف حالات الاستخدام، ولأن مهام النظام المعتمد على الكائنات يتم إنجازها من خلال التفاعلات التي تحدث بين كائنات النظام، فإن نموذج Sequence Diagram يصور هذه التعاملات ومن ثم يطلق عليه اسم النموذج التفاعلي (Dynamic Model). أما نموذج Class Diagram فيركز على الأصناف فقط ولا يصور هذه التفاعلات فيطلق عليه اسم النموذج الساكن (Static Model).

يظهر المستخدم على شكل عصا (ربما يظهر على شكل مستطيل)، وتظهر كائنات النظام على شكل مستطيل. يطلق على الخطوط الرأسية التي تظهر أسفل كل من المستخدم والكائنات اسم خطوط الحياة (Lifelines) والتي تمثل الترتيب الزمني. وإذا ظهر خط الحياة على شكل خط متقطع Dashed Line فذلك يعني أن الكائن في هذه المرحلة غير فعال Inactive، أما إذا ظهر على شكل مستطيل، فإن ذلك يعني أن الكائن فعال Active Object، أي في حالة تنفيذ (في حالة تفاعل). أما الخطوط الأفقية فتمثل رسائل مرسلة أو مستقبلة بالترتيب وتظهر البيانات العائدة نتيجة إرسال رسالة على شكل خط أفقي متقطع.

يوضح الشكل رقم (٥,٧) مثالاً لنموذج Sequence Diagram للسيناريو الذي تم مناقشته سابقاً والخاص بتسجيل حركة مبيعات لعميل معرف مسبقاً.



الشكل رقم (٥,٧). مثال لنموذج Sequence Diagram.

ولتوضيح تفاعلات الكائنات في نموذج Sequence Diagram، تعال سوياً نتتبع الشكل رقم (٥,٧) حيث يبدأ التفاعل عندما يرسل المستخدم Customer Service Clerk رسالة create إلى الصنف Order لإنشاء كائن جديد من هذا الصنف اسمه anOrder، عندئذ ينشئ هذا الكائن ويصبح فعالاً مباشرة (لاحظ أن خط الحياة يظهر بشكل مستطيل) والذي يرسل رسالة AddToOrder لكائن موجود من النوع Customer اسمه aCustomer لإضافة نفسه إلى الطلب الجديد. لاحظ أن الرسالة create تشاور مباشرة للكائن anOrder بما يفيد أنه كائن جديد، أما الرسالة AddToOrder تشاور إلى خط الحياة مباشرة مما يفيد أن الكائن موجود قبل ذلك ولكن يصبح فعالاً عند استقبال الرسالة، وبعد ذلك يعيد الكائن aCustomer بيانات للكائن anOrder لكي ينهي المعالجة الخاصة به، وأخيراً يعيد الكائن anOrder معلومات حول الطلب الجديد للمستخدم Customer Service Clerk ثم يصبح كائناً غير فعال.

في الحقيقة توجد العديد من الأساليب لتسمية كائنات نماذج Sequence Diagram حيث عادة يوجد خط أسفل اسم الكائن ويبدأ بحرف صغير Small، بينما اسم الصنف يبدأ بحرف كبير Capital.

يستخدم الشكل رقم (٥,٧) أسماء عامة في تسمية الكائنات لتوضيح أصناف هذه الكائنات، فعلى سبيل المثال الكائن anOrder يمثل اسماً عاماً لكائن من النوع Order حيث يبدأ بأداة نكرة وبحرف صغير Small ثم يشتمل على اسم الصنف لتوضيح نوعه، وبالمثل الكائن aCustomer يمثل اسم كائن عام. توجد طريقة أخرى في التسمية وهي تسمية الكائن بأي اسم متبوعاً باسم الصنف وبينهما الرمز ":"، فمثلاً كائنات الشكل رقم (٥,٧) يمكن إعادة تسميتها هكذا anOrder:Order و aCustomer:Customer (يجب وضع خط أسفل الأسماء مثل الأسلوب الأول). هذا الأسلوب في التسمية يفضل استخدامه عندما تكون أسماء الكائنات خاصة بمجال مشكلة النظام فعلى سبيل المثال: appollo13:Spacecraft، أو lisa:Editor، أو susan:Professor.

ربما توجد هناك حاجة لاستدعاء إجراء من صنف مباشرة (وليس من كائن)، فعندئذ نستخدم اسم الصنف مباشرة داخل نموذج Sequence Diagram ويدخل في التفاعل لإتمام مهمة ما حيث أحياناً يحتوي الصنف على إجراءات وصفات من النوع Shared (أي إجراءات وصفات تستدعى من صنف وليس من كائن).

وكما هو ملاحظ من الشكل رقم (٥,٧)، تكتب أسماء الرسائل (الإجراءات) فوق خطوط الرسائل. الرسالة <<create>> المرسلة للكائن anOrder تمثل استدعاء حدث فريد لإنشاء كائن من النوع Order والذي يختلف برمجته باختلاف لغة التطوير المستخدمة.

أما الرسائل الأخرى فما هي إلا استدعاء لإجراءات معرفة داخل أصناف الكائنات مثل الإجراء AddToOrder وهو إجراء معرف داخل الصنف Customer تم استدعاؤه بواسطة الرسالة ()AddToOrder من الكائن Order، وأخيراً يُسند اسم للبيانات العائدة من الكائن نتيجة تنفيذ إجراء معين ويتم وضعه فوق الخط المتقطع مثل customer details وهي معلومات العميل بواسطة الكائن aCustomer ومثل order details العائدة من الكائن anOrder.

تستخدم نماذج Sequence Diagram في أوقات مختلفة وبطرق مختلفة حيث يمكن تطويرها في بداية المشروع لتوثيق حالات الاستخدام والسيناريوهات المختلفة، وذلك مثل نماذج Class Diagram والتي يتم تصميمها في البداية بقليل من التفاصيل ثم إضافة جميع التفاصيل في مراحل متأخرة. وأحياناً أخرى، يفضل بعض المطورين تصميم نماذج Sequence Diagram في مراحل متأخرة من تطوير النظام. وكما رأينا سابقاً، يظهر المستخدم في نموذج Sequence Diagram ولكن في بعض الأحيان لا يظهر المستخدم وعندئذ تبدأ التفاعلات بواسطة صنف أو كائن ما. تذكر عزيزي القارئ أن المستخدم (Actor) ربما يكون إنساناً أو نظاماً فرعياً آخر أو جهازاً.

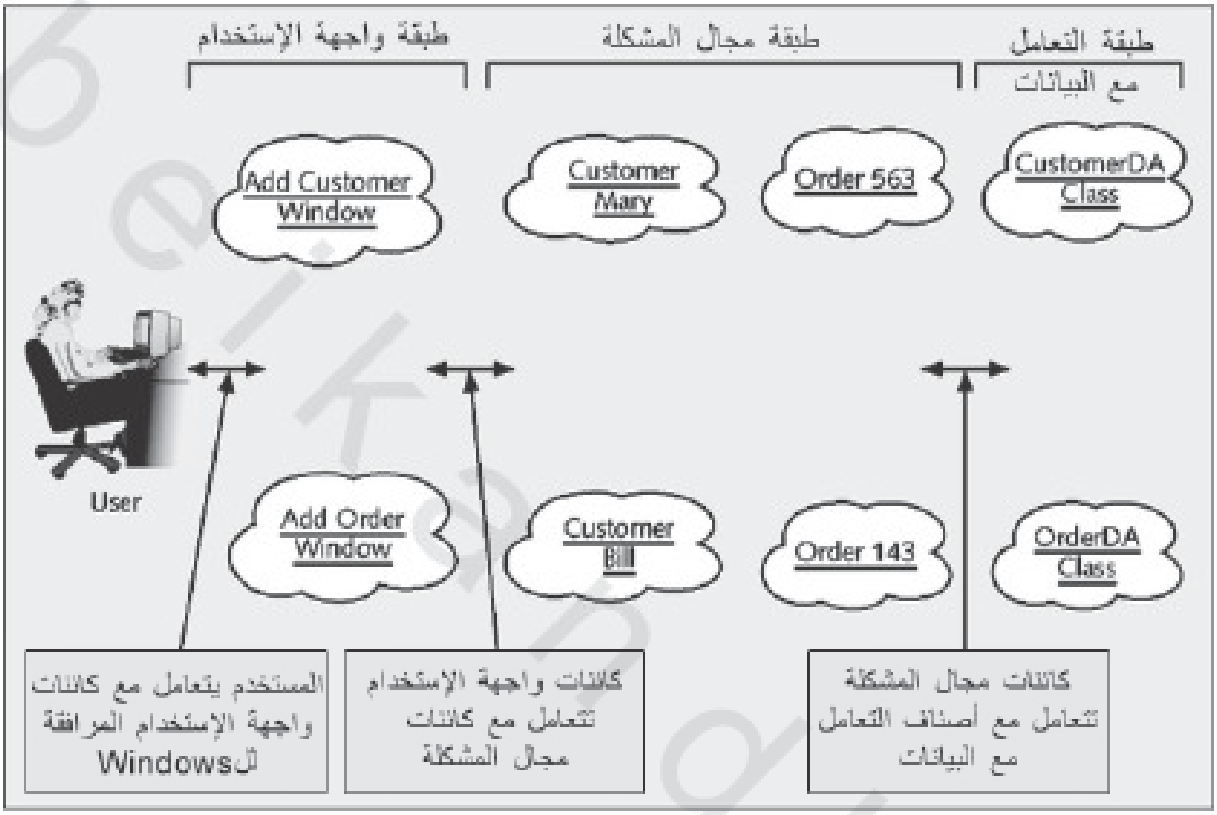
استخدام أسلوب التصميم ثلاثي الطبقات في تطوير النظم المعتمدة على الكائنات
Using Three-Tier Design in OO Development

من أهم مميزات التحليل والتصميم باستخدام لغة UML (عن الطريقة التقليدية) هو استخدام نفس النماذج في كل من مرحلة التحليل والتصميم وذلك على عكس الطريقة التقليدية التي تستخدم نماذج DFD ونماذج ERD خلال مرحلة التحليل وتستخدم نماذج تركيبية أخرى خلال مرحلة التصميم. وهذه تعتبر أهم مميزات تطوير النظم المعتمدة على الكائنات حيث تستخدم نفس النماذج خلال المراحل المختلفة تطوير النظم.

