UNIVERSIDADE FEDERAL DO PARANÁ

ALEXANDRE HASSELMANN LANGE
DIOGO ALEXANDRO PEREIRA

APRENDIZADO DINÂMICO DE MODELAGENS DE DESENVOLVIMENTO DE SOFTWARE ATRAVÉS DE DINÂMICA DE SISTEMAS

CURITIBA
2015

ALEXANDRE HASSELMANN LANGE
DIOGO ALEXANDRO PEREIRA

APRENDIZADO DINÂMICO DE MODELAGENS DE DESENVOLVIMENTO DE SOFTWARE ATRAVÉS DE DINÂMICA DE SISTEMAS

Trabalho apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação, Departamento de informática, Setor de Ciências Exatas da Universidade Federal do Paraná.

Orientador: Prof. Ms. Setembrino S. Ferreira Jr

CURITIBA
2015

Dedicamos este trabalho aos nossos pais, por tudo que fizeram ao longo de nossas vidas.

AGRADECIMENTOS – Alexandre Lange

Primeiramente a Deus, pois sem ele nada disso seria possível.

Aos meus pais, Francisco e Rosane, por todo o apoio e ajuda na educação de seus filhos.

A minha querida esposa, Rafaele, por todo o carinho, paciência, amor e ajuda nesses anos.

A todos nossos professores que contribuíram e nos deram conhecimento para estarmos aqui, em toda nossa vida acadêmica.

Em especial ao nosso orientador Setembrino, por acreditar na ideia deste trabalho, ajudar nos objetivos e pela paciência.

Ao meu amigo Diogo, coautor desse trabalho, pela enorme contribuição, assistência e esforço nesse trabalho.

Ao meu irmão, Paulo Felipe, e todos os meus amigos que durante essa caminhada estiveram ao meu lado dando suporte e auxílio, e que sempre acreditaram em mim.

AGRADECIMENTOS – Diogo Pereira

A todos nossos professores que contribuíram e enriqueceram nossos conhecimentos em toda nossa vida acadêmica.

Em especial ao nosso orientador Setembrino, por acreditar na ideia desse trabalho, pela paciência e conversas inspiradoras.

Ao meu amigo Lange, coautor desse trabalho, pela coragem de embarcar nesse projeto e contribuição para que chegássemos a este resultado.

Aos meus pais, Leonardo e Maria, pela força e incentivo. Por acreditarem e priorizarem a educação ao longo de minha vida.

Ao meu irmão Diego, um grande amigo em todos os momentos, inclusive na ciência da computação.

Aos meus grandes amigos Phyllipy e Érico, irmãos que escolhi ao longo desta vida.

Ao Felipe e a Marina, ironicamente questionaram a minha capacidade de concluir esta obra. Estranhamente isso serviu de incentivo.

Aos Mavarax: Diegn, Tito, Caio, Junior Tilt, Marxillien, Maguila, Yudi, Jones, Branco, Tomate e Artur, pela amizade indescritível.

Aos grandes amigos do BCC, BronxCabralCapãoXaximAhú, estágio, Shablaw-GG e tantos outros que foram fundamentais ao longo dessa jornada.

“A reader lives a thousand lives before he dies.
The man who never reads lives only one.”
George R. R. Martin

RESUMO

O presente trabalho se propõe a orientar gerenciadores de projetos de desenvolvimento de software a agregar valor internamente durante o processo, para evoluírem de acordo com o seu ambiente e objetivo. Usando um ambiente teórico, utilizaremos o Scrum como metodologia e apresentaremos potenciais variáveis desse sistema. Então, através da Dinâmica de Sistemas, iremos refinar o modelo de desenvolvimento aplicado, aprofundando em análises do sistema como um todo, a fim de ponderar os resultados obtidos. A partir destes resultados teremos algo concreto e aplicável, a fim de aumentar a eficiência do processo, ou detectar o declínio do modelo aplicado e realizar a troca da metodologia antes de atingir níveis ruins de gerenciamento de pessoal e produção. Teremos no final um método PDCA diferenciado, de forma a proporcionar estratégias de melhor compreensão da modelagem ao ambiente inserido, permitindo assim as tomadas de decisões através do acompanhamento contínuo do processo, desta forma, de ciclo em ciclo será possível refinar a modelagem utilizada ou certificar-se de que esse não é o melhor caminho para tal ambiente. Se esse for o caso, já estar pré-analisado o melhor modelo a ser utilizado naquele projeto. Nesse contexto, o gerente de projetos estará hábil para um bom entendimento do seu local de aplicação, entender como trabalham as variáveis ao seu redor e retirar disso uma forma de melhorar seu gerenciamento e resultados.

