السلام عليكم, تحدثنا في المحاضرة السابقة عن العديد من الـ process models منها:

Waterfall – Incremental – Prototype – Spiral – Unified

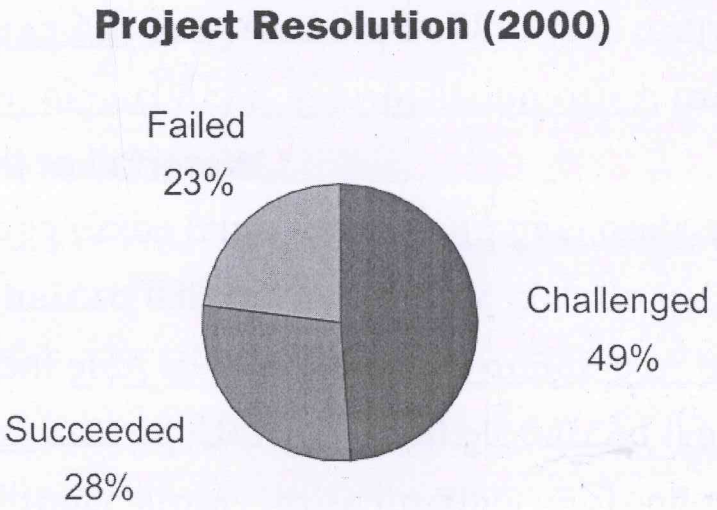
وكلاً منها له استخدامه ولا يوجد جواب محدد ومطلق حول استخدام كل نوع منها فكل نموذج يتمتع بإيجابيات وسلبيات ولكن يمكن القول بأنه يوجد حالات معينة ينصح باستخدام نموذج بدلاً من الآخر ومنها:

- إذا كانت المتطلبات غير واضحة ومبهمة تماماً يمكن استخدام نموذج الـ **Prototype**
  لوضع تصور أولي عن المنتج المراد تنفيذه ولمساعدة الزبون في تحديد المتطلبات التي يريدها.
- إذا كان المشروع كبير يمكن استخدام **Unified** (كما يمكن استخدام **Spiral**).
- إذا كان الفريق متوسط الحجم أو فريق التطوير مبتدئ يمكن استخدام **Waterfall or Incremental**.
- إذا كان المشروع يتطلب تطوير المنتج بأسرع وقت ممكن وأراد المستخدم المنتج كاملاً نستخدم الـ**Waterfall** وإذا أراد جزءاً منه نستخدم الـ**Incremental** ولكن اذا كانت المتطلبات مبهمة وغير مكتملة مع الحاجة للتطوير بسرعة نستخدم الـ**Incremental** ولا نستخدم الـ**Prototype** لأنه بعد الانتهاء من النموذج الأولي للمنتج المطلوب يهمل ولا يرقى ليكون منتج نهائي وعلينا البدء بتطوير المنتج من الصفر بعد تحديد المتطلبات وذلك يعتبر مكلفاً بالنسبة للوقت المحدد .
- إذا كان المشروع ذو خطورة عالية يمكن استخدام **Spiral**.

انتهت المحاضرة السابقة عند الـUnified Process والتي تعتبر أكثر الـmodels رسمية وتستخدم في حالة المشاريع الكبيرة والمتوسطة وهي تتألف من عدد من الأطوار وكل طور نحتاج إلى عدد من الدورات على الـ workflow لإنجاز طور واحد من أطوار الـUP(والتي من الممكن أن تكون غير مستخدمة) كما أن دورة واحدة على الأطوار لا تضمن إنتاج منتج نهائي, أي من الممكن أن نحتاج لدورات أُخرى على الأطوار.

مقدمة:

كل هذه التحسينات والأفكار التي ضيفت على الـPlan-Driven Process Model من Incremental Development و Fast Delivery لم تستطع مواكبة متطلب الـ Rapid Development and Delivery (التطوير والإنتاج السريع) فلم تستطع الاستجابة للأسواق الجديدة و الحالات الاقتصادية المتغيرة وظهور العديد من المنتجات والخدمات المنافسة, فحسب تقرير لـStandish Group (تهتم في دراسة عوامل فشل المشاريع البرمجية وتسمي تقاريرها CHAOS reports) لعام 2000 كانت نسب نجاح وفشل المشاريع البرمجية كالتالي:



حيث **Challenged** هي نسبة المشاريع التي تخطت التكلفة المالية للمشروع **Over-Budget** والإطار الزمني المتوقع **Over-Time estimation** وبالقليل من المتطلبات والمزايا المطلوبة **Fewer Features and Functions** وكان 82% من هذه المشاريع العامل في فشلها هو **Waterfall-style scope management** أو بمعنى اخر **Plan-Driven Process Model**...

