ISSN: 1646-8929

*IET Working Papers Series*
**No. WPS09/2009**

**Mário Rui Sampaio Tomás (FCT-UNL)**
(e-mail: mariotom_net@hotmail.com )

# Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação

**IET**
**Research Centre on Enterprise and Work Innovation**
**Centro de Investigação em Inovação Empresarial e do Trabalho**
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
Monte de Caparica
Portugal

**Métodos Ágeis**

**suas características, pontos fortes e fracos e possibilidades de aplicação**[1]

Mário Rui Sampaio Tomás (FCT-UNL)

# ÍNDICE:



[1] Trabalho realizado sob orientação do Prof. António Brandão Moniz para a disciplina "Factores Sociais da Inovação" do Mestrado Engenharia Informática realizado na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

## Resumo:

*Os métodos ágeis estão cada vez mais a ser discutidos pelas empresas, sendo que muitas têm receio de adoptar esta metodologia, devido às dificuldades na reestruturação organizacional. Serão aqui discutidos características, alguns pontos fortes e fracos desta abordagem, quando deve ser escolhida, bem como os meios informáticos que ajudam à implementação da mesma. Ficará reforçado que este método tem aplicação prática e que ganhará vários adeptos nos próximos tempos.*

*Este é, por excelência, o método para projectos de pequena e média dimensão para a Web, com cada vez mais empresas a apostar em criar ferramentas para a aplicação destes conceitos. Estas ferramentas têm de assentar no controlo do estado das tarefas e funcionalidades para determinada 'release' e na distribuição delas pelas equipas de trabalho.*

*É também fundamental permitir que o cliente ajude de modo integrado com a aplicação a capturar requisitos para serem resolvidos pelas equipas de trabalho.*

## Abstract:

*Agile methods are increasingly being discussed by companies, and many are afraid to adopt this approach due to the difficulties in organizational restructuring. In this working paper some characteristics, strengths and weaknesses of this approach will be discussed, as well as when it should be chosen, and the IT tools that help to implement it. It will be reinforced that this method has practical application and that it will gain more followers in the near future.*

*This is,* par excellence*, the method for small-and medium-sized projects for the Web, and many companies are investing in the creation of tools for applying these concepts. These tools must be based on monitoring the status of tasks and functionalities for a given 'release' and the distribution of these tasks and functionalities to work teams.*

*It is also essential to allow the client to help in an integrated manner with the application to capture the application requirements to be solved by work teams.*

**Key-words**: Agile methods; organizational restructuring

**JEL codes**: M15; O31

# 1 Introdução:

Até cerca do ano de 2001, a maioria dos modelos de sucesso para a realização de projectos de *software* eram baseados no modelo 'Waterfall', onde existe uma grande fase de planificação que dá todo o suporte ao desenvolvimento posterior. Pela utilização deste modelo, é muito difícil às empresas aceitarem mudanças de requisitos nas fases posteriores ao planeamento, já que tal exige uma regressão à fase de planificação, de onde pode resultar uma mudança drástica no código produzido.

As ideias subjacentes que guiam o desenvolvimento ágil, começaram significativamente antes da década de 90, mas só em 2001 são publicados sob a forma de manifesto os princípios desta metodologia revolucionária e como veremos mais ágil e mais adequada a determinados tipos de projectos [Secção 2]. Este manifesto, '*Manifesto for Agile Software Developer*'[1] define os doze princípios defendidos pela metodologia em causa, a partir dos quais foram criados "Modelos Ágeis de Processo" para os colocar em prática no mercado.

O modelo em 'Cascata' ou 'Waterfall Model' será referenciado sempre que traga algo de útil à comparação com a abordagem por 'Métodos Ágeis'.

Além dos doze princípios, encontramos no Manifesto [1] as linhas orientadoras que fundamentam esta nova abordagem. Os conceitos chave são:

-'Indivíduos e interacções em vez de processos e ferramentas.'

-'*Software* que funciona em vez de documentação abrangente.'

-'Colaboração do Cliente em vez de negociação de contratos.'

-'Resposta a modificações em vez de seguir um plano.'

Os elementos relegados para segundo plano, não têm de deixar de existir, assumem sim, papéis menos importantes. As mudanças (por exemplo, ao nível da estrutura organizacional, da forma de pensar e trabalhar dos colaboradores) a que esta nova prática obriga, assustam os que vêem nela uma alternativa para a evolução, e esta foi uma das principais razões pela qual a sua entrada no mercado foi lenta. Apesar de tudo, novos casos de estudo foram realizados e daí foram retiradas diversas recomendações para a sua introdução.

É um engano pensar que, segundo esta metodologia que se abandona por completo os métodos tradicionais e as práticas de engenharia de software, até hoje adquiridas, no entanto o processo é transformado e certas fases são encurtadas e outras ganham mais relevo.

Na próxima secção caracteriza-se a 'Metodologia Ágil', as suas características e definem-se os processos mais conhecidos que a suportam. Serão vistas também as vantagens e desvantagens, da metodologia ágil e explica-se

resumidamente um método para a escolha entre 'Metodologias Ágeis' ou 'Clássicas' para um projecto. Na secção 3, discute-se como pode ser abordada a introdução dos processos ágeis numa organização. Na secção seguinte (Secção 4), listam-se ferramentas que suportam a 'Metodologia Ágil' e as suas características, fazendo-se uma comparação qualitativa entre elas. Finalmente, na secção 5, apresentam-se as principais conclusões a que se chegou.

