كيف تحلّل المشكلة

الغرض من هذا الموضوع هو تقديم بعض الأفكار والأساليب لتجزئة المشكلة الكبيرة إلى عدة مشيكلات صغيرة يسهل حلها بشكل فردي لمساعدتك أثناء دراسة المادة.

البدأ في السيطرة على المشكلة

للتعرف على المشكلة يجب أن تركّز على مالذي سيعمله النظام بدلاً من أن تفكّر في كيف سيقوم بعمله. المشكلة تتواجد في العالم الحقيقي ، وحلها يتواجد في الكمبيوتر وبرامجه.

وأحد النقاط الهامة التي تساعدك في التعرف على النظام هي أن تنظر للنظام من وجهة نظر المستخدم ; كيف سيستخدمون النظام وكيف ستساعدهم خصائص النظام في تحقيق وتنفيذ أهدافهم ومهامهم.

إذا لم تقم بالتعرف على المشكلة بشكل صحيح في البداية ، فربما ستبني نظاماً يقوم بحل المشكلة الخاطئة ، بمعنى أن الحل لم يتطابق مع المشكلة الحقيقية.

يمكن تلخيص الأفكار السابقة باقتباس من كتاب "أطر المشكلة" للكاتب مايكل جاكسون: "إنه من المهم التركيز مباشرة على المشكلة ، وعدم الذهاب مباشرة إلى تصميم الحل. الحاسب الآلي والبرمجيات هي الحل; والمشكلة تتواجد في العالم الخارجي للحاسب الآلي. مهما كنت حريصاً ، سيكون من الصعب أن تفصل المشكلة عن حلها. [...] هذا الكتاب يتحدث عن تحليل المشكلات ، وليس عن الحلول"

إذا كنت من المهتمين في برمجة الحاسب الآلي ، ستكون دائماً مستعجلاً وستقوم بكتابة الكود البرمجي لحل المشكلة قبل أن تقوم بالتعرّف في البداية على المشكلة. فحاول ألا تقوم أبداً في حل المشكلة إلاّ بعدما تتعرّف عليها وعلى أصحاب المصلحة فيها واحتياجاتهم بشكل جيّد.

تجزئة المشكلة – Breaking down the problem:

عادةً تكون المشكلة معقدة قليلاً ، وبالتالي حلولها أيضاً ستكون معقّدة. لذلك تحتاج إلى التفكير بوسائل تساعدك على التحكم بهذا التعقيد. والأسلوب التقليدي لعمل هذا هو أن تقوم بتحليل وتجزئة المشكلة الأساسية إلى عدة مشاكل أو مشيكلات فرعيّة ، بحيث يكون بمقدورنا التّعامل مع هذه المشيكلات وحلها بسهولة. وفي مرحلة أخرى سنقوم بعكس العملية وسندمج حلول هذه المشيكلات مع بعض ليصبح لدينا حل للمشكلة الرئيسية.

أطر المشكلة – Problem framework:

في عالم حل المشاكل ، ستجد بأن هنالك الكثير من المشاكل الفرعية التي تظهر باستمرار في مجالات مختلفة ، وغالباً ستكون هنالك حلولاً جاهزة لمثل هذه المشاكل ، ويمكننا في هذه الحالة أن نقوم بإعادة استخدام نفس الحل الموجود لهذه المشكلة في أماكن أخرى.

الكاتب (وليس المغني) مايكل جاكسون يقول في كتابه بأن أغلب المشاكل الموجودة في الحياة تكون كبير ومعقدة بحيث يصعب التعامل معها في مرحلة واحدة ، ولذلك يجب علينا تحليلها إلى عدة مشيكلات صغيرة. ويدّعي مايكل بأن هذه المهمّة يمكن عملها بسهولة عن طريق أطر المشكلة (problem frames). المقصود بأطر المشكلة هو محاولة التتعرف على الحالات (المشاكل) التي تتكرر ، ومن ثم نقوم بوضع هذه المشاكل في تصنيف واحد (class). الفائدة من هذا هو

أن مطوّري البرامج سيستخدمون أطر العمل التي أنشؤوها كدليل مساعد للمستقبل ، فعندما يرون مشكلة قد سبق لهم التعامل معها سيستطيعون الاستفادة من واستخدام الحل الموجود مسبقاً في إطار العمل.

