Agile Methods

د. محمد الأحمد

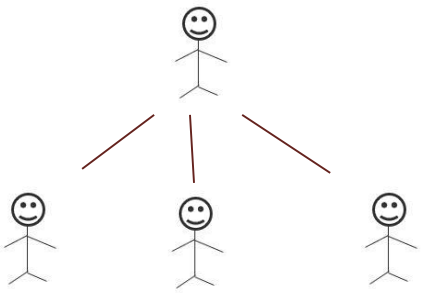
Agile Methods

وتسمى المنهجيات الرشيقة، الهدف منها وأهميتها: تطوير نظام برمجي بسرعة دون الحاجة للكثير من الوثائق أو الخطط بل نضع خطط مرحلية لمدة أسبوع أو أسبوعين وعند الانتهاء منها نضع خطة جديدة ويكون الزبون جزء من فريق التطوير حيث يقيّم كل جزء من النظام ويعطي ملاحظاته مما يساعد في مواكبة التطورات بشكل سريع.

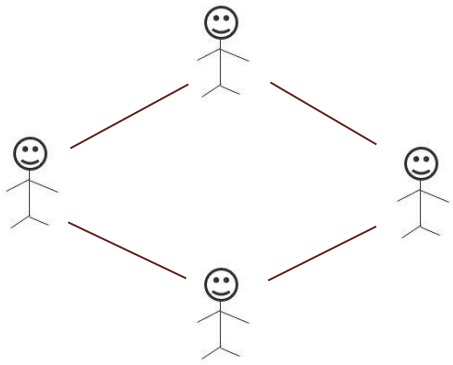
أهم ميزة لـ agile method هي الاستجابة للتغييرات وكلما كان هناك تغييرات كان ذلك أفضل، حيث أن الزبون في أي لحظة قد يغير رأيه وإن حدث ذلك في مرحلة متقدمة لا يوجد مشكلة بل نستطيع التغيير بسهولة وذلك لكوننا نطور بشكل تزايدي.

بما أن هدفنا التسريع سيكون تركيزنا على الكود بشكل أساسي أكثر من عمليات التحليل والتصميم فعلياً يكون التحليل والتصميم والتكويد مرحل متداخلة تجري مع بعضها البعض وبالتالي يكون التركيز على المبرمجين وعلى عملهم وعلى التواصل القائم بينهم ويمكن القول إن agile method بحاجة إلى agile teamفرق رشيقة قادرة على التواصل والتفاهم فيما بينها أما عدد أعضاء الفريق فهو من 3 إلى 12 لأنه إذا كان العدد كبير يصبح التفاهم صعب وإذا كان العدد أقل تصبح الانتاجية قليلة ، وأهم ما في ال agile method هي طريقة الإدارة، دوماً عند تطوير النظم البرمجية لدينا إدارة للمشروع project management

- في plane-driven إدارة المشروع لها الشكل التالي:



أما في **agile methods** تكون بالشكل التالي:



من ميزاته أيضاً توافر التفاهم بين أعضاء الفريق وأن يمتلكوا مهارات متقاطعة ومتكاملة، أحياناً أحد الأفراد يحتاج أن يقوم بالتحليل والتصميم والتكويد وكل شخص يكون على إطلاع معه بعمل زملاءه لأن عددهم قليل وعملهم متكامل.

XP، SCRUM

سندرس مثالين عن هذه المنهجيات: **XP،SCRUM** .

Rapid Software Development: تطوير البرمجيات السريع، الهدف السرعة نهتم بموعد التسليم، أحد أهم مشاكل المشاريع البرمجية أن التسليم يتجاوز الخطة الزمنية، مع **agile** نستطيع تجاوز هذه العقبة لذا يجب أن تكون نظمنا مرنة وقابلة للتطوير ولإضافة خدمات جديدة بشكل سريع ومرن دون انهيار النظام.