# 2 Metodologia Ágil

De acordo com as linhas orientadoras da 'Metodologia Ágil', as pessoas têm um papel fundamental no desenvolvimento dos projectos, sendo para isso essencial que em todas as equipas exista uma boa comunicação entre os intervenientes, haja motivação e que cada indivíduo se preocupe com a qualidade. É pois valorizada a entrega de um produto funcional e adequado ao que o cliente realmente deseja; a preocupação centra-se na produção do *software* pedido, sendo a maioria da documentação gerada a partir das ferramentas usadas na produção. O cliente é frequentemente chamado a intervir, iteração a iteração, tendo um papel decisivo na definição dos novos requisitos, contrariando a prática de quase tudo ser planificado e acordado no inicio do projecto. É geralmente aceite, e muitas vezes é comprovado na prática, que numa 'Metodologia Plan-driven/Waterfall' o plano definido grande parte das vezes não chega ao fim igual ao que foi proposto no início do projecto. Nenhum projecto é totalmente previsível, portanto ser 'ágil' é ter conhecimento desta realidade e aceitar que os requisitos habitualmente mudam, em suma, estar pronto para acomodar a mudança de forma simples e rápida.

Pretende-se então a redução dos ciclos de entrega, maior adaptabilidade e flexibilidade a alterações ou ao aparecimento de novos requisitos dos *stakeholders*, assim como o cumprimento dos prazos de entrega.

Existem já algumas abordagens/modelos ágeis de processo, que seguem estes princípios. Entre estas temos as seguintes listadas e na respectiva subsecção serão explicadas as suas principais diferenças: ([21]):

-XP (Extreme Programming) [2],[3]

-SCRUM [4], [5]

-Feature Driven Development (FDD) [6]

-Dynamic Systems Development Method (DSDM) [7]

-Lean Development [8]

-Crystal [9]

Os *stakeholders* e gestores de projectos tendem a ser mais exigentes e a querer reagir mais rapidamente ao mercado. Pretende-se então com este método um incremento da produtividade, uma conclusão mais rápida, uma entrada antecipada do projecto no mercado (pode não ser a sua versão final), retorno mais rápido do investimento, maior qualidade, menores custos e aumento da satisfação do consumidor.

Os clientes são considerados parte da equipa de desenvolvimento, uma vez que a todo o momento são questionados sobre prioridades e testes de versões.

A equipa de desenvolvimento inspecciona relatórios, define novas metas e atribui tarefas em iterações curtas, oferecendo ao cliente versões do *software* incrementalmente mais funcionais e melhoradas.

Quer-se portanto que as aplicações vão ao encontro das necessidades reais de negócio. Não se entrega um grande projecto ao cliente que não foi por ele testado, e que só quando o vê se apercebe que não era bem aquilo que necessitava, apesar de este corresponder às especificações acordadas.

## 2.1 Características fundamentais a encontrar em equipas ágeis

Podem-se encontrar vários *papers*, de autores reconhecidos a enfatizar a importância das pessoas e das equipas neste processo, como por exemplo [22]. De onde se retira a ideia de que o processo se deve adaptar às equipas. É fundamental que certas características-chave estejam presentes entre as pessoas da equipa ágil segundo Roger S. Pressman[21]:

-Competência: Quer em metodologias ágeis, quer em tradicionais diz respeito a talento inato, habilidades específicas relacionadas a software e a um conhecimento global do processo que a equipe decidiu aplicar. A habilidade deve ser ensinada a todos os membros das equipas ágeis.

-Foco Comum: Os membros podem ter diferentes competências e habilidades, mas todos eles deve ter o mesmo foco, que é entregar um incremento de software em funcionamento ao cliente dentro do prazo prometido.

-Colaboração: A equipa precisa de colaborar uns com os outros, com o cliente e com os gerentes de forma a conseguir analisar, avaliar e usar/comunicar informação de forma eficiente.

-Capacidade de tomada de decisão: É importante que a equipa tenha autonomia para tomar decisões de tópicos técnicos e de projecto.

-Habilidade de resolver problemas vagos: Uma equipa ágil lida continuamente com ambiguidades e será confrontada com modificações, os problemas que resolvem num dia, podem não ser significantes numa fase posterior, porque teve de se mudar a maneira de realizar algo. No entanto a

equipa aprendeu a resolver aquele tipo de problema.

Respeito e confiança mútua: citando DeMarco e Lister [23] a equipa tem de funcionar como um todo, e deve-se tornar "tão fortemente aglutinada que o todo é maior que a soma das partes".

Auto-organização: "(1) A equipa ágil organiza-se para o trabalho ser feito; (2) a equipa organiza o processo para melhor acomodar seu ambiente local; (3) a equipa organiza o cronograma de trabalho conseguir melhor entrega do incremento de software."

É neste pontos que se deve formar uma equipa ágil preparada para enfrentar as dificuldades e conseguir ter sucesso.