Palavras-chave: Métodos Ágeis. *Scrum*. PDCA. Dinâmica de Sistemas, Gerenciamento de Projetos. Desenvolvimento de Software.

ABSTRACT

This present work proposes to guide project managers of software development to add value internally during the process to evolve according to your environment and goals. Using a theoretical environment, we use Scrum as a methodology and present potential system variables. Then, through the System Dynamics, we will refine the applied model of development, deepening system analysis as a whole, in order to consider the results. As from these results we have something concrete and applicable, in order to increase process efficiency, or to detect the decline of the model and perform exchange method before reaching poor levels of personnel and production management. We will have at the end a different PDCA method in order to provide better understanding of modeling strategies to the environment insertion, thus enabling decision-making through continuous monitoring of the process, so cycle by cycle will be possible to refine the modeling used or certify itself that this is not the best way to such an environment. If this is the case, is already pre analyzed the best model to be used in that project. In this context, the project manager will be able to have a good understanding of their place of application, understand how the variables work around it and taking from this a good way to improve their management and results.

Keywords: Agile Methods. Scrum. PDCA. System Dynamics. Project Management. Software Development.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E/OU SIGLAS

XP	- Extreme Programming
MA	- Métodos Ágeis
IEEE	- Institute of Electrical and Electronics Engineers
FDD	- Feature Driven Development
RUP	- Rational Unified Process
SW-CMM	- Capability Maturity Model for Software
DuPont	- du Pont de Nemours and Company
IBM	- International Business Machines
PDCA	- plan, do, check, action
5W2H	- *Checklist* de sete passos
DBA	- Database administrator
CEO	- Chief Executive Officer
CV	- Comunicação verbal
CVD	- Comunicação virtual direta
CVI	- Comunicação virtual indireta

LISTA DE SÍMBOLOS

- Número (quantidade)

¶ - Parágrafo

SUMÁRIO

1. INTRODUÇÃO

1.1. TEMA

A busca continua por novas técnicas de processos de desenvolvimento é uma característica de qualquer linha de montagem, no desenvolvimento de software isso não é diferente. As organizações estão sempre procurando novas práticas para melhorar seus processos de desenvolvimento, com isso vários estudos emergem no mercado visando um padrão ideal para esse processo, gerando assim um grande número de novos modelos, porém oque ocorre na maioria das vezes é um equivoco entre modelo e padrão.

Grande parte dos processos de desenvolvimento são descritos como modelagens de desenvolvimento de software, pois são boas práticas que já foram estudadas previamente. Porém cada ambiente de desenvolvimento é único, com perfis diferenciados, tornando assim inviável criar um padrão perfeito para o processo de desenvolvimento de software.

O equivoco ocorre em tentarmos aplicar um modelo de desenvolvimento como sendo um padrão rigoroso a ser adotado. A modelagem é capaz apenas de nos orientar, mas para atingirmos o maior nível de aproveitamento dessas informações, é necessário compreender o modelo e adaptá-lo às nossas necessidades.

1.2. JUSTIFICATIVA

O processo de compreensão do modelo escolhido requer análises apuradas sobre as variáveis do sistema, o que geralmente não ocorre nas organizações. A falta desse levantamento de dados leva as organizações a recorrerem às famosas consultorias, onde agentes externos atuam nessa análise e ajudam a compreender a modelagem aplicada.

A compreensão da modelagem ajuda a estabelecer se o modelo é adequado para as atividades empregadas, assim tornando possível saber se o modelo é viável ou se é o momento adequado para trocar para outro modelo. Porém, apenas uma consultoria não é capaz de refinar a modelagem, pois o processo de desenvolvimento é maleável e necessita de análise constante. Somente através do acompanhamento continuo do processo é que somos capazes de refinar a modelagem utilizada.

1.3. OBJETIVO GERAL

Esse trabalho tem como objetivo geral, apresentar técnicas capazes de analisar metodologias de desenvolvimento de software, com a intenção de refinar o seu uso de acordo com as necessidades da organização. O resultado desse estudo será um método PDCA diferenciado, constituído de estratégias capazes de proporcionar um aprendizado sobre a metodologia adotada para o desenvolvimento do sistema analisado.