ويفرّق مايكل بين استخدام كلمة "نظام" لوصف المزيج الكلي بين العالم الخارجي و الحاسب الآلي وبين استخدامها لوصف مزيج الأجهزة (Hardware) والبرمجيات (Software). ففي الوصف الأول نحن نحاول أن نصف النظام الأشمل (والذي تقع فيه المشكلة) وهذا مانريده ، بينما الوصف الثاني فنحن نصف النظام الدقيق (والتي يتواجد فيه الحل). ويقول أيضاً بأن كلمة الآلة "machine" تستخدم في الحالات التي يريد أن يتحدث فيها عن هذا النظام الدقيق والمحدود (الحاسب والبرمجيات).

ويفرق أيضاً بين التمثيل التحليلي (analytic modeling) والتمثيل التناظري (analogic modeling) ، حيث أن التمثيل التحليلي يعني بأننا نصف النظام في العالم الخارجي ، بينما التمثيل التناظري يعني بأننا نصف النظام الموجود في داخل الحاسب الآلي. في هذه المرحلة نحن سنركز على الأنشطة والمهام التحليلية في عملية تمثيل الذي يحدث في نظام الحياة الواقعية بدلاً من التفكير في كيفية تمثيل النظام في الحاسب الآلي.

توجد خمسة أطر أساسية للمشكلة ذكرها مايكل في كتابه. في مايلي هذه الأطر الخمسة مع وصف بسيط عن كل إطار.

إطار السلوك المطلوب - The Required Behavior frame:

"فكرته هي" كما يذكر الكاتب "أنه يوجد جزء من العالم الفيزيائي يكون سلوكه محكوم بحيث يقوم بتلبية بعض الحالات. المشكلة هي أن نقوم ببناء آلة تقوم بتقديم هذا التحكّم"

يعني هنا ستقوم الآلة بعمل بعض الأشياء بحسب الحالات التي تحدث في الواقع ، لكل حدث تكون هناك استجابة معيّنة من قبل الآلة. مثل نظام التحكم بالإشارات الضوئية والذي يقوم بالتحكم بتدفق حركة السيارات حيث يقوم بجعل الإشارة خضراء لمدة طويلة في حالة الزحام والعكس في حالة قلة السيارات.

إطار السلوك المأمور - The Commanded Behavior frame:

"فكرته هي أنه يوجد جزء من العالم الفيزيائي يكون سلوكه محكوم من قبل الأوامر التي تصدر من المشغِّل. المشكلة هي أن نقوم ببناء آلة تقوم بقبول الأوامر التي تصدر من المشغِّل ومن ثم القيام بالتحكم المناسب".

هنا ستقوم الآلة بتنفيذ الأشياء التي يطلبها الشخص المتحكِّم أو مشغِّل الآلة. مثل جهاز التحكّم عن بعد لأجهزة التلفاز على سبيل المثال.

إطار عرض المعلومات - The Information Display frame:

"فكرته هي أنه يوجد جزء من العالم الفيزيائي تكون حالاته وسلوكه بحاجة مستمرة إلى المعلومات. المشكلة هي أن نقوم ببناء آلة تقوم بأخذ المعلومات من العالم ومن ثم تقوم بعرضها في المكان وبالشكل المطلوب".

مثال: معلومات السرعة والمسافة المقطوعة التي تعرضها شاشة السيارة.

إطار قطعة العمل البسيطة - The Simple Workpiece frame:

"فكرته هي أن المستخدم سيكون بحاجة إلى أداة تسمح له بإنشاء وتحرير تصنيف معيّن من نصوص الحاسب القابلة

للتحرير ، أو كائن رسومي ، أو أي شيء مماثل لها، بحيث نستطيع أن ننسخها أو نطبعها أو نحللها أو نستخدمها بطرق أخرى. المشكلة هي أن نقوم ببناء آلة تستطيع أن تعمل مثل هذه الأداة".