نتيجة للتطور الكبير والتنافس الحاصل بين الشركات يكون التغيير أساسي والمطلوب منا تطوير برمجيات مرنة قابلة لتغيير لكي تعكس هذا التطور في بيئة العمل، عملياً ما يميز **RSD** أن جميع مراحل التطوير متداخلة وحتى نحقق الهدف الأساسي من **agile** يجب أن نستخدم أدوات (**tools**) مثل **CASE Tools** وغالباً الواجهات تكون جاهزة أو نستخدم بنيات أو سمات جاهزة، إذا جاءت الـ**agile methods** للتغلب على البيروقراطية التي كانت موجودة في **plane-driven** من حيث وجود **documentation** والتوثيق التي تسبب ضياع في الوقت كما أن **plane-driven** تتميز بوجود فترات متباعدة بين مراحل التطوير فلا تستطيع مواكبة التغييرات بعكس الـ **agile** .

إذاً أهمية الـ**agile** هو التغلب على التوثيق ومراحل التطوير الطويلة المملة لعدم الحاجة لتخطيط كامل للمشروع وبالتالي التركيز على الكود دون إهمال التصميم بشكل كامل لكن بشكل أخف من الـ **plane-driven**.

من شروط المنهجيات الرشيقة أنها تقدم خدماتها على شكل إصدارات فلا نقوم بتطوير النظام كتلة واحدة ونقدم نظام فعال (يعمل) بشكل سريع ومتقبل للتغييرات وبالتالي هو مناسب لبيئة الأعمال الحديثة.

مبادئ ال Agile

1) شمول الزبون: الزبون جزء من فريق التطوير (أو أي شخص مرشح من قبل الزبون) عندما نقوم بتطوير أي نظام نسأل من هم الstakeholder أصحاب المصلحة (المستفيدون) وهم أشخاص أو مؤسسات ممثلة بأشخاص مثلاً عندما نطور نظام في جامعة يكون المستفيدون هم الطلاب والمدرسين والجامعة ممثلة بالعميد.

الفرق بين ال actor وstakeholder: هو أن ال actor يمكن أن يكون أشخاص أو غير أشخاص: وقت، مؤسسة ... أما ال stakeholder هو أشخاص تحديداً.

عند تطوير أي نظام نبدأ بتحديد الstakeholder والتواصل مع الزبون لتوفير ممثل عنه يفهم بيئة العمل ويكون مع فريق التطوير طوال الوقت بهدف تحديد المتطلبات وإعطاء الfeed back لنقوم بالتعديل وفقاً لرأي الزبون لنصل إلى رضاه.

2) نطور على شكل إصدارات بشكل تزايدي حيث يحدد الزبون أولويات كل إصدار.

3) التأكيد على الأشخاص وليس الاجرائية، التركيز على المبرمجين والتواصل معهم والتفاهم فيما بينهم كلما زاد التفاهم كلما ازداد الإنجاز وأي تأخير أو مشكلة يؤثر على سير التطوير.

4) تقبل التغيير بشكل المتقدم على شكل إصدارات.

5) الحفاظ على البساطة: لتجنب ضياع الوقت في التخطيط والتطوير.

تطبيقات المنهجيات الرشيقة:

النظم ذات الحجم الصغير أو المتوسط بالإضافة للنظم غير الحرجة التي لا تؤثر على حياة الأشخاص، غير مناسبة للنظم الكبيرة كونها تعتمد بشكل أساسي على documentation والتي تكون قليلة في agile.

لا يمكن استخدامها إذا كان الزبون لا يستطيع المشاركة في التطوير لأنها سوف تفشل حتماً.

فريق التطوير يجب أن يكونوا مدربين للعمل تحت هذه البيئة غالباً جميع أعضاء الفريق يشاركون في جميع مراحل التطوير من تكويد وتخطيط وتصميم واختبار، في حال وجود الكثير من المستفيدين يجب تحديد الأولويات بينهم.

للحفاظ على البساطة يجب التعاون بين أعضاء الفريق.

العقد مع الزبون: في plane-driven بعد الانتهاء من التحصيل وتحديد المتطلبات نصل إلى contract مع الزبون أي نوقع عقد يشبه دفتر شروط على تطوير النظام يحوي المتطلبات الوظيفية وغير الوظيفية في ال plane-driven يعد أساسي وجود العقد أما في ال agile ليس أساسي مجرد تسليم نظام فعال كل فترة هو عقد بحد ذاته.