لقد قدمنا في الفصل الأول أسلوب ثلاثي الطبقات (Three-Tier) والذي من خلاله تستطيع التمييز بين مرحلة التحليل OOA ومرحلة التصميم OOD حيث يحتوي هذا الأسلوب على ثلاث أنواع من الأصناف: أصناف واجهة النظام (GUI Classes)، وأصناف مجال المشكلة (Problem Domain Classes)، وأصناف التعامل مع البيانات (Data Access Classes)، وذلك يتطلب من مطوري النظم المعتمدة على الكائنات تقسيم أصناف النظام إلى ثلاث طبقات أثناء تصميم النظام وإنشائه. أولاً: أصناف مجال المشكلة وهي أصناف الكائنات المتعلقة بمجال عمل المستخدم. ثانياً: تعريف أصناف واجهة المستخدم وهي الأصناف التي تسمح للمستخدم التعامل مع أصناف مجال المشكلة. وأخيراً: أصناف التعامل مع البيانات وهي الأصناف التي تجعل أصناف مجال المشكلة تتفاعل مع قواعد البيانات. وفي حالة الانتهاء من تطوير الطبقات الثلاث بشكل منفصل، يتكون النظام بواسطة التفاعل بينها.

يوضح الشكل رقم (٥,٨) أسلوب ثلاثي الطبقات لنظام معالجة الطلبات للعملاء (مثال الفصل الأول نفسه) حيث يتفاعل المستخدم مع واجهة الاستخدام والتي تتكون من كائنات رسومية مثل القوائم (Menus)، والأزرار (Buttons)، والصناديق النصية (Text Boxes). وذلك بالضغط على الزر الأيسر للفأرة مثلاً أو الضغط على أحد مفاتيح لوحة المفاتيح ليستجيب النظام لهذه الأحداث. لاحظ أن المستخدم لا يتفاعل مباشرة مع كائنات مجال المشكلة ولكن يتفاعل مع كائنات واجهة الاستخدام التي بدورها تتفاعل مع كائنات مجال المشكلة بناء على أحداث

المستخدم. ويفصل أصناف واجهة الاستخدام عن أصناف مجال المشكلة، يمكن التركيز على أصناف مجال المشكلة بشكل مستقل عن أصناف واجهة الاستخدام.



الشكل رقم (٨،٥). الطبقات الثلاثة في التصميم ثلاثي الطبقات.

عندما يتم إنشاء كائن من أصناف مجال المشكلة، فيتطلب ذلك إنشاء إجراء لتخزين معلومات هذا الكائن في قاعدة بيانات النظام لاسترجاعه لاحقاً. إن إجراءات المعالجة المسؤولة عن تخزين معلومات كائنات أصناف مجال المشكلة واسترجاعها يتم عزلها ووضعها في طبقة أخرى من الأصناف وهي أصناف التعامل مع البيانات (يتم تخصيص صنف من هذا النوع لكل صنف من أصناف مجال المشكلة). وتوجد ثلاثة طرق لتخزين بيانات كائنات مجال المشكلة (كما سترى في كل من الفصل الثالث عشر والرابع عشر)، ويفضل إجراءات معالجة قواعد البيانات، تستطيع أن تركز فقط على أصناف مجال المشكلة بشكل مستقل عن قواعد البيانات.

إن فصل أصناف النظام إلى ثلاث طبقات (أصناف واجهة الاستخدام، وأصناف مجال المشكلة، وأصناف التعامل مع البيانات) يدعم فكرة إنشاء النظام المكون من مكونات ليست وثيقة الصلة (Loosely Coupled System Components)، وهذا مفيد جداً لأنك تستطيع تعديل أحد المكونات دون تأثير يذكر على المكونات الأخرى. فعلى

سبيل المثال تغيير قاعدة بيانات النظام يتطلب فقط تغيير أصناف التعامل مع البيانات دون تأثير على كل من أصناف واجهة الاستخدام وأصناف مجال المشكلة، وبالمثل تغيير واجهة النظام يتطلب تغيير أصناف واجهة الاستخدام ولا يتطلب تغيير أصناف التعامل مع البيانات وأصناف مجال المشكلة. إن التطوير مستخدماً أسلوب ثلاثي الطبقات يجعل عملية صيانة النظام وتحسينه سهلة وبسيطة، هذا بالإضافة إلى قاعدة إعادة استخدام المكونات في نظم أخرى والتي تعتبر أهم أهداف تطوير النظم المعتمدة على الكائنات.

إن أسلوب التصميم ثلاثي الطبقات يقدم أيضاً إطار عمل لكل من مرحلة التحليل ومرحلة التصميم حيث أن مرحلة تحليل النظم تتعلق فقط بتعريف أصناف مجال المشكلة وتوثيقها كما هو واضح في نموذج Class Diagram السابق الذي يحتوي على الصنف Customer والصنف Order وواضح أيضاً في نموذج Seqence Diagram السابق الذي يحتوي على كائنات أصناف مجال المشكلة (الكائن anOrder، والكائن aCustomer). عندما نعرف كيفية إنشاء نماذج لأصناف مجال المشكلة والتفاعلات بين كائناتها عندئذ تكون قد أنشأت ما يسمى بالنماذج المنطقية (Logical Models) لمتطلبات النظام والذي يمثل الهدف الرئيس لمرحلة تحليل النظم. وتسمى هذه النماذج بالنماذج المنطقية لأنها توضح المعالجة المطلوبة فقط دون توضيح كيفية برمجتها.

وعند الانتقال إلى مرحلة تصميم النظم يتطلب ذلك اتخاذ بعض القرارات المتعلقة بواجهة الاستخدام وقاعدة بيانات النظام، ومن ثم إضافة أصناف واجهة الاستخدام وأصناف التعامل مع البيانات إلى نماذج مرحلة التحليل وإنشاء نماذج أخرى أكثر تعقيداً تسمى النماذج المادية (Physical Models) والتي توضح كيفية تطوير النظام.

يعرض الشكل رقم (٥,٩) أحد النماذج المادية الذي يوضح كيفية تفاعل المستخدم مع النظام وكيفية تفاعل النظام مع قاعدة البيانات كالتالي:

١- يكتب المستخدم معلومات الطلب الجديد مستخدماً شاشة addOrderWindow (كائن من كائنات واجهة الاستخدام)، والتي تحتوي على صناديق نصية وعناوين وأزرار.

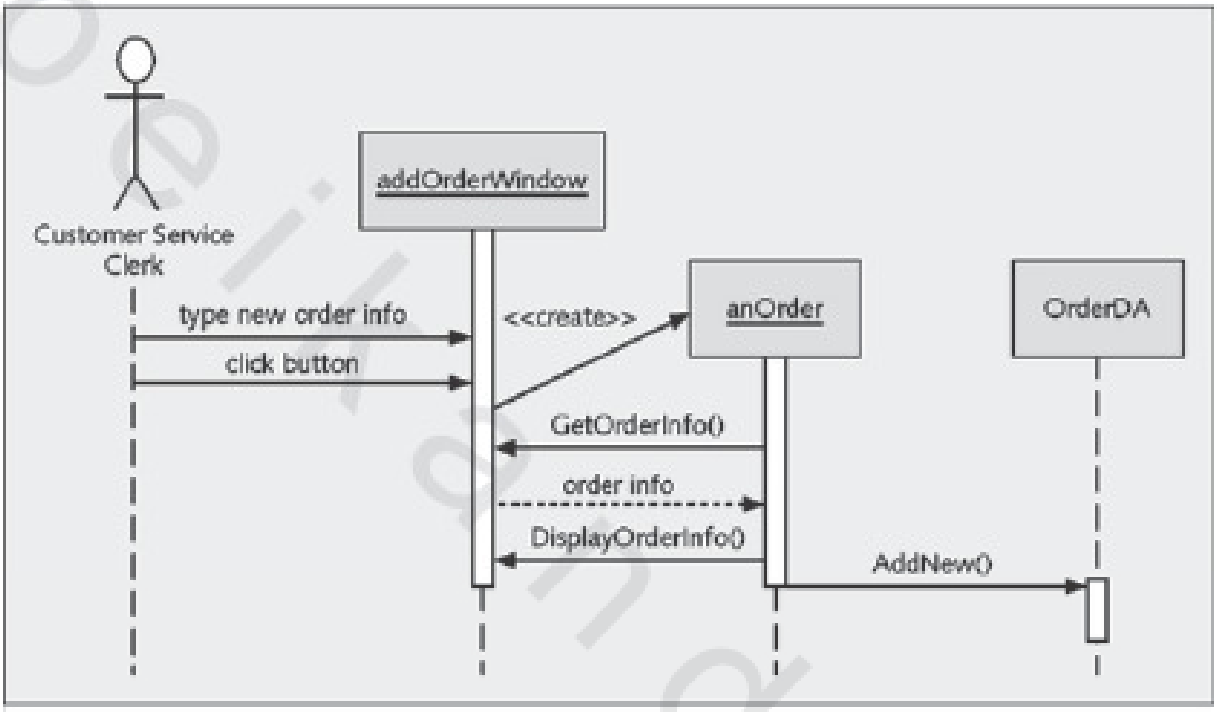
٢- ثم يضغط المستخدم على أزرار الشاشة addOrderWindow طالباً من النظام إضافة الطلب.

٣- يرسل الكائن addOrderWindow رسالة <<create>> إلى الصنف Order لإنشاء طلب جديد (كائن anOrder).

٤- يطلب الكائن anOrder المعلومات التي أدخلها المستخدم من الكائن addOrderWindow مرسلاً الرسالة GetOrderInfo وعندئذ يستجيب الكائن addOrderWindow مرسلاً الرسالة order info (معلومات الطلب).

٥- يكمل الكائن anOrder المعالجة المطلوبة ويرسل رسالة للكائن addOrderWindow لعرض تفاصيل بيانات الطلب لكي يتمكن المستخدم من مراجعتها وذلك باستدعاء الإجراء DisplayOrderInfo.