1.4. OBJETIVOS ESPECÍFICOS

Apresentar os conceitos de modelagens de software, suas principais características e diretrizes, para que haja uma compreensão sobre o motivo de serem aplicadas. Também será abordado detalhadamente um modelo em especifico, SCRUM, somente para fins didáticos para que possamos acompanhar um processo de análise.

Apresentar os princípios de abordagem sistêmica, para esclarecer como o processo de análise sobre o modelo será efetuado. Detalhar os componentes da abordagem, tais como pensamento e dinâmica de sistemas, para que haja compreensão das técnicas que serão empregadas. Abordar conceitos de aprendizagem, com o objetivo de enfatizar o resultado desse estudo, o qual consiste de uma base de conhecimento.

Apresentar os principais pontos das práticas de PDCA, para que todo o estudo realizado possa ser interpretado e aplicado como um produto. Detalhar as fases do ciclo e seus objetivos, esclarecendo como a evolução do PDCA também resultada na evolução da própria modelagem.

Gerar uma visão diferenciada sobre o processo de utilização de modelagens de desenvolvimento de software, com recursos suficientes para que possam ser objetos de estudos em diversas ramificações da engenharia de software.

1.5. ESTRUTURA DA MONOGRAFIA

Esse trabalho faz uso de conceitos de várias áreas distintas, resultando em inúmeras teorias, por essas características a fundamentação teórica é extensa e abrangente. A fundamentação tem inicio a partir do capítulo de número (2).

Os Subcapítulos da fundamentação teórica são constituídos por informações de Engenharia de Software, Métodos de Gestão e Sistemas, nas identificações (2.1), (2.2) e (2.3), respectivamente.

No subcapítulo de Engenharia de Software (2.1) é possível encontrar informações gerais sobre modelagens de desenvolvimento e características desses processos, assim como informações detalhadas da modelagem SCRUM em (2.1.7), a qual é objeto de estudo desse trabalho.

No subcapítulo de Métodos de Gestão (2.2) é possível encontrar informações gerais sobre os conceitos dessa área, quais as finalidades e objetivos dessa estrutura, assim como informações detalhadas do método PDCA, o qual nesse trabalho será redefinido como um método que irá abranger as técnicas de análise aqui apresentadas.

No subcapítulo de Sistemas (2.3) é possível encontrar informações sobre os conceitos de Teoria Sistêmica, a qual será utilizada como ferramenta de análise para os estudos efetuados. Também é possível encontrar conceitos de aprendizagem no subcapitulo (2.3.3), que têm como finalidade esclarecer o real objetivo das análises.

O capítulo de número (3) é constituído do desenvolvimento do estudo proposto por esse trabalho, assim como análises e exemplos de técnicas empregadas e descritas na fundamentação teórica. Assim como uma conclusão

sobre as análises efetuadas no mesmo capítulo, também serão apresentados os resultados obtidos no refinamento da modelagem utilizada.

No capítulo de número (4), encontra-se o objetivo desse trabalho, uma descrição de modelo compatível para se aplicar as técnicas apresentadas no capitulo (2), assim como conceitos de aprendizagem que devem ser utilizados para o melhor aproveitamento das análises efetuadas.

Para encerrar, são apresentadas as Referências bibliográficas utilizadas, fundamentais para o desenvolvimento.

2. FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo será apresentado um breve histórico sobre gerências de projetos, mais detalhadamente a modelagem SCRUM. Também serão apresentados conceitos sobre abordagem sistêmica e aprendizado, pois respectivamente correspondem às ferramentas e resultados propostos por esse trabalho.

2.1. ENGENHARIA DE SOFTWARE

Desenvolver um software é naturalmente uma atividade complexa. Atualmente, um desafio que cresce é aproximar a parte tecnológica da parte de negócio, além de não existir uma única solução para cada cenário de desenvolvimento. Outro ponto é que lidamos o tempo todo com pessoas, o que torna o sucesso do produto totalmente relacionado à competência da equipe e a maneira como trabalham. Para dificultar ainda mais essa construção, muitas vezes não fazemos uso de um processo bem definido, e de certo modo específico naquele ambiente, para apoiar as atividades do projeto.

Dentro da construção de um software, processo são um conjunto de passos bem definidos com os responsáveis da execução, a escolha de ferramentas de apoio e os artefatos produzidos. Em outras palavras, podemos definir o processo, como o jeito e maneira que a equipe deverá trabalhar para alcançar o objetivo: desenvolver o software com qualidade, dentro de prazos, custos e requisitos definidos.