بالنسبة للصيانة: تعد الagile سيئة بالنسبة للصيانة بسبب قلة ال documentation فيوجد مشكلة في عملية الصيانة ولا يمكن تحديد الأخطاء ومكانها وإصلاحها لذلك لا تستخدم ال agile في النظم الحرجة.

مقارنة بين ال plane-driven وAgile:

الفرق الأساسي هو وجود سرعة في Agile لذلك تدعى منهجيات رشيقة (light) أما في plane-driven يوجد بطئ فهي منهجيات ثقيلة (heavy).

في plane-driven يوجد الكثير من التخطيط والوثائق أما في Agileلا يوجد بل يتم الاتفاق على سير التطوير بين الفريق فيوجد أريحية في العمل.

لذلك الAgile مناسبة لتطبيقات الويب بشروط ألا تكون critical خلال أسبوع يمكن أن نسلم نسخة أساسية من المشروع.

في plane-driven كل مرحلة تتم على حدى ويكون خرج كل مرحلة دخل المرحلة التي تليها أما في Agile يتم التخطيط والتصميم والتكويد معاً.

القضايا التقنية المتعلقة بالAgile :

يوجد خوف من الagile اذا كان أصحاب المؤسسة لا يعرفون ما هي الagile يكون لديهم خوف منها كون معروف عنها أنها سريعة وتحوي أخطاء لكن هذا الشيء ليس صحيح بالعكس نتيجة المراجعة عليها بسبب الإصدارات الكثيرة هذه العمليات تحسن من الجودة لكن بشكل أساسي أننا بحاجة للاعتماد على فريق عمل مرن.

غالباً عندما تكون الأنظمة ذات دورة حياة طويلة نستخدم plane-driven مثل البنوك أما الأنظمة التي تكون دورة حياتها صغيرة تطور عن طريق ال Agile.

معظم الشركات تفضل الأمان الموجود في الأنظمة الثقيلة لأنه عند انهيار النظام يوجد وثائق للقيام بعملية الصيانة.

الكلفة في Agile منخفضة وفي ال plane-driven مرتفعة.

Extreme Programming

أول منهجية رشيقة في التسعينيات ترتكز على السرعة بتقديم نظام فعال خلال فترة قصيرة يتراوح الفرق بين إصدارين أسبوعين وأسبوعين.

طريقة عملها: يتم جمع المتطلبات الوظيفية بشكل أساسي عن طريق قصة (story) وهي توصيف الزبون للخدمات ثم تقسيم هذه القصة إلى مهام (task) وفريق التطوير مكون من مبرمجين (2) والزبون أحد المبرمجين يقوم بتوكيد هذه القصة (لا يوجد تخطيط) والمبرمج الأخر يقوم بـreview على الكود لاكتشاف الأخطاء ثم في القصة التالية يتبادلان

الأدوار عند الانتهاء من القصص ينتج نظام فعال بعدد من الخدمات الأساسية نعرضه على الزبون للتعديل وإجراء التغييرات، تعتمد على تقنيتين:

- **Pair programming:** البرمجة الثنائية يوجد مبرمجين (2) أحدهما يكتب الكود والاخر يراجعه ثم يتبادلان الأدوار.
- **Test driver programming:** لكتابة الكود نبدأ بالtesting نكتب كود على أساس وجود test مكون من مجموعة بيانات عندما ينجح الtest نتابع البرمجة وإلا نقوم باستكشاف الأخطاء وإصلاحها.

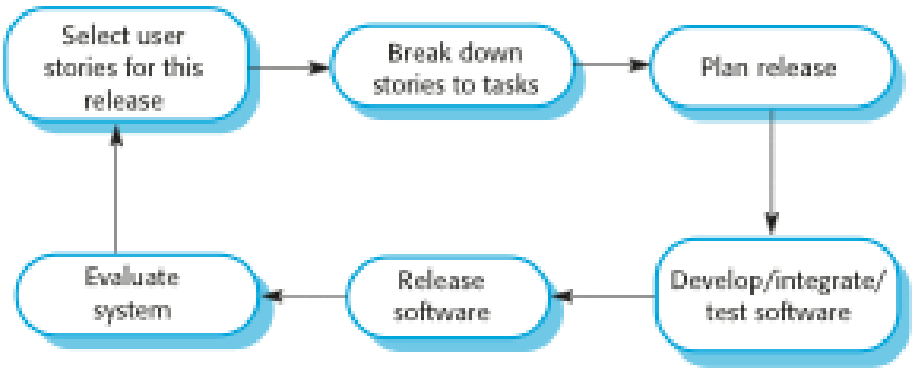
وهي أكثر منهجية مستخدمة ومنتشرة لأن التركيز على الكود والبرمجة.