## 2.2 Modelos Ágeis de Processo

Não é intenção documentar todas as abordagens, cada uma delas dava para um relatório, descreve-se de seguida as principais características dos processos mais conhecidos.

### 2.2.1 Extreme Programming (XP)

O XP usa uma abordagem orientada a objectos como seu paradigma de desenho. O processo é composto por quatro actividades: Planeamento, Projecto, Codificação e Teste, que são repetidas iteração a iteração.

Planeamento: É criado pelo cliente, um conjunto de histórias que descrevem características e funcionalidades necessárias para o software a ser construído. Cada história dá entrada no sistema de controlo da metodologia e é indexada e o cliente atribui-lhe um valor de prioridade. Os membros da equipa analisam esta lista e atribui-lhe custos, se a história precisar de mais de três semanas, pede-se ao cliente que a divida. Novas histórias podem ser adicionadas a qualquer momento. O próximo passo é a equipa em colaboração com o cliente decidir que histórias vão ficar prontas na próxima iteração e em que data.

Projecto: A filosofia inerente é KIS (*keep it simple*), é desencorajado o desenvolvimento de uma funcionalidade extra, porque o programador *developer* acha que mais tarde deve ser precisa. Frequentemente geram-se protótipos, operacional de partes do projecto ou da totalidade. O XP encoraja a *refrabricação*, uma técnica de construção/projecto. (é o processo de alterar e aperfeiçoar o sistema de software interno, sem que se altere o comportamento externo.)

Codificação: Antes do código, recomenda o processo, que se crie uma bateria de testes unitários para que a história fique satisfeita. Então o foco dos programadores é a satisfação destes testes unitários. Para a codificação o XP, recomenda que esta seja feita em pares. (Duas cabeças trabalham melhor do que uma), isto garante outros aspectos como qualidade, e rapidez (existe algum trabalho cientifico que comprava que o trabalhar em pares, não prejudica o

rendimento, pelo contrario habitualmente consegue-se mais produtividade).

Teste: Os testes unitários, são mantidos ao longo das várias iterações e passam a fazer parte de uma bateria de testes de regressão, que não é mais do que todos os testes unitários agrupados para serem testados periodicamente de uma vez em períodos curtos (pode ser de horas, ao final do dia, final da semana). A ideia é confirmar que nada deixou de funcionar.

### 2.2.2 Scrum

Foi desenvolvido inicialmente por Jeff Sutherland e por sua equipa no início da década de 1990. O Scrum, usa um conjunto de "padrões de processo de software", que são adequados para projectos com prazos apertados e requisitos que mudam frequentemente. Cada padrão de processo define um conjunto de actividades:

Pendência: Consiste numa lista priorizada de requisitos ou características do projecto que fornecem valor de negocio para o cliente. O gerente avalia e define prioridades quando necessário.

Sprints: Consiste em unidades de trabalho que são necessárias para satisfazer um requisito definido na pendência, num determinado período de tempo (tipicamente 30 dias, e durante este tempo os items em pendência relacionados com as unidades de trabalho, não podem ser mexidos). Quer-se um ambiente estático a curto-prazo para os trabalhadores.

Reuniões Scrum: São reuniões curtas ( cerca de 15 minutos) realizadas diariamente pela equipa Scrum, onde todos respondem a três perguntas-chave:

- O que a membro fez desde a última reunião da equipa?
- Que obstáculos o membro está a encontrar?
- O que é que o membro vai realizar até à próxima reunião da equipa?

Demos: Entrega da nova *release* funcional ao cliente, de forma a que as novas funcionalidades possam ser testadas e avaliadas pelo cliente.

Estas frases são iteradas até que se dê o projecto de software por concluído, quer seja porque já cumpre as necessidades ou por outros factores, como uma necessidade de entrada antecipada no mercado.

### 2.2.3 Feature Driven Development ( FDD )

Começou por ser concebido por Peter Coad e seus colegas, e mais tarde Stephen Palmer e John Felsing estenderam e melhoraram o processo orientado a objectos que pode ser aplicado a projectos de software de tamanho moderado e grande.

As Features ou “Características “ são funções que o cliente valoriza e que podem ser implementadas em menos de duas semanas, e tem um formato próprio para serem descritas. Há então uma serie de passos até que se cumpra a realização da característica. E antes é preciso também agrupa-las e prioriza-las à semelhança dos outros processos ágeis anteriores.

### 2.2.4 Dynamic Systems Development Method ( DSDM)

É um processo que tenta fornecer maneira de construir e manter sistemas que satisfazem às restrições de prazo apertadas por meio do uso de prototipagem incremental em um ambiente controlado de projecto. Este processo segue o princípio de Pareto 80-20, em que neste caso 80% de uma aplicação pode ser entregue em 20% do tempo que levaria a entregar a aplicação completa (100%).

Este processo é também iterativo como os anteriores, agora pelo facto de seguir o princípio referido, isto é, apenas um certo trabalho é necessário para que cada incremento facilite o avanço para o incremento seguinte. Os detalhes podem ser completados depois, quando mais requisitos forem conhecidos.

As iterações são feitas segundo o ciclo de vida DSDM, que também se encontra documentado no livro [21].

