a partir dos mais críticos ou complexos para os mais triviais e simples, desenvolve-se, em um primeiro momento, todos os elementos de maior risco para a arquitetura, não deixando muitas surpresas para depois.

5.1.3 Iterativo e Incremental

Assim como nos métodos ágeis, o UP preconiza o desenvolvimento baseado em ciclos iterativos de duração fixa, em que, a cada iteração, a equipe incorpora à arquitetura as funcionalidades necessárias para realizar os casos de uso abordados no ciclo.

Cada ciclo iterativo produz um incremento no *design* do sistema, seja produzindo mais conhecimento sobre seus requisitos e arquitetura, seja produzindo um código executável. Espera-se que, em cada iteração, todas as disciplinas previstas sejam executadas com maior ou menor intensidade (ver. Figura 5.1). Então, assim como nos métodos ágeis, cada ciclo iterativo vai implicar executar todas as atividades usuais de desenvolvimento de software.

A integração contínua reduz riscos, facilita os testes e melhora o aprendizado da equipe sobre o sistema, especialmente nos primeiros momentos, quando decisões críticas precisam ser tomadas com pouco conhecimento sobre o sistema em si.

Em geral, a fase de concepção, por ser curta, é executada em um único ciclo. Já as fases de elaboração, construção e transição podem ser executadas a partir de uma série de ciclos iterativos. Porém, no caso de projetos muito grandes, a fase de concepção pode ser subdividida em ciclos nos quais se explorem diferentes características de um sistema.

5.1.4 Focado em Riscos

Em função das priorizações dos casos de uso mais críticos nos primeiros ciclos iterativos, pode-se dizer que o UP é focado em riscos. Se esses casos de uso são os que apresentam maior risco de desenvolvimento, então devem ser tratados o quanto antes para que esse risco seja resolvido enquanto o custo para tratá-lo ainda é baixo e o tempo disponível para lidar com as surpresas é relativamente grande.

Esse tipo de abordagem (tratar primeiro os problemas mais difíceis) tem sido um valor incorporado a vários modelos de desenvolvimento modernos. Os requisitos ou casos de uso de maior risco são os mais imprevisíveis. Assim, estudá-los primeiramente, além de garantir maior aprendizado sobre o sistema e decisões arquiteturais mais importantes, vai fazer que riscos positivos ou negativos sejam dominados o mais cedo possível (um risco positivo é, por exemplo, o sistema ser mais simples do que inicialmente imaginado).

5.2 Fases do Processo Unificado

O Processo Unificado divide-se em quatro grandes fases:

a) *Concepção* (*inception*): trata-se da elaboração de uma visão em abrangência do sistema. Nessa fase são levantados os principais requisitos, um modelo conceitual preliminar é construído, bem como são identificados os casos de uso de alto nível (Wazlawick, 2011), que implementam a funcionalidade requerida pelo cliente. Na fase de concepção calcula-se o esforço de desenvolvimento dos casos de uso e constrói-se o plano de desenvolvimento, composto por um conjunto de ciclos iterativos nos quais são acomodados os casos de uso. Pode haver alguma implementação e teste, caso seja necessário elaborar protótipos para redução de risco.
b) *Elaboração* (*elaboration*): nessa fase, as iterações têm como objetivo, predominantemente, detalhar a análise, expandindo os casos de uso, para obter sua descrição detalhada e situações excepcionais (fluxos alternativos). O modelo conceitual preliminar será transformado em um modelo definitivo, cada vez mais refinado, sobre o qual serão aplicados padrões de análise e uma descrição funcional poderá ser feita, bem como o *design* lógico e físico do sistema. Um código será gerado e testado. Contudo, essas atividades não são as que vão ocupar a maior parte do ciclo iterativo, pois haverá proporcionalmente mais carga de trabalho em análise e *design* do que em codificação e teste.
c) *Construção* (*construction*): a fase de construção possui iterações nas quais os casos de uso mais complexos já foram tratados e a arquitetura já foi estabilizada. Assim, as atividades de suas iterações consistem predominantemente na geração de código e testes do sistema. Com a automatização da geração de código e a introdução de modelos de desenvolvimento dirigidos a teste, pressupõe-se que um bom *design* possa dar origem rapidamente a um código de alta qualidade.