Temos a engenharia de software como sendo, de acordo com o *IEEE*, a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, operação e manutenção de software. Sistemática, pois, parte do princípio de que existe um processo de desenvolvimento definindo as atividades que deverão ser executadas. Disciplinada por que parte do princípio de que os processos definidos serão seguidos. Quantificável por que se deve definir um conjunto de medidas a serem extraídas do processo durante o desenvolvimento de forma que as tomadas de decisão relacionadas ao desenvolvimento do software (por

exemplo, melhoria de processo) sejam embasadas em dados reais, e não em ideias e pensamentos empíricos que muitas vezes acabam levando a tomadas de decisões erradas. Alguns de seus principais objetivos são:

- Qualidade de software;
    - Requisitos funcionais e de desempenho explicitamente estabelecidos;
    - Padrões de desenvolvimento documentados;
- Processo
    - Produtividade no desenvolvimento, operação e manutenção de software;
- Permitir que profissionais pudessem ter controle sobre o desenvolvimento de software dentro de custos, prazos e níveis de qualidade desejados.

É importante destacar que a engenharia de software trata o software como um produto. Assim sendo, devemos retirar de nosso escopo programas feitos por diversão, brincadeiras ou um passatempo de programador. Estão fora do escopo, também, programas pequenos e descartáveis criados com o objetivo de resolver um pequeno problema de alguém e que futuramente não terão valor algum agregado. Mas se algum programa assim interessar a outras pessoas, e assim passar a ter valor, aparecerá demandas para melhorar sua qualidade, aumentar suas funções e prolongar sua vida. E se aparecer alguém disposto a investir nele para mais tarde ganhar dinheiro, nesse momento, o programa entrou no escopo da Engenharia de Software e se tornou um produto. Muitos softwares famosos e bastante utilizados percorreram esse caminho.

Outro ponto importante para citar é que todo software tem usuários. Alguns produtos são feitos por encomenda de um cliente que pagará por sua produção. Outros, chamados de produtos de prateleira, são vendidos no mercado aberto, a quem se interessar. Neste caso, quem faz o papel do cliente é quem define que recursos e funções se esperam do produto; talvez um departamento de vendas, ou de marketing, ou de definição de produtos de uma organização, ou até, para produtos menores, os próprios desenvolvedores, entrando no papel de investidores de risco. E existem todos os casos intermediários, em que um produto parcialmente

pronto, e que já vem sendo utilizado por alguém, é completado, adaptado ou incrementado, por encomenda de um cliente.

Como todo produto industrial, o produto de software tem seu ciclo de vida, que designa todas as etapas do desenvolvimento. O objetivo de tal segmentação é definir balizas intermediárias que permitam a validação do desenvolvimento do software, isto é, a conformidade do software com as necessidades manifestadas, e a verificação do processo de desenvolvimento e a adequação dos métodos aplicados.

Geralmente o ciclo de vida de um software atende as seguintes atividades:

- Definição dos objetivos, consistindo em definir a finalidade do projeto.
- Análise das necessidades e viabilidade, quer dizer a expressão, recolha e a formalização das necessidades do requerente (o cliente) e do conjunto dos constrangimentos.
- Concepção geral. Trata-se da elaboração das especificações da arquitetura geral do software.
- Concepção detalhada, que consiste em definir precisamente cada subconjunto do software.
- Desenvolvimento, quer dizer a tradução numa linguagem de programação das funcionalidades definidas nas fases de concepção.
- Testes unitários, que permitem verificar individualmente que cada subconjunto do "software" é aplicado em conformidade com as especificações.
- Integração, cujo objetivo é assegurar a intercomunicação dos diferentes elementos (módulos) do software.
- Validação, isto é, a verificação da conformidade do software às especificações iniciais.
- Documentação, destinada a produzir as informações necessárias para a utilização do software e para desenvolvimentos de manutenção.
- Produção.
- Manutenção, compreendendo todas as ações corretivas e evolutivas no software.

Normalmente, o software é desenvolvido dentro de projetos determinados. Todo projeto tem uma data de início e uma data de fim, uma equipe e objetivos. O responsável por um projeto é chamado de gerente de projeto. O trabalho realizado dentro de um projeto pode ser descrito por um conjunto de atividades, que podem possuir relações de dependência, paralelismo, e decomposição em atividades mais elementares.