## 2.3 Justificações acerca das empresas presentes no documento

Existe investigação por parte de privados e universidades na área de metodologias e ferramentas ágeis, uma das empresas que mereceu especial destaque neste documento, pois é frequentemente destacada na impressa, em empresas de TI e consultores, foi a Outsystems [10] que ainda recentemente foi distinguida com mais um prémio, na categoria de “*Best Software Development Solution*” dos “*24th Annual CODiE Awards da Software & Information Industry Association's (SIIA)”*.[25].

Trata-se de uma empresa multinacional de software nascida em 2001, reconhecida como construtora de uma das melhores aplicações para a construção e aplicação das metodologias ágeis “industry-leading Agile Platform”[25]. Este prémio na CODiE é de grande importância e salienta o seu reconhecimento, dentro do sector de TI “CODiE Awards-the industry’s only peer-reviewed awards program”. A “Outsystems Agile Platform”, fornece ferramentas para integrar, montar, publicar e mudar aplicações para a Web usando métodos ágeis, estando presente em cerca de 4000 ambientes de TI e 138 países, como tal mereceu uma investigação mais detalhada e isso reflecte-se no documento.

As restantes empresas, faladas foram localizadas ao se pesquisar pelo tema, e foram consideradas como potenciais concorrentes, devido a alguns números apresentados, o Vision Project [17], um projecto da empresa Sueca Visionera AB formada em 2003, que afirma que mais de 55 mil utilizadores usam a plataforma

em 35 países diferentes.

O TargetProcess foi criado em 2004 e tinha em Junho de 2008 cerca de 300 clientes empresariais, e o OnTime da Axosoft ronda as 7000 instalações das ferramentas ágeis. Os números revelam que as companhias têm alguma cota de mercado, e foram merecedoras de serem melhor avaliadas. Após posteriores investigações, outra produto muito usado e associado a companhias de renome deveria ter sido incluído nas análises posteriores, *Version One – Agile Project management*[26], incluído em mais de 10 mil equipas, estimando-se 70 mil utilizadores espalhados por 50 países.

Ao nível das faculdades, algumas tem investigado associadas a determinadas empresas, por exemplo, recentemente a Carnegie Mellon em parceria com a Outsystems procurou melhor a experiência dos utilizadores em contacto com a plataforma ágil, acção essa que foi valorizada e integrada pela Outsystems para a incorporação dos resultados numa próxima *release*.[27]

## 2.4 Vantagens e desvantagens

Em particular, os 'Métodos Ágeis' reduzem o tempo da entrega da primeira versão do *software* pedido. Como o cliente vai verificar mais cedo o que realmente foi produzido e vê se é mesmo aquilo que pretende, o número de projectos falhados por não corresponderem aos desejos do cliente é muito reduzido.

Os 'Metodos Ágeis' seguem um processo iterativo de desenvolvimento e de sucessivas entregas ao cliente, o qual vai constatando a evolução e participando na avaliação e definição das novas funcionalidades a acrescentar. Todo o processo está virado para responder a esta evolução das necessidades e adaptabilidade.

A empresa Outsystems [10] no *paper* [11] comparou as duas metodologias, relativamente aos ganhos de valor para o cliente e *stakeholders,* representado a essa informação sob a forma de gráfico [Figura 1]. Da análise do gráfico verifica-se que no período inicial os "Métodos Ágeis" entregam uma primeira versão que satisfaz os requisitos mínimos, enquanto a 'Metodologia Waterfall', considerando o mesmo momento, ainda está em evolução, normalmente sem nada funcional para entregar ao cliente. Uma vez que, nos 'Métodos Ágeis', o produto/*software* está a ser produzido de acordo com o valor que o cliente pretende, vai entregando ao longo do tempo uma versão mais adequada. Pelo contrário, no modelo de 'Waterfall', existe uma longa planificação que atrasa a entrega de algo visível, e essa entrega inicial, segundo a Outsystems, é a que tem maior correspondência com o que o cliente requisita; posteriormente, os requisitos tendem a mudar, a adaptação do produto a estas mudanças é difícil (pode ser mais morosa, envolver maiores custos, por exemplo), o que implica um retorno menor de valor para os clientes e *stakeholders*. O que se comenta aqui vai no sentido certo, o gráfico é que está consideravelmente exagerado, podemos ver na [Figura 2] um gráfico menos tendencioso, em que partem do mesmo ponto e chegam ao mesmo resultado mas com percursos diferentes. Tendo a acreditar que o correcto para projectos adequados a estes processos ágeis gerasse

habitualmente gráficos semelhantes aos da [Figura 1], mas com o modelo de waterfall mais próximos dos resultados do processo ágil.