٦- وأخيراً يرسل الكائن anOrder رسالة إلى الصنف OrderDA (الصنف المسؤول عن تخزين كائنات الصنف Order وصيانتها) طالباً منه إضافة طلب جديد لقاعدة البيانات، عندئذ يعتني الصنف OrderDA بكل التفاعلات المطلوبة مع قاعدة البيانات لتخزين طلباً جديداً.



الشكل رقم (٥,٩). نموذج Sequence Diagram مع إضافة كائنات واجهة الاستخدام وأصناف التعامل مع البيانات.

يعمل الأسلوب ثلاثي الطبقات جيداً مع كل من تطوير النظم التكرارية وتطوير النظم التزايدية حيث يمكن زيادة بعض حالات الاستخدام Use Cases في كل دورة تطوير. أولاً، يمكنك تعريف أصناف مجال المشكلة الهامة وتوصيفها فقط بحالات الاستخدام المحددة، ثم إنشاء نماذج Sequence Diagram ولكل حالة استخدام أمر سيناريو موضحاً فقط كائنات أصناف مجال المشكلة. بعد ذلك يمكنك كتابة البرنامج المطلوب لتعريف أصناف مجال المشكلة ثم القيام باختبارها.

بعد ذلك يمكنك أن تعرف أصناف واجهة المستخدم لكل حالة استخدام أو سيناريو ثم توسعة كل Sequence Diagram لكي يشمل على أصناف واجهة المستخدم، ثم تطوير أصناف واجهة الاستخدام واختبارها وتكاملها مع أصناف مجال المشكلة. الآن قد تم أثناء الدورة الأولى إنتاج أول جيل من النظام (نموذج أولي) للتقييم بواسطة المستخدمين.

بعد تطوير أصناف واجهة الاستخدام، يمكنك الآن كتابة أصناف التعامل مع قواعد البيانات وذلك بتخصيص صنف من هذا النوع لكل صنف من أصناف مجال مشكلة لتخزين وصيانة كائناته. أيضاً يمكن إضافة أصناف لنماذج Sequence Diagram ثم اختبار وتكامل أصناف التعامل مع قواعد البيانات مع أصناف مجال المشكلة. وأخيراً إن كلاً من أصناف واجهة الاستخدام، وأصناف مجال المشكلة، وأصناف قواعد البيانات جاهزة للعمل سوياً في أول دورة.

يتم إعادة العملية نفسها خلال الدورة التالية لزيادة قليل من حالات الاستخدام الجديدة. لاحظ أن أسلوب ثلاثي الطبقات يعمل جيداً مع تطوير النظم التكرارية لأنك تستطيع أن تنتقل بسهولة من التحليل إلى التصميم إلى البرمجة أثناء كل دورة لكي يتم بناؤه دائماً فوق النماذج السابقة. أيضاً يستطيع المستخدم تقييم أجيال النظام ومكونات النظام تظل مستقلة بقدر الإمكان.

يتبع تنظيم هذا الكتاب الأسلوب ثلاثي الطبقات حيث يناقش الباب الثاني (بداية من الفصل القادم) أصناف مجال المشكلة وكيفية برمجتها بلغة VB .NET، ويناقش الباب الثالث أصناف واجهة الاستخدام وكيفية برمجة نماذج Windows بلغة VB .NET وكيفية برمجة نماذج الويب مستخدماً تقنية ASP .NET وشرح كيفية تفاعلها مع كل من أصناف مجال المشكلة والمستخدم. أما الباب الرابع فسيناقش أصناف التعامل مع قاعدة البيانات وتفاعلها مع أصناف مجال المشكلة، ثم ربط الأنواع الثلاث من الأصناف في تطبيق واحد وشرح كيفية تفاعلها سوياً في الباب الخامس حيث نذكر مثالاً لتطبيقات Windows ومثالاً آخر لتطبيقات الإنترنت.

التعرف على حالة الدراسة: شركة برادشو مارينا للسفن
Introducing the Bradshaw Marina Case Study

نتعرف في هذه الفقرة على حالة الدراسة الرئيسة التي ستبقى معنا خلال جميع أبواب الكتاب وهي احتياج شركة برادشو مارينا للسفن لنظام معلومات ومن ثم سوف نعرض كلاً من مفاهيم تطوير النظم المعتمدة على الكائنات وبرامج لغة VB .NET مستخدمين هذا المثال. ونعرض أيضاً في هذه الفقرة نماذج UML المطلوبة لتوصيف متطلبات النظام، ثم سنعرض لاحقاً نماذج UML إضافية لتوصيف بقية النظام مستخدمين لغة VB .NET.

عندما تدرك شركة ما احتياجها لنظام معلومات، فلابد أن تلجأ إلى فريق تطوير لكي يصمم هذا النظام ويطوره، وأول مهام هذا الفريق هو تحليل أعمال هذه الشركة وتحديد الوظائف التي سينجزها النظام. تصف الفقرة القادمة أعمال شركة برادشو مارينا للسفن ثم يبدأ فريق التطوير في مرحلة التحليل المعتمدة على الكائنات OOA لتحديد حالات الاستخدام والسيناريوهات المطلوبة ومن ثم سينشئ نماذج Use Cases.

وبعد ذلك يبدأ فريق التطوير في تعريف أصناف مجال المشكلة وإنشاء نموذج Class Diagram، وأخيراً يبدأ الفريق في توصيف التفاعل بين كائنات أصناف مجال المشكلة لإنجاز مهام النظام من خلال إنشاء نماذج Sequence Diagram. عزيزي القارئ تقبل دعوتي لتكون عضواً أساسياً في فريق تطوير نظام معلومات لشركة برادشو مارينا.

نبذة عن شركة برادشو مارينا للسفن
Exploring the Background of Bradshaw Marina

شركة برادشو مارينا للسفن هي شركة خاصة تؤجر مراسي القوارب لعملائها كما تقدم خدمات القوارب، وهذه الشركة تقع على بحيرة كلينتون (بحيرة كبيرة داخل الولايات المتحدة). وقد تم تصنيع هذه البحيرة خلال عقد السبعينات للتحكم في الفيضانات وتوليد كميات محدودة من الطاقة الكهربائية. مهندسو الجيش الأمريكي يديرون البحيرة ويمنعون إنشاء أي بناء قريب من شواطئها، ومن ثم خلقوا حياة برية مثالية بالإضافة إلى إنشاء منتزه رائع لمحبي السفن والقوارب. تعتبر شركة برادشو مارينا أكبر شركة من ثلاث شركات تعمل على بحيرة كلينتون حيث تمتلك الشركات الثلاث حوالي ٦٠٠ مركباً داخل المراسي (٤٥٠ مركباً شراعياً و١٥٠ مركباً آلياً). تمتلك شركة برادشو مارينا وحدها ٣٥٠ مركباً شراعياً و٧٥ مركباً آلياً، وبالرغم من ذلك فإن شركة برادشو مارينا تخطط لزيادة هذه الإمكانيات.

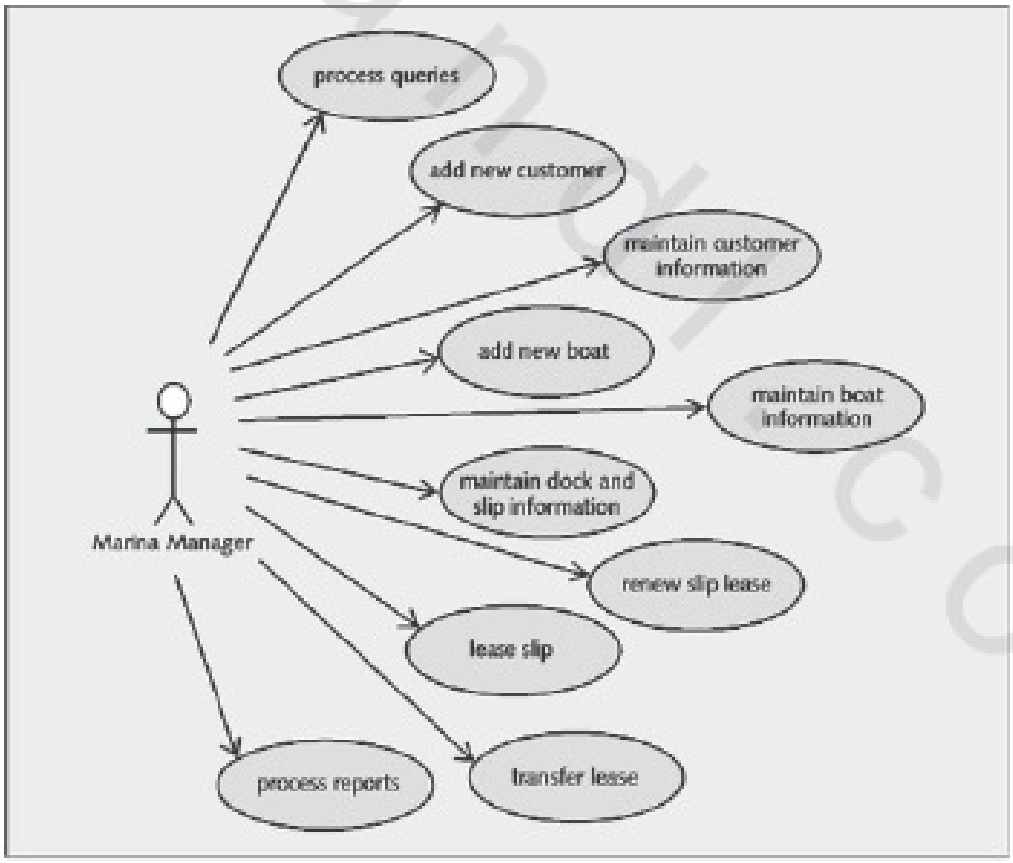
تريد شركة برادشو مارينا امتلاك نظاماً حاسوبياً لمعالجة مهام العملاء ومراسي المراكب والعقود والمراكب. بداية سوف يحتفظ النظام بالمعلومات الأساسية لكل من العملاء ومراسي المراكب والمراكب بالإضافة إلى إنجاز مهام الأعمال اليومية التي تشمل إنشاء عقد جديد وحساب قيمة إيجار مرسى معيناً وتخصيص مرسى لكل مركب. أيضاً تريد الشركة أن تستخدم النظام في البحث عن معلومات مثل المراسي غير الشاغرة والمراسي المؤجرة.