ممكن خلال يوم واحد أن يقدم إصدار واحد أو أكثر.

يوجد فيها تخطيط ولكن خطط بسيطة كلما انتهينا من قصة نخطط للأخرى.

نعتمد على الtest لكتابة الكود بشكل أساسي.

بما أن الxp هي Agile يجب أن تحقق مبادئ الAgile



Extreme Programming Practices

1. **التخطيط المتزايد:** على القصص المختارة كل قصة مجزأة لمهام وكل قصة يتم التخطيط لها.
2. **إصدارات صغيرة:** النظام يصدر على شكل إصدارات خدماتها قليلة.
3. **تصميم بسيط.**
4. **Test-First-Development:** الاعتماد على tasting لكتابة الكود.
5. **Refactoring (إعادة التصنيع):** تقنية تستخدم في الكود والتصميم وهي عملية إعادة تسمية المتحولات والصفوف والتوابع ولا نقوم بتغيير البنية الداخلية وإنما فقط الخارجية أي الشكل بهدف أن يكون الكود سهل وبسيط readable قابل للقراءة والتحسين وتصميم الأخطاء على موضوع قلة الوثائق.

6. **Pair programming.**

7. **الملكية الجماعية:** بشكل عام يعتبر الكود ملك للمبرمجين معاً وهذا يمنح الكود قوة وثقة بالجودة.

8. **Onsite customer:** الزبون موجود كمحدد للوظائف والتقييم.

ملاحظات:

- الجودة تقاس بالإنتاجية (*productivity*) وهي مقدار الانجاز خلال وحدة الزمن

$\frac{output}{input}$ بالنسبة للبرمجة الثنائية $\frac{line\ of\ codes}{Man*manth}$

أي انها حاصل قسمة عدد أسطر الكود على عدد الأشخاص المشاركين في كتابته ضرب الزمن وتختلف الانتاجية حسب القصة التي نقوم ببرمجتها وفي الإصدارات المتقدمة تصبح الإنتاجية أكبر لكون الزمن أقل أي أنه في أول الإصدارات تكون الإنتاجية قليلة ثم تزداد مع الإصدارات التي تليها والجودة بشكل عام تتحسن، هناك طريقة أخرى لقياس خرج البرمجة وهي *function point* ويتم قياس الكود بعدد الوظائف المقدمة فيه حسب طرق رياضية معينة.

- يوجد أدوات تساعد في الاختبار مثل *Funit* وهي أداة موجودة في الجافا تقوم بإجراء اختبارات الوحدة على الكود.

فوائد pair programming:

a) دعم فكرة الملكية الجماعية للكود.

b) عملية المراجعة لتحسين الكود.

c) نشر المعرفة بين أعضاء الفريق.

ملاحظة: ليس شرط أن يكون الفريق مكون من 2 فقط بل ممكن لكل 2 أن يعملوا معاً.

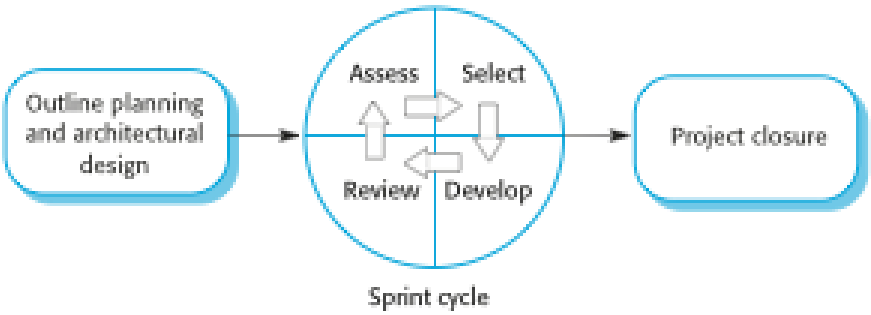
Agile project management

في xp لا يوجد إدارة وبالتالي لا يوجد ضياع في الوقت والجهد كون المدير يصرف 90٪ من الوقت والجهد على التواصل مع الفريق والزبون كما في ال plane-driven.