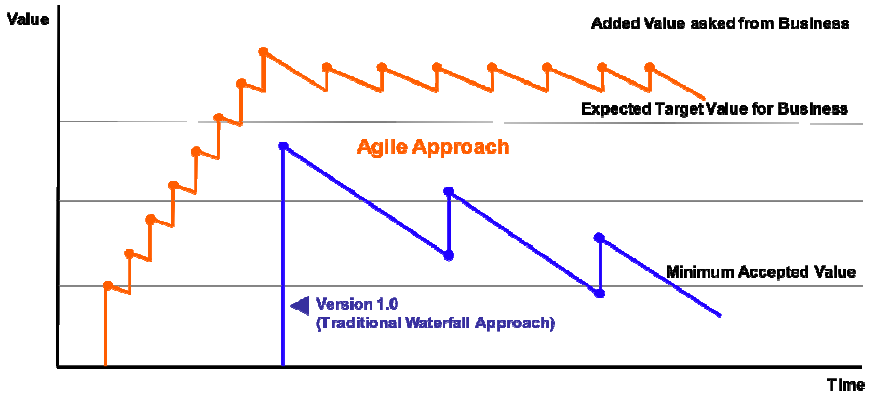
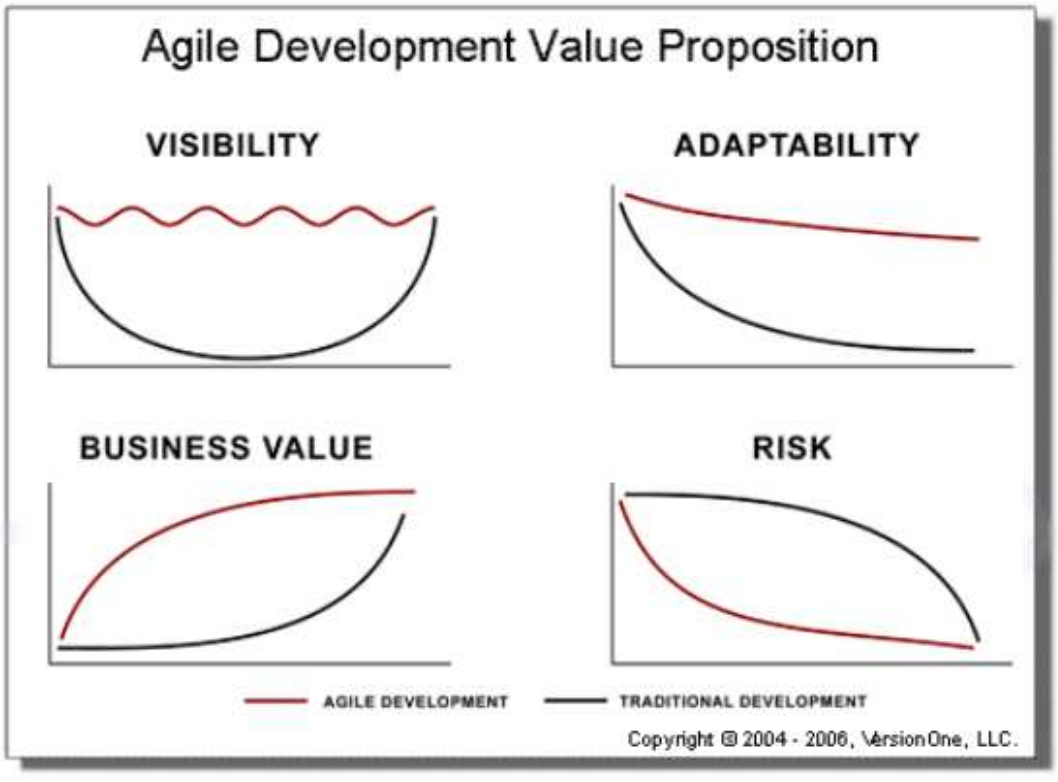



Diagram A: In a traditional waterfall model, software value decreases as time passes. Conversely, with an Agile approach, IT quickly delivers new functionality on time and with significant value. Computing this in terms of daily value provides the total benefit of ownership for the application.

Figura 1




Agile Development Value Proposition

Figura 2

Outra vantagem da 'Metodologia Ágil' é o aumento do controlo por parte dos gestores, uma vez que se baseia no que está realmente a ser produzido e no

que vai ser feito a curto prazo. Como tal, há menos especulação, há mais visibilidade e adequação das medições e avaliações do estado das funcionalidades e tarefas realizadas.

Este método aproxima os *developers* e gestores pois existe uma maior e melhor comunicação (bom ambiente pode ser sinónimo de maior produtivade dos indivíduos). É especialmente adequado para projectos direccionados para a Web, onde os requisitos vão evoluindo e não se exigem muitos trabalhadores. A maioria dos relatórios e documentação são produzidos pelas ferramentas de trabalho, o que alivia as equipas de trabalho.

Uma desvantagem apontada aos 'Métodos Ágeis' é o facto de estes não serem escaláveis. Na realidade, estes não foram desenhados para projectos muito longos (uma discussão sobre isso pode ser encontrada aqui [12]), existindo contudo abordagens mais escaláveis, como o Scrum (que é mais adaptado a esta necessidade mas também serve para as de menor porte). Alistair Cockburn e Jim Highsmith afirmam no *paper* [22] *"Agile development is more difficult with larger teams. The average project has only nine people, well within the reach of the most basic agile processes. Nevertheless, it is interesting to occasionally find successful agile projects with 120 or even 250 people".*

Outra possível desvantagem dos 'Métodos Ágeis' passa pelo menor controlo de custos. Tipicamente, nesta metodologia, o projecto termina quando o cliente não levantar mais funcionalidades relevantes que deseje ver concretizadas, em oposição a ser acordado um preço e um plano. Daqui tira-se que os custos e durações podem variar e podem ser de difícil gestão para a organização.