تريد الشركة تحسين النظام فيما بعد وإضافة بيانات خدمة المراكب التي تساعدها على متابعة المهام مثل تغيير اتجاه المركب ودهان قاع المركب وصيانة موتور المركب. هذا بالإضافة إلى زيادة إمكانية إصدار الفواتير مثل إنجاز فواتير عقود إيجار المراسي وخدمات المراكب وإيصالات المدفوعات وتقارير أخرى حول الخدمات.

تحديد حالات استخدام وسيناريوهات نظام برادشو
Identifying Bradshaw Use Cases and Scenarios

إن أول خطوة في عملية تحليل النظم المعتمدة على الكائنات هي تحديد حالات الاستخدام التي تقع داخل نطاق النظام. وبما أن أهم أحداث النظام هي الأحداث التي تتعلق بالعملاء مثل إيجار مرسى لأحد العملاء وحدث شراء مركبات ما لأحد العملاء وهكذا. ولأن هذه الأحداث تتعلق بالعملاء (Customers) والمراكب (Boats) والمراسي (Slips)

فإن حالات الاستخدام ستركز عليها، ولذلك سيخصص فريق التطوير الكثير من الوقت مع موظفي شركة برادشو مارينا للتحدث حول الأحداث التي تتعلق بالعملاء والتي ينتج عنها حالات استخدام. فعلى سبيل المثال إن حالات الاستخدام المتعلقة بالعملاء تشمل إضافة عميل جديد (add new customer)، وتعديل معلومات عميل (maintain customer information). إن حالة استخدام add new customer تنفذ عندما يوجد عميل جديد يريد أن يستأجر مرسى، وحالة الاستخدام maintain customer information تنفذ عندما يغير العميل معلوماته مثل عنوانه أو هاتفه. وبالمثل فإن حالات الاستخدام المتعلقة بعقود الإيجار تشمل إيجار مرسى (lease slip)، وتجديد عقد إيجار مرسى (renew slip lease)، ونقل عقد إيجار (transfer lease)، بينما حالات الاستخدام المتعلقة بالمراكب تشمل إضافة مركب جديد (add new boat)، وتعديل معلومات مركب (maintain customer boat). هذا بالإضافة إلى أن النظام لابد أن يحتوي على حالات استخدام لتعديل معلومات مراسي المراكب ومعلومات الأرصفة (Docks) التي تحتوي على مراسي المراكب. وأخيراً لابد أن يحتوي النظام على حالات استخدام معالجة الاستعلامات (process queries) ومعالجة التقارير (process reports). وفي نهاية عملك مع فريق التطوير سيتم إنشاء نموذج Use Case الموضح في الشكل رقم (٥,١٠).



الشكل رقم (٥,١٠). نموذج Use Case Diagram الخاص بشركة برادشو مارينا.

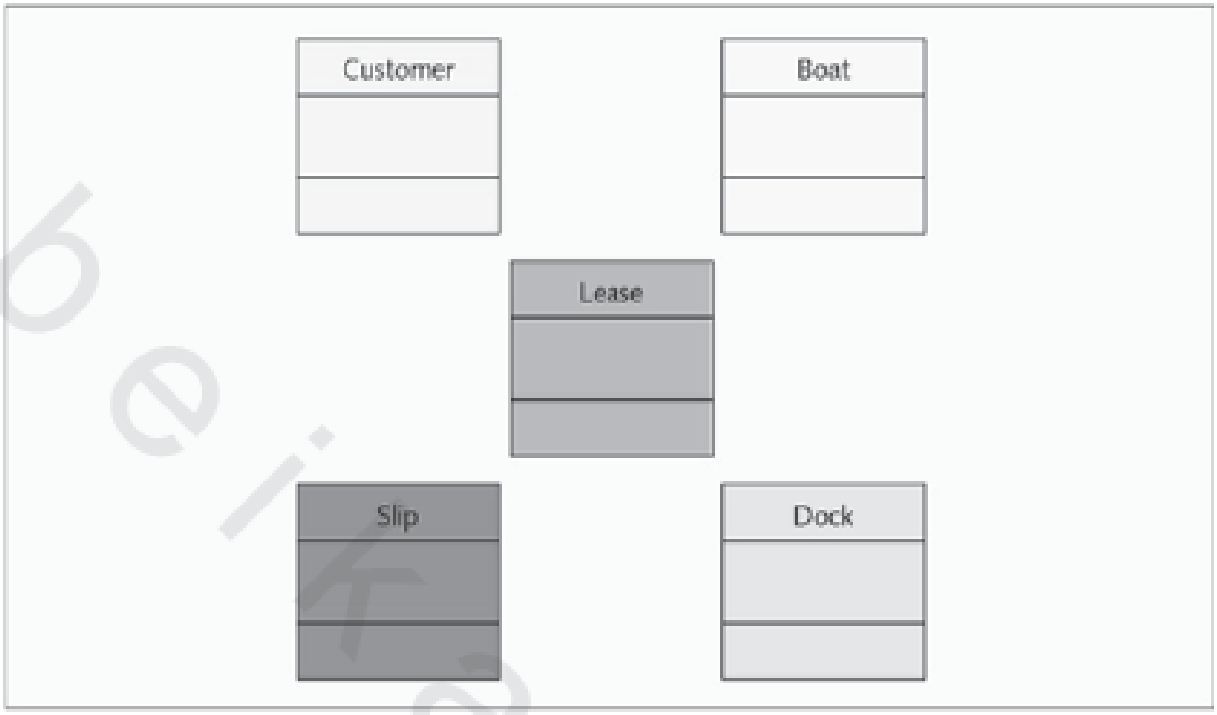
ربما يوجد أكثر من سيناريو لكل حالة استخدام، ولذلك فربما تقرر مع زملائك في فريق العمل تقسيم قائمة حالات الاستخدام إلى مجموعات والعمل عليها بشكل منفصل. فعلى سبيل المثال إن حالة الاستخدام lease slip (إيجار مرسى) ربما يوجد لها أكثر من سيناريو، ولمعرفة ذلك لابد من مناقشة أكثر مع موظفي شركة برادشو. أحد سيناريوهات حالة استخدام lease slip هو إيجار المرسى لعميل موجود (lease slip to existing customer) وسيناريو آخر هو إيجار المرسى لعميل جديد (lease slip to new customer). بل أكثر من ذلك، فربما يكون العقد سنوياً (lease annual slip to customer) أو عمل عقد يومي (lease daily slip to customer).

إن تحديد سيناريوهات حالات الاستخدام بدقة متناهية يتطلب وقتاً طويلاً حيث يمر العمل بالعديد من المحاولات لكي تستطيع أن تنشئ مجموعة سيناريوهات شاملة وغير متداخلة لحالات الاستخدام الهامة. فعلى سبيل المثال إذا أخذنا في الحسبان جميع احتمالات حالة استخدام lease slip لوجدنا أنه بالنسبة للعميل ربما يوجد عميل موجود أو عميل جديد وأيضاً يوجد نوعان من العقد (عقد سنوي، وعقد يومي)، وفيما يلي جميع السيناريوهات المحتملة لهذه الحالة:

- تأجير عقد سنوي لعميل موجود.
- تأجير عقد سنوي لعميل جديد.
- تأجير عقد يومي لعميل موجود.
- تأجير عقد يومي لعميل جديد.

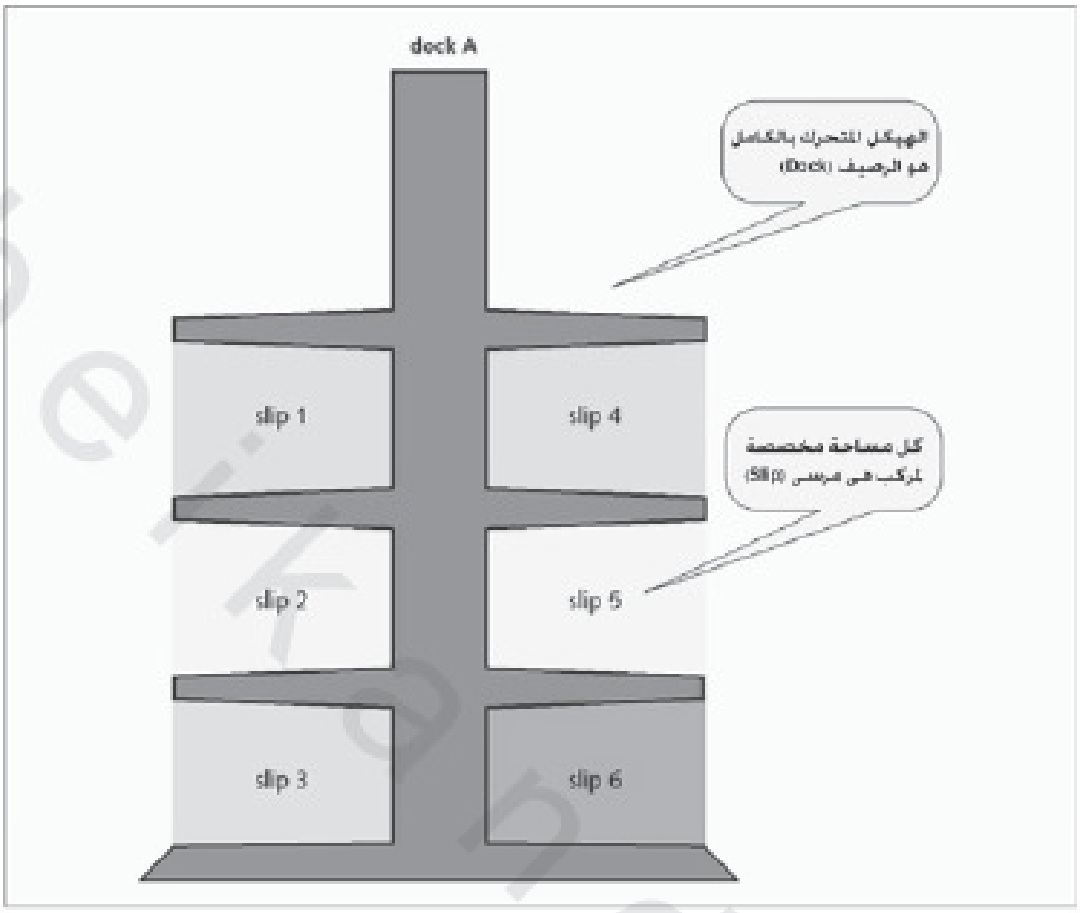
تحديد أصناف مجال مشكلة شركة برادشو
Identifying Bradshaw Problem Domain Classes