Scrum

منهجية إدارية أو إطار عمل إداري لتنظيم مشاريع رشيقة فترة العمل بين أسبوعين إلى شهر لنحصل على إصدار من النظام ويتقدم النظام على شكل إصدارات، أعضاء الفريق من 5 لل 9 أشخاص، بحاجة لفريق متفاهم ليس متكامل وإنما متقاطع المهارات.

يتم العمل عليها بشكل أساسي ضمن ما يسمى **the sprint cycle**



نبدأ بالتخطيط والتصميم المعماري بالاستعانة مع **Serum Owner** [الزبون] أصحاب سلطة قوية هو الذي يحدد ماذا سوف يتضمن النظام ضمن ما يسمى **System backlog** ثم ندخل ضمن حلقة [سباق] **sprint** تمتد من أسبوعين حتى شهر يتم فيها بشكل أساسي عملية اختيار المتطلبات الأساسية وتطويرها ثم نقوم بـ**review** ثم يقوم الـ**owner** بتقييم النظام بهدف إجراء التعديلات، وأخيراً نقوم بالتسليم ونكرر المراحل السابقة حتى نطور النظام بشكل كامل.

Teamwork in Scrum

- ✓ **Product owner**: جزء من فريق التطوير في بعض المراحل، كل نهاية دورة يقوم بالتقييم ويعطي الخدمات ولكنه منعزل عن التطوير.
- ✓ **Scrum Master**: الإدارة لا تعطيه سلطة وانما تعطيه دور ليكون مسهل لتحديد المهام والتدريب ويكون على اتصال مع **product owner**.
- ✓ فريق التطوير: يتم تحديد المهام ثم يبدأ التطوير، كل يوم يوجد مقابلة عبارة عن 15 دقيقة في نهاية اليوم يتناقش فيها ما تم انجازه في هذا اليوم وما الذي سوف يتم إنجازه غداً وما هي الصعوبات التي واجتنا فبنهاية كل مرحلة تجري مقابلة مع **product owner** من أجل التقييم وإجراء التعديلات.

إذاً لدينا 3 اجتماعات:

i. يومي بين فريق التطوير.

ii. شهري مع **product owner**.

iii. شهري بين فريق التطوير لمناقشة الدروس المستفادة.

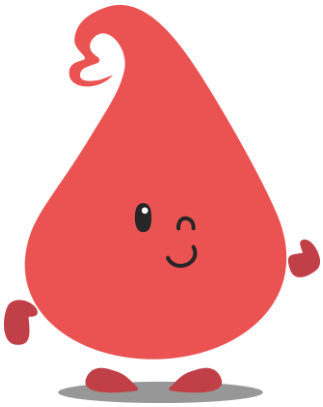
عملية جمع المتطلبات تدعى product backlog مع أولوياتها بعد جمع المتطلبات وتقسيم العمل بين أفراد الفريق ينعزل الفريق عن product owner بهدف منع التشويش ويتم التواصل عن طريق Scram Master والذي مهمته حماية الفريق من العوامل الخارجية.

كل فرد من الفريق على إطلاع تام بالعمل.

فوائد Scram:

1) تقسيم النظام إلى أجزاء على شكل إصدارات كل إصدار يستغرق شهر على الأكثر.
2) جميع أفراد الفريق على إطلاع تام بالعمل.
3) بنهاية كل مرحلة يتم التقييم العمل من قبل الزبون وإجراء التعديلات.
4) وجود روح العمل الجماعية

Large System Development

بعض أجزاء النظم الكبيرة يمكن أن يتم تطويرها باستخدام ال agile لكن ليس الأجزاء الحساسة منها.

انتهت المحاضرة..