## 2.5 Método de escolha

O 'Método baseado na análise do Risco' [13] tenta integrar quando necessário os pontos fortes dos 'Métodos Ágeis' e 'Plan-driven'. Este método está documentado com precisão no livro [14] e é usado como maneira de avaliar o risco inerente a usar um 'Método Ágil' ou 'Plan-Driven' para determinado projecto:

Passo 1: Aplicar as análises de risco a áreas especificas associadas às respectivas metodologias. Isto forma a primeira base de análise.

Passo 2: Avaliar os resultados do passo anterior e verifica-se se é um bom projecto para ser puramente 'Ágil' ou 'Plan-driven'. Caso o seja, deve-se ir para o passo 4.

Passo 3: Estamos neste passo se o projecto não foi identificado como sendo ideal apenas para uma metodologia, o que significa que partes do projecto têm associados valores de risco diferentes. Assim sendo, a equipa de desenvolvimento poderá, se possível, desenvolver uma arquitectura que suporta 'Métodos Ágeis' onde se aplicam melhor os seus pontos fortes e os riscos são

minimizados. A metodologia, por defeito, seria a 'Plan-driven', que suportaria o restante trabalho.

Passo 4: Aqui definem-se resoluções para os riscos assumidos e desenha-se o projecto.

Passo 5: É importante nesta metodologia complexa estar atento às partes individuais do projecto, e a ele como um todo, o que se torna impossível sem uma equipa experiente. Pode ser necessário voltar uns passos atrás se se detectar algum problema.

Esta é uma abordagem que não visa a simples escolha de um método ou do outro, na maioria das organizações escolhe-se pelas vantagens e desvantagens referidas anteriormente. Esta análise visa apenas referir que é possível integrar os 'Métodos Ágeis' e os 'Plan-driven', embora isso não seja acessível à maioria das organizações.

# 3 Introdução do processo Ágil numa organização

É sabido que o processo de transição ou adopção da 'Metodologia Ágil' não é muito fácil, uma vez que para haver sucesso é importante que todos os intervenientes estejam focados neste modo de trabalhar.

MikeCohn e Doris Ford, em [15], explicitam resultados muito interessantes, resultantes da introdução dos métodos em sete empresas durante quatro anos em quatro estados diferentes (particularmente usando a ferramenta Scrum), havendo diversidade no tamanho das equipas, na complexidade e distribuição dos membros (equipas distribuídas ou locais).

Constatou-se que a maioria dos *developers* reagiu à mudança com uma mistura de cepticismo, entusiasmo e cautela, verificando-se em alguns o gosto por contribuir com outros artefactos, para além do código. Numa abordagem ágil que não incentiva a produção destes documentos, esta iniciativa deve ser cortada prioritariamente pelos colegas, que tendo uma visão global do projecto, percebem que é trabalho desnecessário para a implementação daquela funcionalidade. Abordagens como Scrum ou XP aceleram os ciclos de projecto fazendo os *developers* interagir mais com os gestores em períodos mais curtos. Os *developers* tendem a ver esta aproximação como controlo das datas de entrega e dos prazos excedidos. O gestores devem mostrar que o seu desejo é remover obstáculos e não criticar se a tarefa é mais longa do que o esperado. Outra constatação, foi o gosto/necessidade de os *developers* conseguirem visualizar a sequência de trabalho que têm para realizar. Nos modelos 'Heavyweight/Plan-driven' é frequente serem usados gráficos de Gantt sobre processos. Ora, esse formato não existe nesta metodologia, pelo que é sugerido por Cohn e Ford que provisoriamente sejam definidos *sprint-types*. Um sprint é uma iteração, de onde sai uma versão executável do projecto para o cliente, contendo várias funcionalidades. Os sprints variam de duração (1semana -1 mês), dependendo dos projectos.

Estes *sprint-types* demonstram uma sequência semelhante a outros métodos mais antigos ('*Prototyping'*, *'Requirements capture', 'Analysisand design', 'Implementation and Stabilization'*), sendo que o uso desta formalização ajuda no processo de adaptação e de aprendizagem. À medida que se adaptam melhor à formalização dos "Métodos Ágeis", este processo intermédio é abandonado. Este modelo foi concretizado com sucesso nas organizações em que foi testado.

Nos 'Métodos Ágeis', as decisões são tomadas mais rapidamente e há mais comunicação. Pelas tentativas já realizadas, para esta metodologia, não se recomenda aceitar projectos distribuídos nos primeiros 2 a 3 meses de conversão das equipas. Há a necessidade de resolver questões políticas e culturais e isso é complicado em projectos distribuídos. Se houver essa necessidade, recomenda-se que o máximo possível de intervenientes se reúna e

trabalhem juntos no projecto durante uma ou duas semanas, num mesmo local.

Quando a metodologia está enraizada a equipa trabalha bem, e trabalha sobre questões essenciais. Bons profissionais são necessários e ajudam o gestor a distribuir e calcular melhor os tempos para realizar determinadas tarefas. Grandes discrepâncias não são recomendáveis.