بعدما قمت أنت وزملائك في فريق التطوير بتحديد حالات الاستخدام والسيناريوهات المختلفة يجب عليكم أن تكتشفوا أصناف مجال المشكلة حيث من الضروري أن تتقابلوا مع فريق عمل شركة برادشو للاستفسار عن الأشياء المتعلقة بإنجاز مهام العمل والذي سيتضح أنهم العملاء (Customers)، والمراكب (Boats)، والعقود (Leases)، والمراسي (Slips)، والأرصفة (Docks)، وأول خطوة يجب أن تخطوها في هذا الخصوص هو إنشاء نموذج Class Diagram أولي والذي سيشتمل على الأصناف الهامة كما هو واضح في الشكل رقم (٥,١١).



الشكل رقم (٥,١١). نموذج Class Diagram المبدئي لشركة برادشو ماريا.

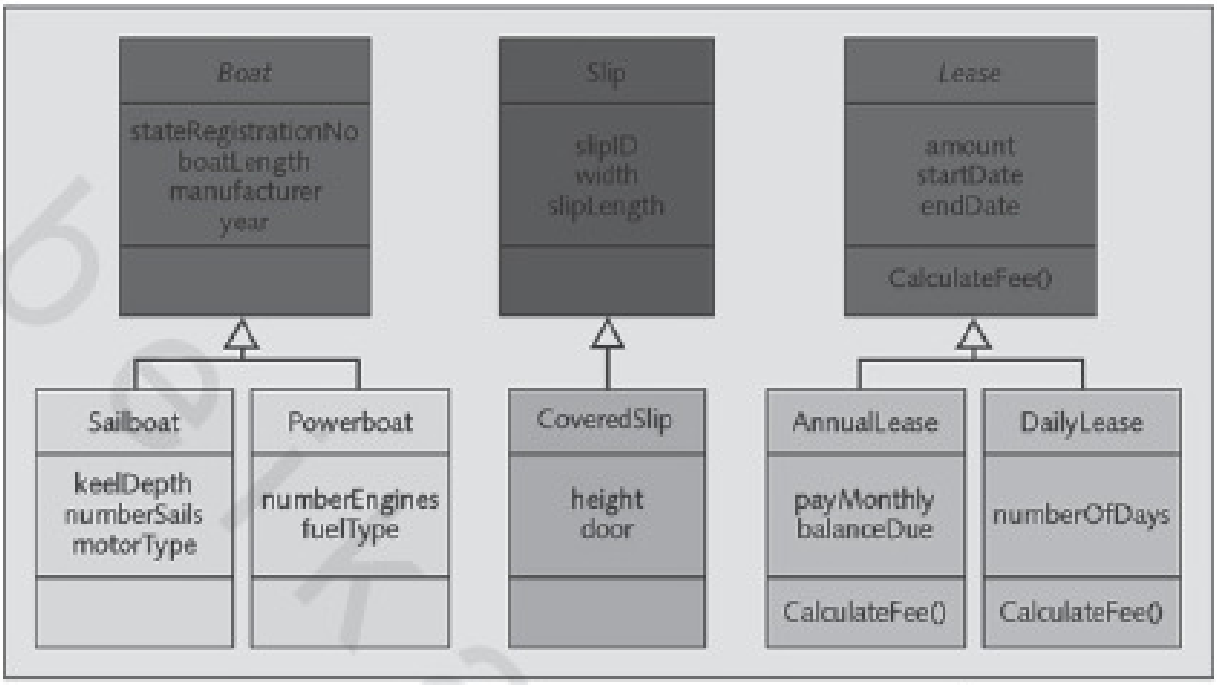
إن أول الأسئلة التي يجب أن يسألها فريق التطوير هو عن الفرق بين Docks (الأرصفة) و Slips (المراسي) لأن كثيراً من الأحيان يحدث سوء فهم عند استخدام الألفاظ ويجب أن يتحدث كل من مستخدمي النظام ومطوري النظام نفس اللغة ؛ لذلك من الضروري أن يتم تعريف المصطلحات، فمثلاً نسأل ماذا تعني بالضبط كلمة Dock، وماذا تعني بكلمة Slip. تذكر أن أصناف مجال المشكلة تعكس معرفة المستخدم التفصيلية حول طبيعة عمله. وتذكر أيضاً أن مهمتك هنا هو التعلم للحصول على معرفة المستخدم بقدر ما يمكن. وربما تسأل هل Slip هو نوع خاص من Dock (لرسم شجرة التوارث) أم هو شيء آخر. وبعد الحصول على التعريفات والرسومات التوضيحية من شركة برادشو تبين أن معنى Dock هو الرصيف العائم الذي يستخدمه العملاء للوصول إلى مراكبهم (انظر الشكل رقم ٥,١٢).



الشكل رقم (٥,١٢). الرصيف يحتوى على عدة مراسي.

أما Slip فيعني المرسى الذي يقف عليه المركب وهو الذي يؤجر للمستخدم، وبالتالي المرسى ليس نوعاً مخصوصاً من الرصيف بل هو شيء منفصل عن الرصيف ولكن يرافقه، وهذا المثال يوضح أهمية تعلم المعرفة المتوفرة لدى المستخدم حول عمله بواسطة فريق التطوير.

وبالرجوع إلى نموذج Class Diagram الأولي نلاحظ أننا نريد معلومات أكثر حول الصنف Boats، والصنف Slips، وكذلك الصنف Leases. فعلى سبيل المثال، شركة برادشو لديها نوعان من المراكب: المراكب الشراعية Sailboats، والمراكب الآلية Powerboats، كما يوجد لدى الشركة نوعان من المراسي: مرسى عادي Regular Slip، ومرسى مغطى Covered Slip. وأخيراً تقدم شركة برادشو نوعان من العقود: العقد السنوي Annual Lease، والعقد اليومي Daily Lease. من خلال هذه المعلومات نكتشف أن نموذج Class Diagram الأولي يحتاج إلى بعض التعديل لإضافة مفهوم التوارث حيث نضيف ما تسمى شجرة التعميم والتخصيص (Generalization/Specialization Hierarchies) كما هو واضح في الشكل رقم (٥,١٣).



الشكل رقم (٥,١٣). نموذج Class Diagram لشركة براندشو ماريا الذي يوضح التوارث.

لأن المراكب المتواجدة لدى الشركة إما أن تكون مركباً شراعياً أو مركباً آلياً؛ لذلك فإن الصنف Boat هو صنف مجرد حيث أنه موجود فقط لغرض التوارث (أي لاحتواء العناصر المشتركة فقط) (مكتوب بخط مائل). وكذلك الصنف Lease هو صنف مجرد لأن أي عقد يجب أن يكون يومياً أو سنوياً. أما الصنف Slip (المرسى) على الجانب الآخر ربما يكون عادياً أو مغطى، وبما أن المرسى المغطى هو نوع مخصوص من النوع العادي فإن الصنف Slip (العادي) ليس مجرداً بل واقعي (Concrete) ويمكن أن ننشئ منه كائنات.

ومع استمرارية توصيف أصناف مجال المشكلة ستحصل على معلومات تفصيلية ودقيقة حول كل صنف (الصفات Attributes). مثلاً جميع المراكب لها صفة Registration No. (رقم التسجيل)، وصفة Length (الطول)، وصفة Manufacturer (المصنع)، وصفة Model Year (سنة الموديل). أما المراكب الشراعية فلها صفات إضافية خاصة بها مثل Keel Depth (عمق السفينة)، وصفة Number of Sails (عدد الأشرعة)، وصفة Motor Type (نوع الموتور). أما المراكب الآلية فلها صفات إضافية مختلفة مثل Number of Engines (عدد المحركات)، وصفة Fuel Type (نوع الوقود).

ويطلق على كل من الصنف Sailboat والصنف Powerboat أصنافاً فرعية لأن كل منهما يرث صفات مشتركة ويتميز بصفات خاصة به.

أما الصنف Slip فيحتوي على الصفة SlipID وهو رقم تعريف المرسى، هذا بالإضافة إلى صفة Width (عرض المرسى)، وصفة Length (الطول). أما المراسي المغطاة (Covered) فلابد من معرفة صفة Height (الارتفاع)، وفي بعض الأحيان يمكن غلق المرسى المغطى لحماية المرسى ولذلك توجد الصفة Door (الباب).