d) *Transição* (*deployment*): a fase de transição consiste na implementação do sistema no ambiente final, com a realização de testes de operação. Também é feita a transferência de dados de possíveis sistemas antigos para o novo sistema e o treinamento de usuários, bem como outras adaptações, que variam de caso para caso. Nessa fase pode haver ainda alguma revisão de requisitos e geração de código, mas não de forma significativa.

Apesar de as fases do Processo Unificado terem diferentes ênfases, espera-se que cada ciclo iterativo tome um conjunto de casos de uso e os desenvolva desde os requisitos até a implementação e a integração de código final. Na fase de elaboração a equipe necessariamente trabalhará mais tempo em questões de análise e projeto do que de implementação e teste. Na fase de construção, porém, os requisitos já terão sido em grande parte desvendados e o esforço recairá mais nas atividades de programação.

Uma das características do UP é o fato de que, a cada fase, um macro-objetivo (*milestone*) é atingido. Ao final da fase de concepção, o objetivo é ter entendido o escopo do projeto e planejado seu desenvolvimento. Ao final da fase de elaboração, os requisitos devem ter sido extensivamente compreendidos e uma arquitetura estável deve ter sido definida. Ao final da fase de construção, o sistema deve estar programado e testado. Ao final da fase de transição, o software deve estar instalado e sendo usado pelos usuários finais (West, 2002).

5.2.1 Concepção

No Processo Unificado, a fase de concepção não deve ser muito longa. Recomenda-se um período de duas semanas a dois meses, dependendo da dimensão relativa do projeto.

Nessa etapa os requisitos de projeto são analisados da melhor forma possível, em abrangência, e não em profundidade. É importante que o analista perceba claramente a diferença entre as necessidades lógicas e tecnológicas do cliente e os projetos de implementação que ele poderia fazer. A ideia é que o analista não polua a descrição dos requisitos com possibilidades tecnológicas de implementação que não foram expressamente requisitadas pelo cliente.

O UP espera que, nessa fase, sejam estabelecidos também os casos de uso. Há basicamente três formas de agir em relação a isso:

a) Obter casos de uso a partir de uma organização dos requisitos funcionais, em que cada grupo de requisitos poderá dar origem a um ou mais casos.
b) Inicialmente, proceder a uma análise de cenários e, posteriormente, extrair deles os casos de uso, ou seja, os requisitos.
c) Trabalhar apenas com casos de uso, sendo eles a única expressão dos requisitos, não havendo outro documento de requisitos, exceto para os requisitos suplementares (Wazlawick, 2011).

De posse de um conjunto significativo de casos de uso, a equipe deve proceder como nas fases iniciais de planejamento do *Scrum* ou do XP, priorizando os casos de uso mais complexos e críticos em detrimento dos mais simples e triviais. Isso pode ser feito, inicialmente, por uma análise bem simples:

a) Descubra os casos de uso que são meros relatórios, ou seja, que não alteram a informação armazenada, e classifique-os como os mais simples de todos.
b) Verifique se existem casos de uso que seguem algum padrão conhecido, como CRUD (Wazlawick, 2011), mestre-detalhe etc. Esses casos serão de média complexidade, pois, embora possuam um comportamento bem conhecido e padronizado, podem esconder algumas surpresas nas suas regras de negócio, ou seja, inclusões, alterações e exclusões que só poderão ser feitas mediante determinadas condições (ver também Seção 7.5.9).
c) Os demais casos de uso serão considerados os mais complexos. É recomendável que eles sejam organizados do mais importante para o menos importante em relação ao negócio da empresa. O analista deve se perguntar qual dos processos é o mais crítico para o sucesso da empresa e, em função disso, fazer a ordenação. Por exemplo, uma venda possivelmente será mais importante do que uma compra, uma compra mais importante do que uma reserva, uma reserva mais importante do que um cancelamento de reserva etc.