Numa abordagem 'Plan-driven' os testes são realizados de uma vez e maioritariamente no final do projecto por *testers*. Numa 'Metodologia Ágil' deve-se evitar que os ciclos curtos sejam aproveitados para que os *testers* se ponham a realizar código ou escrevam os *unittests,* o que deve ser feito pelo próprio programador que está a desenvolver, devido ao conhecimento que tem do seu código.

Os gestores de topo revelaram dificuldade relativamente a orçamentos para um projecto que use 'Métodos Ágeis', pois não há um plano com prazos e custos, este dá-se por terminado quando as funcionalidades presentes satisfizerem o cliente. É recomendado que se definam limites mínimos (e.g. prazo, *budget*) e, perto do final do projecto, haja uma renegociação, consoante as necessidades do cliente.

# 4 Ferramentas que suportam a Metodologia Ágil

Nesta secção pretende-se analisar um pouco as ferramentas disponíveis no mercado que facilitem o desenvolvimento de *software*, segundo a "Metodologia Ágil".

*'Agileplatform'* [16]:

Produzida pela Outsystems parece destacar-se ao ser uma solução unificada que suporta todo o processo de construção para a Web com recurso a 'Métodos Ágeis'.

A plataforma é composta por quatro componentes: *IntegrationStudio, ServiceStudio, ServiceCenter e EmbeddedChangeTechnology (ECT).* O primeiro integra componentes que não estão criados no *ServiceStudio* e que precisam de ser adaptados para serem potenciados na plataforma.

O segundo dá um novo significado ao termo "ágil", e é visível a maior facilidade e rapidez com que se integram objectos e modelos através de fluxos do que sendo tudo codificado. Este ambiente reúne o modelo de dados, a lógica do negócio, a criação das páginas Web, *Web services*, temporizadores, e segurança, que com um *'click'* serão convertidos em código java ou dot.net.

O *ServiceCenter* controla tudo o que seja configurações e permissões do projecto, e monitoriza a sua actividade, erros, etc.

O ECT é um mecanismo que permite receber directamente *feedback* a partir

da aplicação. No decorrer da mesma, selecciona-se a ferramenta, o local que diz respeito à alteração que o utilizador queria ver realizada e descreve-a. Este comentário chega sobre a forma de *tickets* aos *developers* e gestores do projecto, para que seja classificado, atribuída uma ordem de prioridade, e integrado nos próximos *sprints.*

*VisionProject* [17]:

Esta parece ser uma ferramenta bastante completa quanto a relatórios e gestão do projecto, só que não torna a aplicação e esta gestão um só. Permite a gestão de projectos, gestão dos *tickets*, controlo de versões, visualização de durações e custos, partilha de documentos, procura e filtragem dos *tickets*, integração do e-mail e gestão dos fluxos de trabalho. Para além destas, destaca-se a gestão de alertas, bases de conhecimento, fóruns por projecto, configurações, exportação de relatórios e *helpdesk*.

*Pivotaltracker* [18]:

Este *software* apresenta-se como uma alternativa mais simples do que o *Vision*, mas assenta nas mesmas bases, controlando toda uma série de *tickets* e gerindo o seu estado. Incorpora ainda toda uma série de material para o controlo do projecto e produção de relatórios de estado.

*TargetProcess* [19]:

Muito completo e personalizável, parece superar o *VisionProject* e o *Pivotaltracker* em toda a gestão e permite, com recurso a *plugins*, a automatização de certos *inputs* ou *outputs* com as ferramentas de produção de código e teste. Isso é uma característica muito importante, pois nesta 'Metodologia Ágil' onde a produção de código deve ser o foco principal (código, ou algo automático que o gere), é vantajoso que o processo seja o mais automatizado possível, e que implique o menor desperdício de tempo possível.

Existiam mais *softwares*, como por exemplo o da 'OnTime Project Manager Suite' da "*Axosoft*" [20], e o da Version One[26] que podiam ter sido escolhidos para análise; para quem quiser experimentar estes programas e os anteriores referenciados têm opções gratuitas, com limite de tempo ou de utilizadores ou de funcionalidades, que permitem avaliar melhor o conceito na prática. Da exploração, se a intenção for reduzir drasticamente o tempo de produção e se essa vontade for suportada por toda a empresa, parece que vale a pena explorar a ferramenta mais completa e automatizada fornecida pela '*Outsystems*' [16]. Esta ferramenta é muito completa, extensível e focada em tornar ágil todo o processo de criação. No geral, estas ferramentas ágeis favorecem uma entrada mais rápida no mercado e possibilitam que novos requisitos sejam tratados em

horas. Para casos com necessidades mais localizadas, ou com processos desenvolvidos em outros programas, se calhar só é preciso um número mais restrito de funcionalidades e poderá ser encontrado numa das outras ferramentas referenciadas ou ser encontrada uma solução que se integre mais nas ferramentas já usadas.