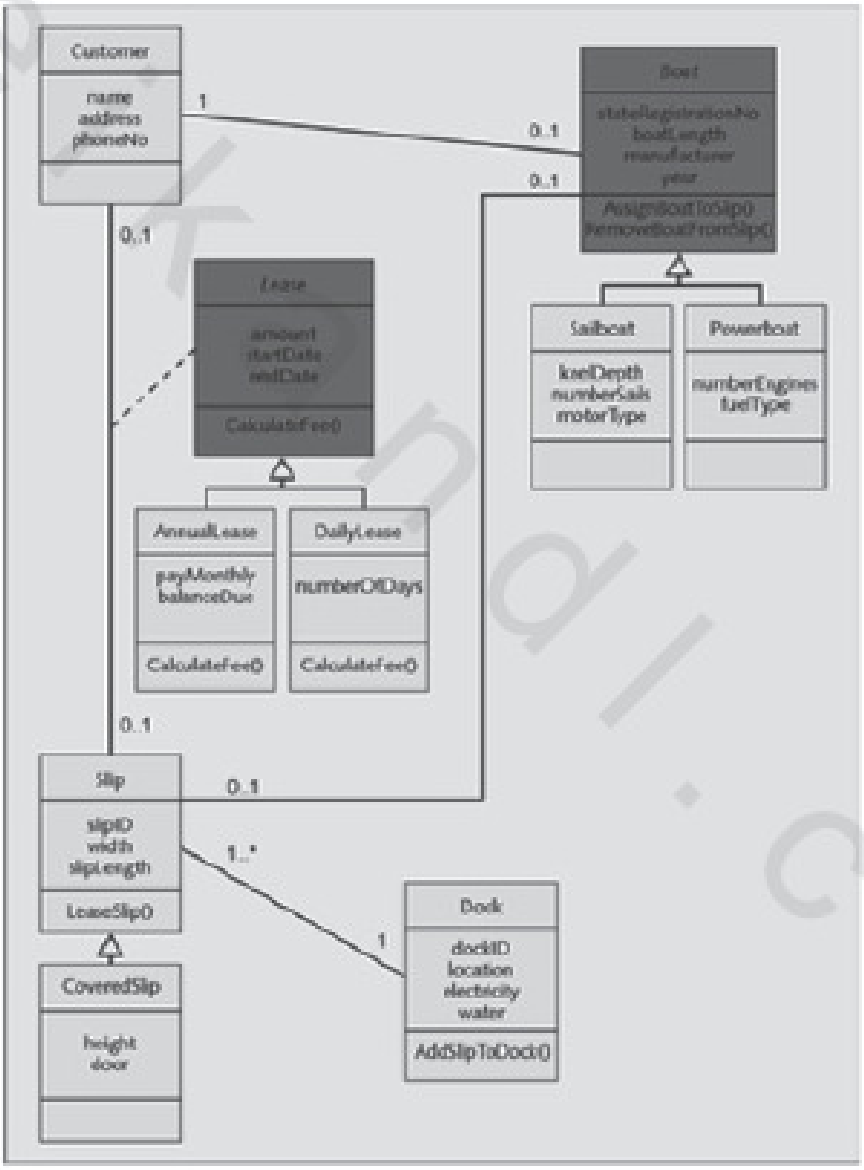
جميع أنواع العقود تتطلب معرفة قيمة الإيجار بالدولار (Amount) كما تتطلب تسجيل كل من تاريخ بداية العقد (Start Date) وتاريخ نهاية العقد (End Date)، ولذلك تعتبر هذه الصفات صفات مشتركة وسوف تعرف في داخل الصنف. وكما ذكرنا سابقاً فإن العميل مخير بين عمل عقد سنوي (Annual Lease) أو عمل عقد يومي (Daily Lease)، فإذا تعاقد العميل عقداً سنوياً فله مطلق الحرية أن يدفع قيمة العقد على أقساط شهرية أو يدفعها دفعة واحدة ويتطلب ذلك أيضاً تسجيل القيمة المتبقية (Balance)، أما إذا تعاقد عقداً يومياً فيجب على العميل دفع قيمة الإيجار كاملة في البداية وذلك لا يتطلب تسجيل القيمة المتبقية.

والعقد اليومي يتطلب تسجيل عدد أيام الإيجار (Number of Days)، وإذا رجعنا إلى كيفية حساب قيمة الإيجار نجد أنها ستكون مختلفة في العقد السنوي عن العقد اليومي، ففي العقد السنوي تعتمد القيمة على عرض مرسى المركب (Width of Slip)، أما في العقد اليومي فيعتمد حساب القيمة على عدد أيام الإيجار؛ ولذلك لابد من تعريف الإجراء CalculateFee في كل من الصنفين بشكل مختلف، ويجب أيضاً أن تظهر في كلا الصنفين في نموذج Class Diagram لتوضيح أنه يوجد خلاف في شفرة كل منهما.

إذا انتهينا من تفاصيل كل من الصنف Boat والصنف Lease والصنف Slip، فننتقل إلى الأصناف الأكثر سهولة وهي الصنف Customer (العملاء)، والصنف Dock (الأرصفة). يوضح الشكل رقم (٥.١٤) نموذج Class Diagram كامل لأصناف مجال المشكلة. يتم تعريف كل رصيف (Dock) بواسطة الصفة DockID التي ستحتوي على حرف أبجدي (أ، ب، ج)، وبواسطة موقعه (صفة Location) والتي تخزن الموقع (شمال الخليج الصغير، ضوء الشاطئ). أيضاً لابد من معرفة هل الرصيف يحتوي على كهرباء (صفة Electricity) ومياه (صفة Water) أم لا. أما الصنف Customer فيعرف عملاء الشركة وهم الأشخاص الذين يمتلكون المراكب (Boats) ويستأجرون المراسي (Slips)، ويجب أن يحتوي النظام على المعلومات الأساسية لعملاء الشركة مثل الاسم (صفة Name)، والعنوان (صفة Address)، ورقم الهاتف (صفة .Phone No).

كما يحتوي كل من الصنف Slip والصنف Boat والصنف Dock على الإجراءات الأساسية (كما هو واضح في الشكل رقم ٥.١٤) حيث لا تتطلب المراحل الأولى من التحليل سرد جميع الإجراءات ولكن نهتم فقط بالإجراءات الهامة. ولأن الصنف Slip هو المسؤول عن تأجير نفسه، فإنه يحتوي على الإجراء LeaseSlip. ولأن الصنف Boat هو المسؤول عن إضافة نفسه لمرسى أو إلغائه، فإنه يحتوي على الإجراء AssignBoatToSlip والإجراء RemoveBoatFromSlip. ولأن الرصيف Dock يحتوي على العديد من المراسي، فإنه يحتوي على الإجراء

AddSlipToDock. وسوف نكمل تصميم نموذج Class Diagram بإضافة علاقات الترابط (Association Relationships) بين الأصناف، وكما نلاحظ على سبيل المثال أن الرصيف يحتوي على العديد من المراسي وأن المرسى موجود في رصيف فإن الخط بين الصنف Dock والصنف Slip يمثل علاقة الترابط بينهما. ويجب كتابة تعددية العلاقة (Multiplicity) على نهايات الخط، فعلى سبيل المثال إن الرصيف يحتوي على مرسى واحد أو أكثر ("*..١")، والمرسى يوجد في رصيف واحد فقط (١).



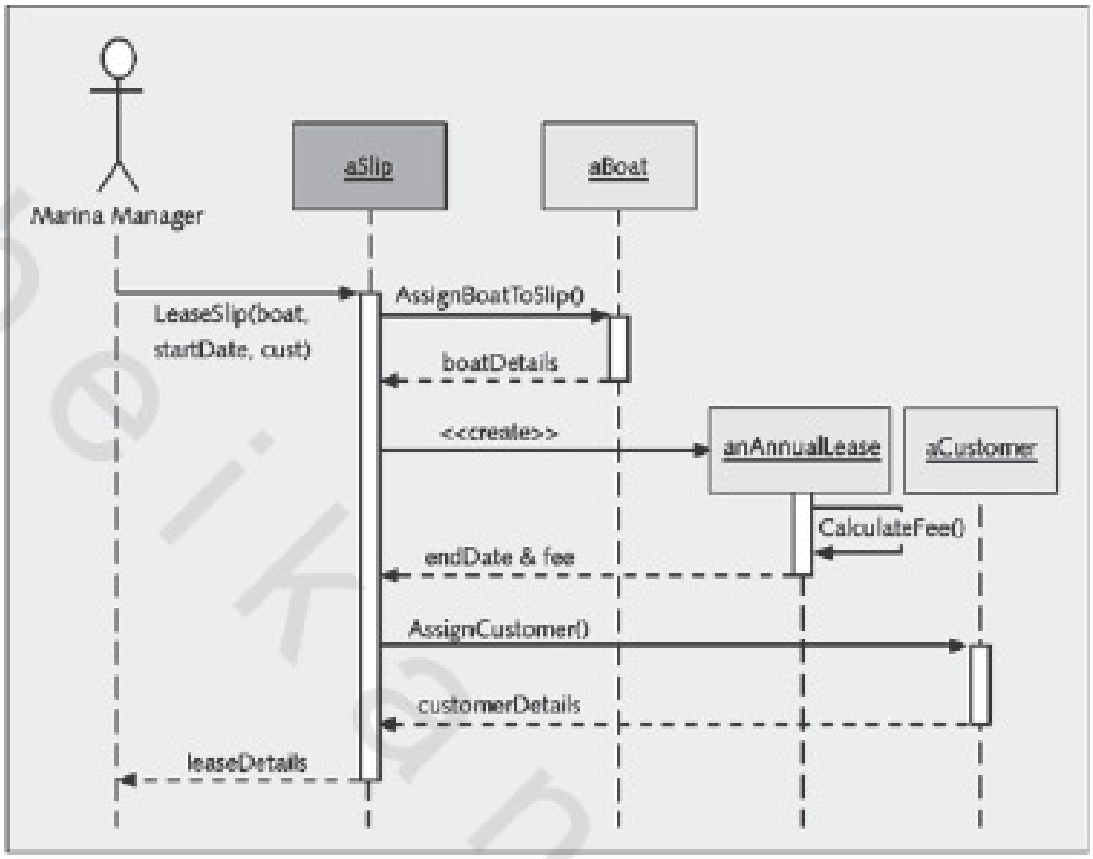
الشكل رقم (٥,١٤). نموذج Class Diagram المحسن لشركة برادشو ماريا.

كما توجد الكثير من علاقات الترابط الأخرى مثل المركب الاختياري (Optionally) يسند إلى مرسى وكل مرسى اختياري يحتوي على مركب، ولذلك فإن تعددية العلاقة كتبت على نهايات الخط هكذا (١..٠)، وهذا يعني أنه يمكن لشركة برادشو الاحتفاظ بمعلومات عن مركب حتى ولو لم يسند إلى مرسى. والعميل الاختياري يملك مركباً أو لا، ولكن لابد أن يملك المركب لعميل واحد فقط، ولذلك يجوز للعميل أن يؤجر مرسى حتى ولو لم يمتلك مركباً، ولا يوجد سبب لشركة برادشو أن تحتفظ بمعلومات عن مركب ليس مملوكاً لعميل. إن قيود تعددية العلاقات تعرف سياسات أعمال الشركة.