حيث كان الفكر السائد هو أن الطرق التي تؤدي إلى المنتجات البرمجية الناجحة هي التخطيط الحذر **Careful Project Planning** وضمان الجودة الرسمية **Formalized Quality Assurance** واستخدام الوثائق الرسمية حتى عند الانتقال من **Framework Activity** إلى أخرى.

كل هذه الأفكار بلورها مجتمع هندسة البرمجيات الذي كان مسؤول عن تطوير برمجيات ضخمة مثل أنظمة المركبات الفضائية والأنظمة الحكومية (السائدة في تلك الفترة)

وكانت هذه البرمجيات تحتاج لزمن تطوير كبير (10 سنوات أو أكثر) والتغيير فيها مكلف فكانت عملية التخطيط والتصميم واستخدام الوثائق بشكل كبير أمراً لا بد منه...

أما مع بدايات 1990 وبدء ظهور البرمجيات الصغيرة والمتوسطة وظهور الحاجة للتطوير السريع (المنافسة) كان لابد من طرق جديدة تواكب الطلب فبدأت فلسفة وطراق الـ**Agile** بالظهور والتبلور حتى عام 2001 حيث وضع التعريف الرسمي للـ**Agile development**.

الـ Agile Development :

Agile تعني الرشاقة والرشاقة في علم هندسة البرمجيات تعني السرعة في التطوير ويتم ذلك بتخفيف بعض النشاطات التي كانت تمارس في الـ **Plan-Driven Approach** والتي كانت سبب في عملية بطء التطوير البرمجي وزيادة

البيروقراطية في العمل البرمجي لكن كما نعلم أن السرعة تتناسب عكساً مع عملية الجودة.

ومن هنا يبدأ مفهوم الـAgile وهو ضبط عملية التوازن بين السرعة وضمان الجودة.

The optimum proportion between speed and quality.

وبالتالي يجب علينا تحديد المراحل والنشاطات **الأقل أهمية** في عملية تطوير المنتج البرمجي بحيث إذا قمنا بإهمالها او التخفيف منها لا يكون تأثيرها كبير.

وهنا يجب التنبيه إلى الاختلاف الكبير بين مفهومي **التخفيف** و**التخلص أو الإزالة** فلا نقوم بالتخلص مثلاً من الـTesting وإنما يمكننا تخفيفها إلى حد ما أو عدم اعتبارها نشاط قائم بحد ذاتها.

المراحل والمهام التي يمكننا إزالتها:

- ← **Analysis Phase**: لأنه في مرحلة الـDesign يمكن أن نقوم بعملية الـ Analysis أيضاً.
- ← **Documentations**: التقارير والوثائق التي يتم تناقلها بين فرق التطوير بين المراحل والنشاطات والتي هي صلة التواصل الرسمية التي يمكن أن يعتمد عليها فريق التطوير.

ومنه نستنتج أن هدف الـ Agile:

هو تحقيق السرعة عن طريق تخفيف بعض الأعمال والنشاطات الروتينية أثناء تصميم المنتج البرمجي.

خصائص وشروط الـ Agile Development:

- الاعتماد على فريق خبير يعتمد على وسائل تواصل مباشر ذات ثقة وخبرة عالية فالـAgile تعتمد وتهتم بالأشخاص أكثر من التخطيط والتعامل الرسمي.

- الحصول على منتج قابل للاستخدام بفترة زمنية قصيرة.
- المراحل تكاد تكون متداخلة أي أنه لا يوجد حدود فاصلة بين المرحلة والأخرى.
- إمكانية التعديل في أي مرحلة من العمل حتى لو كان في الـIncrement الواحدة.
- الزبون (أو من يمثله) هو عضو أساسي من أعضاء فريق التطوير,فالـAgile تهدف لإزالة الحواجز الرسمية بين الزبون والمطورين لتحقيق أكبر قدر من التعاون والسرعة في تحديد المتطلبات والتغييرات.