Para Maximiniano (1997), é de responsabilidade do gerente de projetos assegurarem a realização do projeto dentro dos padrões de desempenho da missão, do prazo e do custo, o que exige a administração das comunicações, recursos humanos, contratos, materiais e riscos. Segundo o mesmo autor, o gestor do projeto deverá:

- Esclarecer de forma objetiva as necessidades do cliente, os produtos do projeto, suas especificações de desempenho e outros objetivos.
- Estabelecer estratégias eficazes para a realização dos objetivos.
- Analisar o contexto em que o projeto será iniciado e realizado.
- Entender o projeto como sistema.
- Ser capaz de coordenar e participar da elaboração de propostas, orçamentos, cronogramas e outras ferramentas de planejamento.
- Ser comunicador e integrador, garantindo que a equipe atenda as exigências relacionadas ao projeto.
- Planejar todas as tarefas necessárias para atender às exigências do cliente e da organização.
- Realizar um controle realista do projeto.

As atividades geralmente são delimitadas por marcos, isto é, pontos que representam estados significativos do projeto. Esses pontos são associados a resultados concretos: documentos, modelos ou módulos do produto, que podem fazer parte do conjunto prometido aos clientes, ou ter apenas utilização interna ao

projeto. O produto em si é um resultado concreto associado ao marco de conclusão do projeto, que pode ser utilizado sozinho, ou como componente de um sistema.

Resumidamente, a engenharia de Software visa à criação de produtos de software que atendam às necessidades de pessoas, instituições, outros softwares ou hardwares e, portanto, tenham valor econômico. Para isso, usa conhecimentos científicos, técnicos e gerenciais, tanto teóricos quanto empíricos. Quanto é praticado por profissionais treinados e bem informados ela atinge seus objetivos de produzir software com alta qualidade e produtividade, utilizando tecnologias adequadas dentro de processos que tirem proveito tanto da criatividade quando da racionalização do trabalho.

2.1.1. Modelagens de Desenvolvimento

"Um modelo de processo é uma abstração de um processo", (SOMMERVILLE, 2003). Esses modelos representam as abordagens utilizadas no desenvolvimento de software dentro das organizações. Com base nesses modelos, diversos processos foram propostos para o desenvolvimento de software, com a finalidade de se construir um produto melhor, de menor custo e mais rapidamente.

Um processo de desenvolvimento de software consiste num conjunto de atividades e resultados associados que geram um software Sommerville (2003). Basicamente, os processos de desenvolvimento têm como foco os aspectos técnicos, como especificação, desenvolvimento, validação e evolução do software e devem ter transparência e flexibilidade para facilitar o gerenciamento do projeto.

De acordo com Sommerville (2003), existe a necessidade de se gerenciar o processo de desenvolvimento de software através de modelos, processos, atividades e ferramentas específicas. O desenvolvimento de um software ganha sentido no contexto de um negócio e de uma organização. É muito importante alinhar os requisitos de negócio com o produto de software, alinhando com a tecnologia e o ambiente em que tal produto será inserido, e gerenciar as atividades de desenvolvimento, verificando prazo, custo e qualidade para que o projeto não termine em fracasso do ponto de vista do negócio.

Na modelagem de um software, podemos delimitar o problema que estamos estudando, quebrando em vários problemas menores, assim restringindo a atenção a um único ponto por vez até chegar à solução. Mesmo que não se utilize uma modelagem formal para desenvolver um software, sempre é feito algum tipo de modelo, mesmo que de maneira muito informal. Porém, esses modelos informais não oferecem uma linguagem que pode ser compreendida por outras pessoas facilmente nem geram documentação, assim não tendo qualidade.

Mas deve-se escolher um tipo de modelagem que seja correto para o sistema. Quando os modelos são adequados para o software que está sendo desenvolvido, os problemas são resolvidos mais claramente, mas quando se escolhe um modelo errado, ao invés de ajudar, ele pode complicar ainda mais o problema, causando confusões e desviando a atenção para detalhes que não são importantes para aquela situação. É nesse ponto que se torna interessante mostrar o que pode ser ajustado, e em que quantidade e ponto, ou o que pode ser tomado como ponto de parada para aceitar que tal planejamento não foi o mais adequado. Ainda assim, possível adotar uma combinação de processos complementares de acordo com as necessidades do projeto e da organização.

Em qualquer situação, os melhores modelos são aqueles que permitem escolher o grau de detalhamento. Dependendo do sistema, um modelo que mostra a interação com o usuário, de execução rápida e simples, pode ser o ideal, mas, em outros casos, pode ser necessário retornar a níveis mais baixos, como ao especificar interfaces para várias plataformas ou quando o sistema se depara com congestionamentos em uma rede.