# 5 Conclusões

Este *working paper* explica as principais fases do processo de introdução/transição da 'Metodologia Ágil', quer seja XP, Scrum, ou outra. Estes métodos só têm valor quando realmente adequados aos tipos de projectos analisados. Maioritariamente projectos para a *Web* e de pequena e média dimensão. Mostra também que a escolha por um método ou outro de desenvolvimento de *software* pode ser feita pela análise de risco para projectos não tão óbvios quanto à melhor metodologia a usar, embora seja de difícil utilização. Para a sucesso da sua implantação a equipa não pode ser descurada, vimos quais as características mais importantes que esta deve ter na secção "2.1 Equipa"

O mercado está cada vez mais receptivo a esta prática e produz cada vez mais material para facilitar a sua adopção. Das aplicações, destaca-se, como foi falado na secção 4, a da '*Outsystems',* sendo ágil até no momento de concepção da própria, acelerando rapidamente a entrega de versões com qualidade e testáveis pelo cliente.

Continua a existir investigação sobre estas metodologias recentes, e na generalidade existe a tentativa de resolver os problemas que se vão encontrando no ambiente empresarial. Um deles era a falta de usabilidade nas aplicações geradas, visto que o foco era em código, no entanto, surgiram propostas para juntar os métodos ágeis e os princípios de usabilidade de Jakob Nielsen's [24]. É uma tendência integrar estas metodologias com outros princípios a fim de as completar e tornar mais poderosas.

# 6 Referências


[1] K. Beck et al., "Manifesto for Agile Software Development", Feb.2001, http://www.agilemanifesto.org, consultado em 31/05/2009.

[2] Ronald E. Jeffries, XProgramming - an Agile Software Development Resource, http://www.xprogramming.com/, consultado em 31/05/2009.

[3] Don Wells, "Extreme Programming – A gentle introduction", http://extremeprogramming.org/, consultado em 31/05/2009.

[4] "ScrumAlliance – transforming the world of work", http://www.scrumalliance.org/, consultado em 31/05/2009.

[5] "SCRUM it's About Common Sense", http://www.controlchaos.com/, consultado em 31/05/2009.

[6] "Feature Driven Development", http://www.featuredrivendevelopment.com/, consultado em 31/05/2009.

[7] "DSDM – CONSORTIUM", http://www.dsdm.org, consultado em 31/05/2009.

[8] "Lean Software Institute", http://www.leansoftwareinstitute.com/, consultado em 31/05/2009.

[9] Alistar Cockburn, "Software development as a cooperative game", http://alistair.cockburn.us/Software+development+as+a+cooperative+game, consultado em 31/05/2009.

[10] Outsystems, http://www.outsystems.com/, consultado em 31/05/2009.

[11] Outsystems, "Transitioning to Agile in an AgileWay – Amplifying Traditional Approches with Agile Technology", http://www.outsystems.com/agile/Solution.aspx?FolderPath=\Root\Contents\Corporate\ITSolutions\AgileIT, consultado em 31/05/2009.

[12] "InfoQueue – Tracking change and innovation in the enterprise software development community", http://www.infoq.com/news/2007/07/agile_team_size, disponível em 31/09/2009.

[13] BarryBoehm, RichardTurner, "Using Risk to Balance Agile and Plan-Driven Methods", IEE Computer Society, Junho 2003.

[14] B. Boehm and R. Turner,"Balancing Agility and Discipline - The Guide for the Perplexed", Addison-Wesley, 2003.

[15] Mike Cohn, Doris Ford, "Introducing an Agile Process to an Organization", IEEE Computer Society, Junho 2003

[16] OutSystems, "Agile Platform", http://www.outsystems.com/agile/Solution.aspx?FolderPath=\Root\Contents\Corporate\ITSolutions\AgilePlatformSolution, consultado em 31/05/2009.

[17] "Vision Project", http://www.visionproject.se/, consultado em 31/05/2009.

[18] "Pivotal Tracker", http://www.pivotaltracker.com/, consultado em 31/05/2009.

[19] "TargetProcess – Agile Project management Software", http://www.targetprocess.com/, consultado em 31/05/2009.

[20] Axosoft, "OnTime Project Manager Suite", http://www.axosoft.com/, disponível em 31/05/2009.

[21] Roger S. Pressman, "Engenharia de Software – Sexta Edição", McGraw-Hill, 2006, pp. 58-76

[22] Cockburn, A. e Highsmith, "Agile Software Development: The People Factor", IEEE Computer, v.34, n.11, nov. 2001, p. 131-33

[23] DeMarco, T. E Lister, T., Peopleware, 2ªed., Dorset House, 1998.

[24] Jakob Nielsen, "Agile Development Projects and Usability", http://www.useit.com/alertbox/agile-methods.html, disponível em 08/06/2009

[25] San Ramon, OutSystems 'Agile Platform' Wins CODiE Award Category for Best Software Development Solution, http://www.reuters.com/article/pressRelease/idUS185847+08-May-2009+BW20090508, 8/05/2009

[26] VersionOne – Simplifing Software Delivery, http://www.versionone.com/, disponível em 06/07/2009

[27] Santa Clara, Carnegie Mellon e Outsystems como parceiros, http://www.outsystems.com/agile/Content.aspx?ContentName=CMU_PR&FolderPath=/Root/Contents/Corporate/LandingPages/News, 10/03/2009