ومنه الـAgile:

هي عملية تطوير المنتج البرمجي بسرعة بحيث نستطيع تطوير أو إنتاج عدة مراحل versions/increments بسرعة عالية وتتطلب تنفيذ مجموعة من الفلسفات والأدوات تتمحور حول **تحقيق تواصل فعال بشكل غير رسمي بين أعضاء فريق التطوير وتفضيل الوصول لمنتج على الوثائق والأوراق والاعتماد على جعل الزبون عضو من فريق التطوير عوضاً عن الرسمية والمعاملات والاستجابة السريعة للتغييرات بدلاً من اتباع الخطط.**

In 2001, Kent Beck and 16 other noted software developers, writers, and consultants [Bec01a] (referred to as the "Agile Alliance") signed the "**Manifesto for Agile Software Development.**" It stated:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

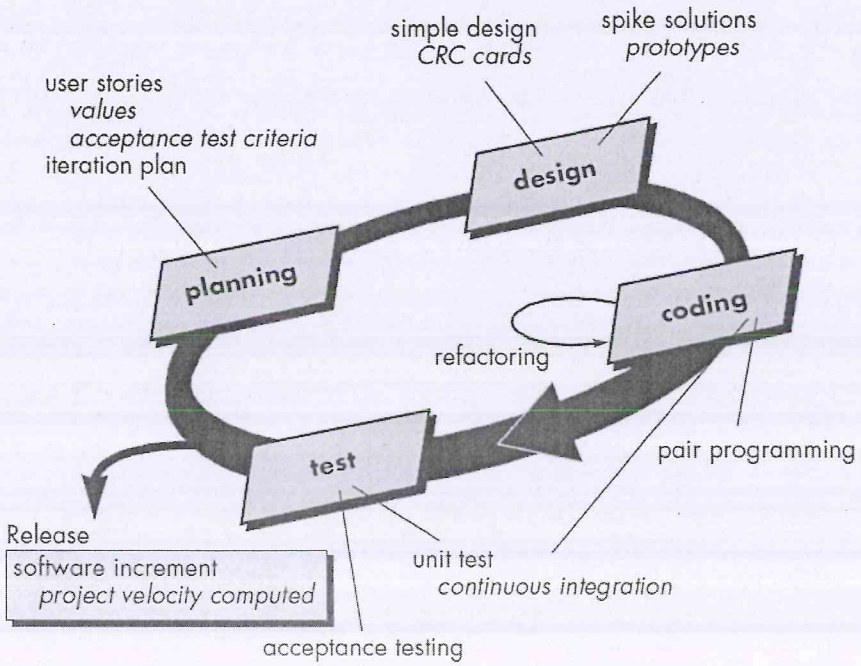
- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

:Agile Approach and Plan-Driven Approach

Property	Agile	Plan-Driven
Project Size	Small to Medium	Small to Very Large
Flow	Incremental, Iterative	Linear, iterative, incremental
Customer Commitment	always	The start of project or increment
Develop new Product	Can be used.	Can be used.
Maintain Existing product	Can't be used, there is no documentation.	Can be used.
Speed	Highest	Slow to High
Respond to changes	Best response	From no response (Waterfall) to good response.
Documentation	No documentation	Good documentation
Team size	small	Small to large
Very detailed Phases	Interleaved	Detailed

بعض النماذج والطرائق في الـAgile Development :

:(XP)Extreme Programming -1



النشاطات التي يمر بها الـXP:

Planning:

في الـXP لايوجد حد فاصل واضح بين الـAnalysis و الـDesign (الـActivities التي تعرفنا عليها في النماذج العامّة) فالـAnalysis غير موجود بل متداخل مع الـPlanning, ووجود الـFlow الدائري يعطي انطباع أن عملية الـXP عملية تزايدية Incremental أي تعتمد على المنتجات الـIncremental عند الانتهاء من كل دورة والـrequirements سيتم توزيعها على increments .

ويطلق على المتطلبات في عملية الـXP اسم User Story مجموعة من القصص او السيناريوهات Scenarios للـوظائف التي يحتاجها الزبون من المنتج والتي يستطيع من خلالها فريق التطوير من فهم السياق العام والوظيفة الأساسية للمنتج المراد تطويره.

ويقوم فريق التطوير بتقسيم كل story الي من مجموعة من النشاطات activities أو المهام Tasks ثم التخطيط للـIncrement القادم بالتعاون مع الزبون وتحديد الـStories ذات الأولوية الأعلى والبدء بها ولايمكن تحديد تاريخ تسليم لهذا الـIncrement فالـagility تعتمد على جمع المعلومات الجديدة مع التقدم في المشروع Explore on the road .

وبعد الانتهاء من الـIncrement الأول يتم تحديد سرعة المشروع Project Velocity:

Project velocity is the number of customer stories implemented during the first release.