إطار التحويل - The Transformation frame:

" فكرته هي أنه توجد بعض ملفات المدخلات في الحاسب تكون بياناتها بحاجة إلى تحويل إلى مخرجات محددة. البيانات المخرجة يجب أن تكون بصيغة معينة ، ويجب أن تكون مشتقة من البيانات المدخلة بحسب قوانين محددة. المشكلة هي أن نبني آلة تقوم بصنع المخرج المطلوب من المدخلات".

الدلالات - Heuristics:

الفوائد من التجزئة والتحليل الجيد للمشكلة هي أنها تساعدك في وصف أو توثيق المشكلة بشكل واضح وأيضاً تساعدك في محاولتك لحل المشكلة ، بالإضافة بالطبع إلى أنها تساعدك في فهم المشكلة. على أية حال ، فإنه لاتوجد قوانين واضحة تساعدك في تحليل المشكلة بشكل جيد أو حتى تساعدك في التأكد من أنك قمت بتحليل المشكلة بشكل جيد. ومن المهم أن تعرف بأنه من المحتمل وجود تشكيلة واسعة من التحاليل الممكنة للمشكلة الواحدة ، وأن العديد من هذه التحاليل ستكون متساوية في جودتها ، ومن النادر أن يكون هنالك تحليل واحد يعتبر الأفضل بين التحاليل الأخرى.

يقول مايكل: "تحليل المشكلة ليس علم صريح. ولكنه من الممكن أن يعطيك دلالات مفيدة ، ومن الممكن أن يكون منظم بشكل منطقي".

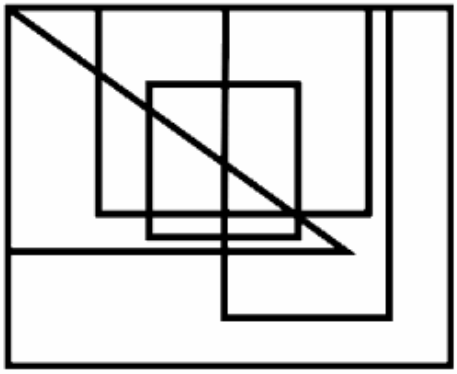
أمثلة على الدلالات المحتملة:

- **التعرف على المشكلة الرئيسية:** عادةً تكون هنالك حاجة رئيسية واضحة يقوم النظام بالتعامل معها ، لذلك ركز عليها ومن ثم توسّع من خلالها.
- **التعرف على المشكلات المساعدة:** توجد العديد من المشاكل الفرعية حول المشكلة الرئيسية ، وعادةً تكون على شكل مشاكل معلومات فرعية ترتبط بالمشكلة الرئيسية.
- **استعمل التحليل القياسيّ للمشاكل الفرعية:** بعض أطر المشكلة تكون بطبيعة الحال ذات طابع مركّب ، وبالتالي ستقود إلى التعرف على المزيد من المشاكل الفرعية.
- **ابحث عن المشاكل الفرعية ذات الإيقاع الزمني المختلف:** إذا كانت هنالك أنشطة تحدث في إطارات زمنية مختلفة بشكل كبير ، فهذه الأنشطة يجب أن تعامل على أنها مشاكل فرعية.
- **ابحث عن المشاكل الفرعية ذات الطابع المختلف:** من الأفضل أن يتم فصل المشاكل الفرعية المتعلقة برغبة المستخدم من النظام (كالتّمنّي) عن المشاكل الفرعية المتعلقة بمالذي يجب أن يحصل عليه المستخدم من النظام.
- **ابحث عن التعقيدات المتبقية:** أي مشكلة فرعية تكون معقدة نسبياً يجب أن تقوم بتجزئتها إلى عدة مشاكل فرعية إضافية.
- **التحقق من حاجة تمثيل المستخدم:** إذا كان مجموعات مختلفة من المستخدمين بحاجة إلى الوصول إلى أي جزء في النظام ، سيتوجب علينا أن نعامل كل مجموعة كمشكلة فرعية مستقلة.