Na fase de concepção, a equipe deve produzir estimativas de esforço para casos de uso, que serão explicadas melhor na Seção 7.5. A partir desse cálculo, deve-se fazer um planejamento de longo prazo, procurando acomodar os casos de uso de acordo com sua prioridade nos diferentes ciclos durante o processo de desenvolvimento. Contudo, esse planejamento não deve ser muito detalhado.

Apenas o primeiro ciclo deve ter um planejamento mais detalhado, com atividades determinadas de acordo com o processo em uso e os tempos, responsáveis e recursos para a execução de cada atividade bem definidos. Os demais ciclos só deverão ter seu planejamento detalhado pouco antes de serem iniciados.

A fase de concepção envolve também o estudo de viabilidade, pois, ao final dela, analisadas as questões tecnológicas, de orçamento e de cronograma, a equipe deve decidir se é viável prosseguir com o projeto.

O marco final (*milestone*) da fase de concepção é conhecido como LCO, ou *Lifecicle Objective Milestone* (marco do ciclo de vida).

5.2.2 Elaboração

A fase de elaboração consiste no detalhamento da análise e da realização do projeto para o sistema como um todo. A elaboração ocorre em ciclos, com partes de *design* sendo desenvolvidas e integradas ao longo de cada ciclo. Os principais objetivos dessa fase, segundo Ambler e Constantine (2000), são:

a) Produzir uma arquitetura executável confiável para o sistema.
b) Desenvolver o modelo de requisitos até completar pelo menos 80% dele.
c) Desenvolver um projeto geral para a fase de construção.
d) Garantir que as ferramentas críticas, processos, padrões e regras estejam disponíveis para a fase de construção.
e) Entender e eliminar os riscos de alta prioridade do projeto.

Essa fase permite analisar o domínio do problema de forma mais refinada e definir uma arquitetura mais adequada e sólida. Além disso, a priorização dos casos de uso mais complexos permitirá eliminar ou mitigar os elementos do projeto que apresentam maior risco.

Assim, embora o Processo Unificado também trabalhe com a perspectiva de acomodação de mudanças, procura minimizar seu impacto mitigando riscos e elaborando uma arquitetura o mais próximo possível do necessário para que as funcionalidades requeridas possam ser desenvolvidas.

A fase de elaboração é caracterizada pela exploração dos casos de uso mais complexos, que vão precisar de mais trabalho de análise do que de implementação, já que será necessário entender e modelar seu funcionamento. À medida que esses casos de uso são estudados e desenvolvidos, a arquitetura do sistema vai se formando. Espera-se que a arquitetura esteja estável no momento em que se passar a considerar apenas os casos de uso padronizados, como CRUD e relatórios. Esses casos de uso não devem impactar a arquitetura.

O marco final (*milestone*) da fase de elaboração é conhecido como LCA, ou *Lifecycle Architecture Milestone* (marco da arquitetura).

5.2.3 Construção

Na fase de construção, um produto completo e usável deve estar desenvolvido, testado e adequado para uso pelo usuário final. Essa fase é realizada em ciclos iterativos e o projeto é desenvolvido também de forma incremental, com novas funcionalidades sendo adicionadas ao sistema a cada ciclo.

Segundo Ambler e Constantine (2000a), os principais objetivos da fase de construção são:

a) Descrever os requisitos que ainda faltam.
b) Dar substância ao *design* do sistema.
c) Garantir que o sistema atenda às necessidades dos usuários e que ele se encaixe no contexto geral da organização.
d) Completar o desenvolvimento dos componentes e testá-los, incluindo tanto o software quanto sua documentação.
e) Minimizar os custos de desenvolvimento pela otimização dos recursos.
f) Obter a qualidade adequada o mais rápido possível.
g) Desenvolver versões úteis do sistema.