والتي يمكن الاستفادة منها في جدولة مواعيد الإصدارات القادمة

Helps estimate delivery dates and schedule for subsequent releases (Increments).

Design:

يقوم فريق التطوير بتصور كامل عن الحل لكل story عن طريق مايسمى CRC cards التي تساعد في عملية التفكير بالمنتج بمفهوم غرضي التوجه

Object-Oriented مع مراعاة مبدأ KIS (Keep It Simple) أي التصميمات البسيطة مفضلة على التصميمات المعقدة.

نلاحظ ان عملية الـXP لا تحوي Quality Assurance أو Quality Check وهذا يتم ضمنيا عن طريق الـtesting واستخدام الـTest unit حيث تعتبر وسيلة الضمان الوحيدة لسلامة المنتج وجودته ولذلك الـ Agile Development يعتمد على Heavy Intensive Testing الاختبارات المركزة...

الـTest unit تتألف من Test cases قد تم وضعها اثناء تصميم الـCRC (مرحلة الـDesign) لكل story وهي عبارة من مجموعة من الـoutputs المتوقعة من تنفيذ كود الـCRC.

عندما يكون الحل مبهم للـUser Story أي يحتاج لتقنية جديدة على الفريق مثلاً(على عكس مانقوم به في الـCRC حين تصميمها فنحن نعلم المشكلة ولدينا تصور كامل للحل) سيكون لدينا حالة شاذة ونقوم بشئ يطلق عليه Spike Solution وهو عبارة عن Prototype نقوم بتطويره بسرعة ويحقق النقاط الرئيسية الأولية للحل نستطيع من خلاله تقليل خطورة الفشل في توصيف المشكلة أثناء تنفيذ البرمجة الفعلية للـstory والتحقق من استيعاب الفريق للـstory على أفضل وجه ممكن...

Coding:

يتم استخدام فريق صغير نسبياً لتكويد عدة CRC معا (شخصين مثلاً) و توزيع فرق أخرى على CRC أخرى وذلك لتحقيق السرعة في الـAgile development ويمكن تطبيق مايسمى Pair Programming في الفريق الواحد فشخص يقوم بالبرمجة والاخر يقوم بمتابعته ومراجعة الكود البرمجي أثناء كتابته و يتم التبديل بينهما من حين إلى آخر...

وعند الحاجة لتعديل أو تحسين الحل لتصميم معين في مراحل متقدمة نقوم بـ Refactoring (إعادة تموضع):

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves the internal structure. It is a disciplined way to clean up code [and modify/simplify the internal design] that minimizes the chances of introducing bugs. In essence, when you refactor you are improving the design of the code after it has been written.

فـعملية **Refactoring** لاتقوم بتغيير الويفة أو تسلسل العمليات التي يقوم بها الكود البرمجي بل تقوم بتحسين البنية الداخلية للكود وتنيفه لتقليل حالات هور الأخطاء البرمجية عند التنفيذ,فـهدف الـ**Refactoring** هو تحسين تصميم الكود البرمجي الذي تمت كتابته مسبقاً.

الهدف الرئيسي لـ**Refactoring** في عملية الـ**XP** هو التحكم بالتعديلات التي نقوم بها مع تقدّم عملية إنشاء المنتج عن طريق التغييرات البسيطة التي نضيفها على التصميم (التي بدورها تحسن بشكل كبير منه), فالـ**XP** تعتمد على القليل من المنتجات المرحلية (عدا الـ**CRC** و الـ**Spike solutions**) والتصميم يعتبرأداة عابرة في عملية التطوير المنتج ويجب أن يكون تحت التعديل باستمرار...

Testing:

بعد الانتهاء من تطوير الكود البرمجي لـ**story** الواحدة نقوم باستخدام الـ **Test Unit** أو الـ**test cases** التي وضعناها اثناء تصميم الـ**CRC** للتاكد من جودة وجهازية الحل

"Fixing small problems every few hours takes less time than fixing huge problems just before the deadline".