إن آخر علاقة ترابط هي التي تربط بين الصنف Customer والصنف Slip حيث يؤجر العميل الاختياري مرسى، والمرسى الاختياري يؤجر إلى عميل، ولكن العلاقة ليست بهذه السهولة فهي معقدة نوعاً ما. في الحقيقة إن عملية التأجير لا تتم إلا إذا وجد كل من العميل والمرسى ووجدت نية الإيجار عندئذ توجد عقد الإيجار؛ ولذلك فإن الصنف Lease يطلق عليه اسم صنف ترابط (Association Class) لأنه مرتبط بعلاقة (علاقة الترابط بين الصنف Customer والصنف Slip) وليس مرتبطاً بصنف مثله، وكما هو واضح في الشكل رقم (٥,١٤) فإن الصنف Lease يتصل (خط متقطع) بالخط الواصل بين الصنف Customer والصنف Slip. إن صنف الترابط يتم برمجته مثل الأصناف العادية تماماً.

تصميم نماذج Sequence Diagram لشركة برادشو
Creating a Bradshaw Sequence Diagram

لقد أشار فريق التطوير إلى أهمية وجود بعض الإجراءات الواضحة في نموذج Class Diagram مثل الإجراء AssignBoatToSlip الموجود في الصنف Boat، والإجراء LeaseSlip المعرف داخل الصنف Slip، وكذلك الإجراء CalculateFee المعرف داخل الصنف Lease. لقد تم التوصل إلى هذه الإجراءات نتيجة التفكير في السيناريوهات المختلفة بجميع حالات الاستخدام المطلوبة وتوثيقها بواسطة تصميم نموذج Sequence Diagram لكل سيناريو. إن هذه الإجراءات الثلاث مطلوبة مثلاً للسيناريو Lease Annual Slip to Existing Customer (إيجار عقد سنوي لعميل معرف مسبقاً) وهو أحد أربع سيناريوهات لحالة الاستخدام Lease Slip. ويوضح الشكل رقم (٥,١٥) نموذج Sequence Diagram لهذا السيناريو، ولكن يجب تصميم نموذج Sequence Diagram لكل سيناريو معرف لحالة استخدام، ولكن في هذه الفقرة اكتفينا فقط بهذا المثال.



الشكل رقم (٥,١٥). نموذج Sequence Diagram الخاص بسيناريو إيجار عقد سنوي لعميل معرف مسبقاً.

وكما هو واضح أن المستخدم في هذا السيناريو هو المستخدم Marina Manager (مدير الشركة) والذي يريد أن يؤجر مرسى لعميل موجود. ولعمل ذلك، يرسل المستخدم رسالة LeaseSlip للكائن aSlip مرسلاً معه الكائن Boat والكائن Customer مع بداية تاريخ العقد (معلومات حول المركب والعميل وبداية العقد)، عندئذ يتفاعل الكائن aSlip بتنفيذ الإجراء LeaseSlip مع الكائنات الأخرى لإتمام العمل حيث يرسل رسالة AssignBoatToSlip للكائن aBoat والذي يعيد تفاصيل المرسى كاملاً، ثم يرسل رسالة Create للصنف AnnualLease (مرسلاً معه تاريخ بداية العقد) لإنشاء عقد سنوي جديد (كائن من النوع AnnualLease)، عندئذ يستدعي الكائن الجديد الإجراء CalaculateFee الخاص به لحساب قيمة الإيجار، وبعد ذلك يعيد تاريخ نهاية العقد وقيمة الإيجار للكائن aSlip، وبعد ذلك يرسل الكائن aSlip للكائن aCustomer رسالةً طالباً منه إسناد نفسه للعقد، ثم يجيب الكائن aCustomer بمعلوماته كاملة. وأخيراً يعيد الكائن aSlip معلومات الإيجار كاملة للمستخدم (تفاصيل المركب، وتفاصيل العقد، وتفاصيل العميل).

وكما ذكرنا سابقاً إن النماذج التي تُنشأ خلال مرحلة التحليل نطلق عليها النماذج المنطقية (Logical Models) وإنها تحتوي فقط على كائنات أصناف مجال المشكلة، ولكن عند الانتقال إلى مرحلة التصميم فيجب توسعة النماذج وذلك بإضافة كائنات واجهة المستخدم التي يتفاعل معها المستخدم وكائنات التعامل مع قواعد البيانات التي تتفاعل مع قواعد البيانات. سوف ترى العديد من نماذج Sequence Diagram خلال هذا الكتاب لتوضيح أمثلة برامج بلغة VB .NET.

عزيزي القارئ الآن وبعدما قمت بتوثيق حالات الاستخدام والسيناريوهات المختلفة مع أصناف مجال المشكلة وبعض التفاعلات بين الكائنات، أنت جاهز الآن لكتابة برامج تعتمد على الكائنات لتعريف أصناف مجال المشكلة لنظام شركة برادشو مارينا وهذا ما سوف نتعلمه خلال الفصل القادم.

وخلال الفصول التي تلي الفصل القادم سوف تتعلم أيضاً كيف تكتب برامج التوارث وعلاقات الترابط وأصناف واجهة المستخدم وأصناف التعامل مع قواعد البيانات، أي تكملة نظام المعلومات المطلوب لشركة برادشو مارينا، كما ستتعلم تطوير النظم التي تعمل على الإنترنت.

ملخص الفصل

Chapter Summary

- إن تحليل النظم يعني دراسة مجموعة من المتطلبات التي يقوم بها النظام وفهمها وتحديدها (System Requirements)، والمتطلبات تعني احتياجات المستخدم من النظام حيث يتم التعبير عنها بلغة المستخدم، وعادة يتم تمثيل تلك المتطلبات مستخدماً أشكال (Diagrams) أو نماذج (Models).
- يشمل أسلوب تطوير النظم المعتمد على النماذج (Model-Driven Development) على إنشاء النماذج المنطقية أثناء مرحلة التحليل وإنشاء النماذج المادية أثناء مرحلة التصميم وذلك لتوضيح ما هو مطلوب وكيف يمكن تنفيذه.
- يختلف الأسلوب التكراري (Iterative Approach) في عملية تطوير النظم عن أسلوب الشلال (Waterfall Approach) حيث تنفذ كل من عملية التحليل والتصميم والبرمجة على التوازي (أي في نفس الوقت) ويكرر كل منها إلى أن ينتهي المشروع، أما أسلوب العمل المتزايد (Incremental Approach) يعني الانتهاء من نظام فرعي وتشغيله قبل الانتهاء من بقية الأنظمة الفرعية. ويوضح النموذج الحلزوني (Spiral Model) الأسلوب التكراري في تطوير النظم.

- تقدم لغة النمذجة الموحدة UML مجموعة مثالية وموحدة من الأشكال والنماذج التي يمكن استخدامها لتوصيف النظم المعتمدة على الكائنات أثناء كل من مرحلة التحليل والتصميم. وتم التركيز على ثلاث نماذج فقط داخل هذا الكتاب.
- يوضح نموذج Use Case Diagram الوظائف الرئيسة للنظام (ما يقدمه النظام للمستخدم)، ويطلق على كل وظيفة اسم حالة استخدام (Use Case) بواسطة المستخدم (يطلق عليه Actor)، ويمكن تحديد حالات الاستخدام للنظام بواسطة حصر الأحداث التي يجب أن يستجيب لها النظام حيث توجد ثلاثة أنواع من الأحداث: أحداث خارجية (External Events)، وأحداث زمنية (Temporal Events)، وأحداث تغيير حالة (State Events). يمكن توثيق كل Use Case بواسطة أكثر من سيناريو (مجموعة متتالية من الخطوات التي يتبعها المستخدم عندما يتفاعل مع النظام).
- يوضح نموذج Class Diagram توصيف أصناف كائنات النظام التي تتفاعل معا لتنفيذ مهام النظام، كما يوضح نموذج Class Diagram علاقة التوارث (Inheritance) بين الأصناف والتي تسمى أيضاً هرمية التعميم والتخصيص (Generalization/Specialization Hierarchy).
- يوضح نموذج Sequence Diagram التفاعلات التي تحدث بين كل من مستخدم النظام وكائنات النظام لتنفيذ حالة استخدام (Use Case) أو لسيناريو محدد خاص بحالة استخدام.
- تستطيع التمييز بين مرحلة التحليل OOA، ومرحلة التصميم OOD مستخدماً أسلوب ثلاثي الطبقات (Three-Tier) الذي يحتوي على ثلاثة أنواع من الأصناف. أولاً: أصناف مجال المشكلة وهي أصناف الكائنات المتعلقة بمجال عمل المستخدم، وثانياً: تعريف أصناف واجهة المستخدم وهي الأصناف التي تسمح للمستخدم التعامل مع أصناف مجال المشكلة، وأخيراً: أصناف التعامل مع البيانات وهي الأصناف التي تجعل أصناف مجال المشكلة تتفاعل مع قواعد البيانات.
- تعرفنا في هذا الفصل على حالة الدراسة الرئيسة التي ستبقى معنا خلال جميع أبواب الكتاب وهي احتياج شركة برادشو مارينا للسفن لنظام معلومات، وهي شركة خاصة تؤجر مراسي القوارب لعملائها كما تقدم خدمات القوارب، وتقع هذه الشركة على بحيرة كلينتون (بحيرة كبيرة داخل الولايات المتحدة). ومن ثم عرضنا كل مفاهيم تطوير النظم المعتمدة على الكائنات ونماذج UML المطلوبة لتوصيف متطلبات النظام.