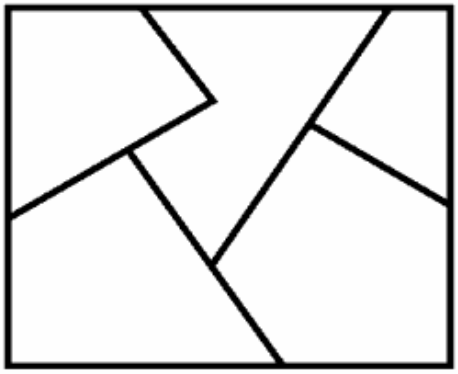
تقييم التحليل:

عندما تصل إلى نهاية عملية التحليل ، كيف يمكنك أن تحكم على نجاح جهودك من عدمه؟ يجب أن تملك مجموعة من المشاكل الفرعية وكل واحدة منها تكون أصغر وأسهل من المشكلة الأصلية. ويجب أن تكون (المشكلات الفرعية) مشكلة مكتملة بذاتها ، بحيث تكون لاتعتمد على أي مشكلة فرعية أخرى حتى نستطيع حلها. ويجب أن تكون مكتملة كمجموعة ، بحيث تغطي مجموعة المشكلات الفرعية جميع القضايا الأساسية المتعلقة بالمشكلة الأساسية.

مايكل يقول بأنه خلال المرحلة الأولية لتحليل المشكلة إلى مشاكل فرعية ، ستجد نفسك عادةً تقوم بصنع العديد من التداخلات بين المشاكل الفرعية التي تقوم بتعريفها والتي تعتبر نظرات مختلفة لنفس المشكلة ، بدلاً من أن تقوم بتجزئة المشكلة إلى مشاكل فرعية مستقلة كلياً. ويجب التنبه إلى هذا. ويمكن ملاحظة الفرق بين عمليتي التداخل و التجزئة في الشكلين التالية:



التداخل



التجزئة

الاحتياجات والخصائص:

أحد الأساليب المختلفة عن أسلوب أطر المشكلة هو أن نركّز على قضية رغبة المستخدمين (أو بشكل أعم أصحاب المصلحة) من النظام. والكاتب دين لفنجويل (Dean Leffingwell) كتب مقالة عن هذه القضية وعرض فيها نظرة عامة وبسيطة عن أهم الأشياء المرتبطة بها.

يبدأ دين بتذكيرنا بأن الغرض من مرحلة تعريف المتطلبات في تطوير الأنظمة هو أن نجيب على هذا السؤال الهام جداً: "ماذا يفترض أن يقوم النظام بعمله؟"

وبعدها انتقل للتنبيه على اختلاف الحاجات الحقيقية لأصحاب المصلحة التي تعرفنا عليها في نطاق المشكلة (problem domain) عن خصائص النظام التي ستقوم بتلبية هذه المتطلبات والتي حددناها في نظاق الحل (solution domain).

يعرف دين حاجة صاحب المصلحة (stakeholder need) على أنها "انعكاس للعمل أو الشخص أو المشكلة العملية التي يجب الاهتمام بها حتى يتم تبرير اعتبار أو شراء أو استخدام نظام جديد".

ويعرّف أيضاً الخاصية (feature) على أنها "خدمة يقدمها النظام حتى يتم تلبية حاجة أو احتياجات أصحاب المصلحة".

وينبّه دين على حقيقة أن الخصائص لا تعتبر كإعادة صياغة لحاجات أصحاب المصلحة. وإنمّا هي استجابة مباشرة للمشاكل التي ذكرها أصحاب المصلحة ، وهي توفِّر حل عام للمشكلة (top-level solution).

ويقول دين بأننا نستطيع أن نصف نظام ما عن طريق تعريف مابين 25 إلى 50 خاصيّة تمثّل سلوك هذا النظام.

وأيضاً يأكّد على أن الاهتمام والتفكير بحالات الاستخدام (use cases) للنظام من الممكن أن تكون مفيدة جداً في عملية وصف سلوك النظام. وهو يعرّف حالة الاستخدام (use case) على أنها "وصف لسلسلة من العمليات التي ينفذها النظام والتي تسفر عن نتيجة ذات قيمة للمستخدم". فيمكننا أن نقول بأن حالات الاستخدام تصف لنا كيف يتفاعل ويتواصل المستخدم مع النظام. وعادةً يكون هنالك عدة حالات استخدام والتي ستدل على كيف يمكن تنفيذ خاصية معينة.