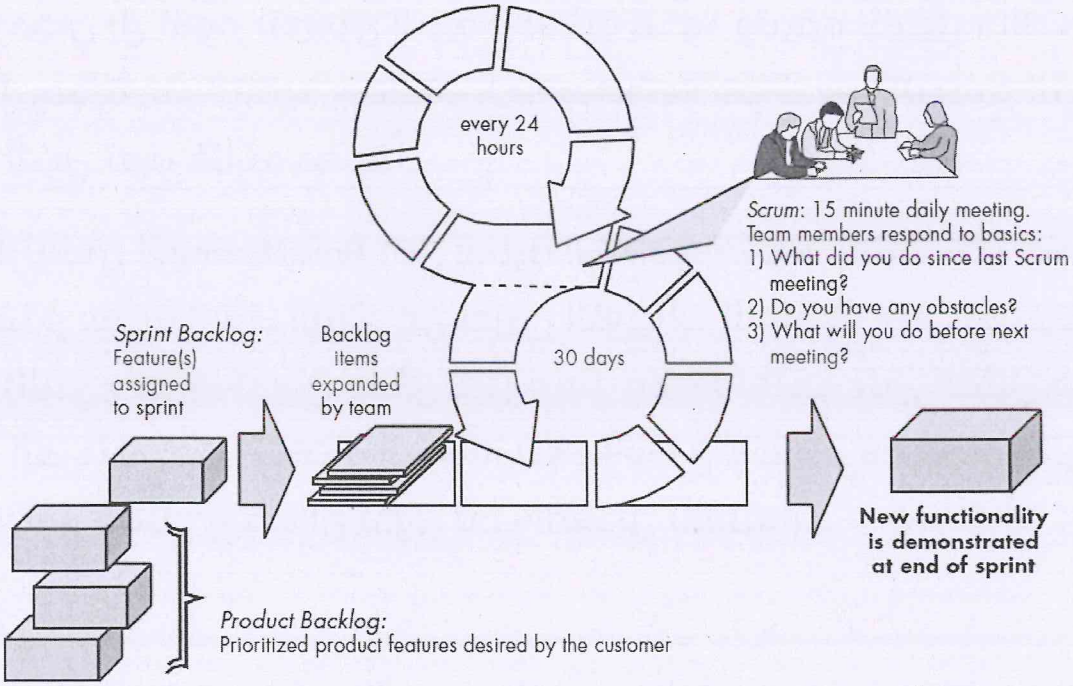
وعند تقديم الـ**increment** للزبون يوجد اختبارات تدعى **Acceptance Tests** أو **Customer Tests** وهي احتبارات معرفة من قبل الزبون يقوم بتعريفها على المنتج بشكل عام وماهو ظاهر وقابل لمراجعته من قبله وعند نجاح الـ**increment** بها

يكون الـ**increment** قد قبله الزبون **Accepted**, يمكن أن نطلق على الـ**increment** قبل قبوله بـ**Beta-Increment**.

بعد إصدار هذا الـ**Increment** نعيد الكرّة ونختار **Stories** جديدة ونخطط للـ**Increment** الجديد وهكذا حتى الوصول للـ**Increment** النهائي الذي يمثل المنتج المطلوب.

SCRUM-2:

أتت الـ**Scrum** لتحل مشكلة أساسية في الـ**XP** فكما علمنا أن الـزبون يشكل جزء من فريق تطوير المنتج مما سيؤدي لوجود توتر **Stress** و تشويش **Interfering** من قبله أثناء العمل إلى المنتج مما يسبب حالة عدم ارتياح في عملية التطوير للفريق البرمجي...



تعتمد الـ**Scrum** على شخص يدعى **Scrum Master** وهو:

1- يكون عقدة التواصل بين الزبون والفريق المطوّر.

2- يضمن عملية التفاعل بين الـ**Sub-teams** بالفريق الواحد.

3- يتأكد من سير عملية تطوير المتطلبات بشكل سليم.

يعتبر الـ**Scrum Master** مسيّر **Facilitator** ولا يعتبر مدير **Manager**.

يقوم الـScrum Master بجمع المتطلبات من الزبون والتي يطلق عليها في هذا النموذج Product Backlog والتي تعتبر قامة To-Do على فريق التطوير القيام بها والتي يمكن أن تكون:

Features, Requirements, User stories and Descriptions

عملية تطوير عنصر واحد من عناصر الـBacklog يطلق عليها اسم Sprint (ينجز ضمن إطار زمني 30 يوم مثلاً Scrum Cycle) وخلال هذا الإطار الزمني يتم تحقيق اجتماع يومي (Scrum Meeting) لمدة 15 دقيقة بين فريق التطوير والـ Scrum Master حتى تنفيذ الـ Increment المطلوب والذي يطلق عليه Shippable Product Increment قابل للشحن و الاستخدام .

ثم يقوم الفريق باختيار عنصر اخر من الـBacklog والبدء بـSprint جديدة وإنتاج Increment جديد وهكذا...

The name is derived from an activity that occurs during a rugby match where a group of players forms around the ball and the teammates work together (sometimes violently!) to move the ball downfield.