المصطلحات الأساسية

Key Terms

صنف تجريدي (Abstract Class)	تطوير النظم المعتمد على النماذج (Model-Driven Development)
الكائن فعال (Active Object)	نموذج مادي (Physical Model)
المستخدم (Actor)	نموذج المنطقي (Logical Model)
صنف ترابط (Association Class)	النموذج الحلزوني (Spiral Model)
النموذج التفاعلي (Dynamic Model)	النموذج الساكن (Static Model)
حدث خارجي (External Event)	خط الحياة (Lifeline)
حدث زمني (Temporal Event)	حالة استخدام (Use Case)
حدث تغيير حالة (State Event)	أسلوب الشلال (Waterfall Approach)
الأسلوب التكراري (Iterative Approach)	

أسئلة المراجعة

Review Questions

١- النموذج هو:

(أ) كائن مادي يُرى من ثلاثة أبعاد.

(ب) تمثيل لعناصر العالم الواقعي.

(ج) أنشئ فقط لأنظمة اللعب البسيطة.

(د) مفيد فقط لأمثلة الفصول.

٢- في بيئة التطوير المعتمدة على الكائنات، النموذج هو:

(أ) أشياء أنشئت فقط أثناء OOD.

(ب) أشياء أنشئت أثناء كل من OOA و OOD.

(ج) نادراً ما يستعمل لوصف متطلبات النظام.

(د) دائماً يكون نموذجاً مادياً.

٣- تحليل النظام يعني:

(أ) تعريف احتياجات النظام لإنجاز العمل.

(ب) إنشاء برامج باستخدام تكنولوجيا برمجة الكائنات المفضلة من قبل المستخدمين.

(ج) إنشاء نماذج تظهر كيفية إنشاء مكونات النظام المتنوعة بواسطة تقنية محددة.

(د) تطوير الأنظمة فقط باستخدام أساليب تطوير برمجة الكائنات.

٤- تصميم النظام يعني :

(أ) تعريف احتياجات النظام لإنجاز العمل.

(ب) إنشاء برامج باستخدام تكنولوجيا برمجة الكائنات المفضلة من قبل المستخدمين.

(ج) إنشاء نماذج تظهر كيفية إنشاء مكونات النظام المتنوعة بواسطة تقنية محددة.

(د) تطوير الأنظمة فقط باستخدام أساليب تطوير برمجة الكائنات.

٥- إنشاء النماذج المنطقية من متطلبات النظام أثناء التحليل والنماذج المادية أثناء التصميم هي فكرة من وراء :

(أ) التطوير التكراري.

(ب) التطوير المعتمد على النماذج.

(ج) التطوير المتزايد.

(د) التطوير الحلزوني.

٦- جزء من النظام يتم استكماله ويوضع موضع التنفيذ قبل انتهاء النظام بالكامل عند استخدام :

(أ) التطوير التكراري.

(ب) التطوير المعتمد على النماذج.

(ج) التطوير المتزايد.

(د) التطوير الحلزوني.

٧- لتأكيد الطبيعة التكرارية للتطوير، يتم عرض المشروع كحلزونة تبدأ في المنتصف وتتحرك للخارج عند استخدام :

(أ) التطوير التكراري.

(ب) التطوير المعتمد على النماذج.

(ج) التطوير المتزايد.

(د) التطوير الحلزوني.

٨- جرادي بوتش، وجيم رامبوتش، وإيفلر جاكوبسون أفراد مسؤولون عن تعريف وتوحيد :

(أ) التحليل والتصميم الهيكلي.

(ب) لغة النمذجة الموحدة.

(ج) النموذج الحلزوني.

(د) OOA و OOD.

٩- نموذج UML الذي يعرض مستخدمي النظام ووظائف النظام يطلق عليه :

(أ) نموذج مهام النظام.

(ب) نموذج الأصناف.

(ج) نموذج تتابع الأحداث.

(د) النموذج التفاعلي.

١٠- نموذج UML الذي يعرض أصناف الكائنات الموجودة في النظام يطلق عليه :

(أ) نموذج مهام النظام.

(ب) نموذج الأصناف.

(ج) نموذج تتابع الأحداث.

(د) النموذج التفاعلي.

١١- نموذج UML الذي يعرض الكائنات المتفاعلة يطلق عليه :

(أ) نموذج مهام النظام.

(ب) نموذج الأصناف.

(ج) نموذج تتابع الأحداث.

(د) النموذج التفاعلي.

١٢- يوجد العديد من نماذج UML التي لا تعرض في هذا الفصل تتضمن :

(أ) نموذج مهام النظام.

(ب) نموذج الأصناف.

(ج) نموذج تتابع الأحداث.

(د) النموذج التفاعلي.

١٣- يعتبر نموذج الصنف مثالاً لـ:

(أ) النموذج المسرد.

(ب) النموذج التفاعلي.

(ج) نموذج التطوير التقليدي.

(د) النموذج الساكن.

١٤- يعتبر نموذج تتابع الأحداث مثالاً لـ:

(أ) النموذج المسرد.

(ب) النموذج التفاعلي.

(ج) نموذج التطوير التقليدي.

(د) النموذج الساكن.

١٥- في نموذج الأصناف، تمثل الخطوط التي ترسم لاتصال صنفين:

(أ) الصفة.

(ب) النموذج.

(ج) علاقة الترابط.

(د) خط الحياة.

١٦- أيٌّ من التالي والذي بعض منه صفر لكن العديد منه كائن واحد من الممكن أن يتضمن في علاقة؟

(أ) *..0.

(ب) *..1.

(ج) 0..1.

(د) 1.

١٧- أيٌّ من التالي والذي بعض منه واحد لكن العديد منه كائن واحد من الممكن أن يتضمن في علاقة؟

(أ) **..0.

(ب) *..1.

(ج) 0..1.

(د) 1.

١٨- في نموذج الأصناف، الرمز الذي يمثل تدرج تعميم وتخصيص:

(أ) خط متقطع.

(ب) مثلث يشير إلى صنف مميز.

(ج) معين في نهاية خط الترابط.

(د) إشارة إلى الخواص المدرجة في رمز الصنف.

١٩- في نموذج تتابع الأحداث، يمثل الكائن النشط :

(أ) خطوط الحياة المتقطعة.

(ب) شكل عصا.

(ج) مستطيل له اسم يكتب بحرف كبير وليس تحته خط.

(د) صندوق صغير على خط الحياة.

٢٠- في نموذج تتابع الأحداث، يمثل المستخدم Actor :

(أ) خطوط الحياة المتقطعة.

(ب) شكل عصا.

(ج) مستطيل له اسم يكتب بحرف كبير وليس تحته خط.

(د) صندوق صغير على خط الحياة.

٢١- أصناف الطبقات الثلاثة في التصميم ثلاثي الطبقات تتضمن كل الآتي ما عدا:

(أ) طبقة نظام التشغيل.

(ب) طبقة واجهة الاستخدام.

(ج) طبقة مجال المشكلة.

(د) طبقة التعامل مع البيانات.

٢٢- في التصميم ثلاثي الطبقات، يفترض أن الطبقة الأولى هي :

(أ) طبقة نظام التشغيل.

(ب) طبقة واجهة الاستخدام.

(ج) طبقة مجال المشكلة.

(د) طبقة التعامل مع البيانات.

أسئلة المناقشة

Discussion Questions

١- ما هو الفرق بين النماذج المنطقية والنماذج المادية؟ صف كيف يمثل نموذج Sequence Diagram نموذجاً منطقياً تارة ونموذجاً مادياً تارة أخرى؟ وهل يمكن لنموذج Class Diagram أن يمثل النموذجان نفسهما؟ وكيف يصبح Class Diagram نموذجاً مادياً؟

٢- يتوافق أسلوب التطوير المعتمد على الكائنات (OO) مع كل من أسلوب التطوير المتزايد (Incremental Development) وأسلوب التطوير التكراري (Iterative Development) في عملية تطوير النظم. ناقش هذه الجملة.

٣- يصبح التواصل مستحيلاً بين أفراد تطوير النظم حول عملهم أثناء المشروع بدون استخدام لغة UML. ناقش هذه الجملة مع ذكر الأسباب.

٤- لقد حددنا سوياً متطلبات نظام المعلومات المبدئية لشركة برادشو مارينا للسفن، ولكن ستحتاج شركة برادشو إضافة متطلبات أخرى في القريب العاجل، وربما تحتاج إلى إضافة متطلبات أخرى بعد فترة زمنية طويلة. ناقش كيف ستستخدم أسلوب التطوير المتزايد (Incremental Development) لتحقيق احتياجات الشركة.

مشاريع الفصل
Projects

١- افترض أن لديك متطلبات نظام موصوفة في ست حالات استخدام. ما هي الخطوات المطلوبة التي يجب اتباعها خلال أسلوب التصميم المعتمد على ثلاثي الطبقات لتطوير النظام؟ افترض أن هناك خطة تتكون من ثلاثة تكرارات حيث يشمل كل تكرار حالتا استخدام. كن محدداً في استخدام النماذج.

٢- كما ذكرنا في سؤال المناقشة الرابع أنه قد تم تحديد متطلبات نظام المعلومات مبدئية لشركة برادشو مارينا للسفن (مرحلة ١)، ولكن ستحتاج شركة برادشو إضافة متطلبات أخرى في القريب العاجل (مرحلة ٢)، وربما تحتاج إلى إضافة متطلبات أخرى بعد فترة زمنية طويلة (مرحلة ٣). قم بسرد ثلاث حالات استخدام على الأقل أثناء المرحلة الثانية، ثم أربع حالات استخدام على الأقل أثناء المرحلة الثالثة.

٣- إذا تم إضافة متطلبات تسجيل خدمات المراكب لعملاء شركة برادشو، ما هو صنف مجال المشكلة الذي يجب إضافته لنموذج Class Diagram؟ وما هي علاقات الترابط التي ستربطه مع أصناف النموذج الحالي؟ وهل ترى ضرورة أهمية توسعة هذا الصنف ليشمل أصنافاً فرعية (اذكرها)؟ وهل توجد هناك أصناف أخرى تريد إضافتها؟ قم بتصميم نموذج Class Diagram كاملاً بعد إضافة الأصناف المقترحة.

٤- يوضح الشكل رقم (٥.١٥) نموذج Sequence Diagram لسيناريو تأجير مرسى لعميل موجود. صمم نموذجاً مشابهاً لسيناريو تأجير مرسى لعميل جديد.