A fase de construção caracteriza-se pela exploração dos casos de uso de baixa e média complexidade, ou seja, os casos de uso padronizados que não vão impactar na arquitetura. Como esses casos de uso são padronizados, o esforço de análise e *design* será menor nessa fase, ficando a maior parte do trabalho concentrada na implementação e teste dos componentes da arquitetura dedicados a esses casos de uso.

O marco final (*milestone*) da fase de construção é conhecido como IOC, ou *Initial Operational Capability Milestone* (marco da capacidade operacional inicial).

5.2.4 Transição

A fase de transição consiste na colocação do sistema em uso no ambiente final. São necessários testes de aceitação e operação, treinamento de usuários e transição de dados a partir de sistemas antigos, que podem ser capturados automaticamente ou digitados.

Nessa fase, também poderá haver a execução do sistema em paralelo com sistemas legados para verificar sua adequação.

Após a conclusão da fase de transição, o sistema entra em evolução, ou seja, depois de aceito e colocado em operação no ambiente final, ele passa a receber atualizações periódicas de forma a corrigir possíveis erros ou implementar novas funcionalidades necessárias (ver Capítulo 14).

O marco final (*milestone*) da fase de transição é conhecido como PR, ou *Product Release Milestone* (marco da entrega do produto).

5.3 RUP – *Rational Unified Process*

A mais detalhada e mais antiga implementação do UP é conhecida como RUP (*Rational Unified Process*[2]), criada por Booch, Rumbaugh e Jacobson através da empresa Rational, subsidiária da IBM desde 2003. A versão RUP é tão importante que algumas vezes é confundida ou considerada sinônima de UP.

Ainda antes de pertencer à IBM, a empresa Rational adquiriu, em 1997, várias outras empresas: Verdix, Objectory, Requisite, SQA, Performance Awareness e Pure-Atria. A partir da experiência acumulada dessas empresas, estabeleceu algumas práticas, que seriam a base filosófica para o novo modelo de processo, mais tarde conhecido como RUP (Kruchten, 2003):

a) *Desenvolver iterativamente tendo o risco como principal fator de determinação de iterações*: é preferível conhecer todos os requisitos antes de se iniciar o desenvolvimento propriamente dito, mas em geral isso não é viável, de forma que o desenvolvimento iterativo orientado à redução de risco é bastante adequado.
b) *Gerenciar requisitos*: deve-se manter controle sobre o grau de incorporação dos requisitos do cliente ao produto.
c) *Empregar uma arquitetura baseada em componentes*: quebrar um sistema complexo em componentes não só é necessário como inevitável. A organização permite diminuir o acoplamento, o que possibilita testes mais confiáveis e maior possibilidade de reuso.
d) *Modelar software de forma visual com diagramas*: UML é indicada como padrão de modelagem de diagramas.
e) *Verificar a qualidade de forma contínua*: o processo deve ser o mais orientado a testes quanto for possível. Se for o caso, utilizam-se técnicas de desenvolvimento orientado a testes, como TDD, ou *Test-Driven Development* (Beck, 2003).
f) *Controlar as mudanças*: a integração deve ser contínua e gerenciada adequadamente com o uso de um sistema de gerenciamento de configuração (Capítulo 10).

O RUP é um produto que, entre outras coisas, inclui uma base de dados com *hiperlinks* com vários artefatos e *templates* necessários para usar bem o modelo. Uma descrição do processo em português pode ser encontrada na internet.[3]

5.3.1 Os Blocos de Construção do RUP

O RUP é baseado em um conjunto de elementos básicos (*building blocks*) identificados da seguinte forma:

a) *Quem*: um *papel* (Seção 2.3.2) define um conjunto de habilidades necessário para realizar determinadas atividades.

[2]Disponível em: <www.rational.com/rup/>. Acesso em: 21 jan. 2013.
[3]Disponível em: <www.wthreex.com/rup/portugues/index.htm>. Acesso em: 21 jan. 2013.