Algo muito importante na hora de se criar um software, é saber estimar o quanto ele irá custar, seja tempo, preço ou alocação de recursos. A estimativa de software é importante para o gerenciamento de um projeto. Decisões como o cancelamento de um projeto podem ser tomadas com base em estimativas de custo e prazo necessários para desenvolver um determinado sistema de software. Estimativas adequadas resultam em decisões gerenciais acertadas, enquanto estimativas irreais causam prejuízo.

Além disso, dados do esforço real comparado com o esforço estimado fornecem uma importante ferramenta para ajuste das estimativas do projeto corrente e nos projetos futuros. A experiência e os dados agregados através desta comparação permitem estimar as atividades futuras com maior acurácia e atuar

mitigando riscos de descumprimento de prazos acordados. Para isso, o gerente deve possuir os registros organizados das atividades concluídas.

2.1.2. Modelos Incrementais

Os modelos incrementais englobam os modelos de desenvolvimento cuja constituição é de pequenos ciclos de desenvolvimento realizados de forma iterativa, ou seja, a cada ciclo novos incrementos são adicionados no software (PRESSMAN, 2006), que ganha funcionalidades no decorrer do projeto. Em cada incremento é realizado todo o ciclo do desenvolvimento de software, do planejamento aos testes do sistema já em funcionamento. Cada etapa produz um sistema totalmente funcional, apesar de ainda não cobrir todos os requisitos.

O Modelo Incremental apresenta diversas vantagens para o desenvolvimento de um software, especialmente se os requisitos não estão claros inicialmente. Outra vantagem para o desenvolvedor é que, em contato com o sistema, o cliente esclarece seus requisitos e suas prioridades para os próximos incrementos, além de contar com os serviços da versão já produzida.

De acordo com Campos (2009), Esse modelo possui vantagens em relação a um modelo sequencial de desenvolvimento. Os incrementos podem ser planejados para gerir os riscos técnicos, uma boa prática adotada por processos modernos. Ele também absorve melhor as mudanças nos requisitos, principalmente quando alguns deles ainda não foram claramente entendidos.

Outras vantagens são:

- A construção de um sistema menor é sempre menos arriscada que a construção de um grande.
- Se um grande erro é cometido, apenas o último incremento é descartado.
- Reduzindo o tempo de desenvolvimento de um sistema, as chances de mudanças nos requisitos do usuário durante o desenvolvimento são menores.
- Melhor gerenciamento de riscos, porque você pode confirmar o resultado com o cliente depois de cada versão do sistema e sempre verificar

se estão fazendo o que está de acordo com o plano ou não, e corrigi-los na próxima versão do software.

- Os testes são simples.

Dependendo das funcionalidades a serem desenvolvidas em cada incremento, atividades de gerenciamento devem realizar o respectivo planejamento, estimação e negociação dos requisitos para cada iteração.

Entretanto, o modelo incremental possui alguns problemas. Algumas funcionalidades dependem de outras, podendo ser interdependentes, por isso pode haver bloqueios no desenvolvimento. Cada fase de uma iteração é rígida e não se sobrepõem uns aos outros. Além disso, alterações em requisitos já desenvolvidos invalidam o cronograma e as estimativas.

2.1.3. Modelos Evolucionários

Nos modelos evolucionários o software é ajustado, melhorado e agrega novas funcionalidades, tornando-se mais completo, a cada ciclo de desenvolvimento (PRESSMAN, 2006). Sistemas de software precisam se adaptar com o passar do tempo. Não há como forçar um desenvolvimento linear até o produto final. A evolução gradual do produto é uma abordagem para solucionar esse problema.

Estes modelos também são iterativos e apresentam características que possibilitem desenvolvermos versões cada vez mais completas do software, porém diferem dos modelos incrementais porque acomodam melhor situações onde apenas os requisitos básicos são entendidos. Os processos evolucionários se caracterizam por dois modelos comuns: prototipação e espiral.

A prototipação é utilizada quando o desenvolvedor não tem certeza sobre a eficiência de um algoritmo, ou se um sistema operacional se adaptaria ou ainda quanto o jeito que deve ocorrer à interação entre o cliente e o sistema. Quando isso ocorre a prototipação é uma excelente alternativa. Devemos lembrar que a prototipação pode ser utilizada em qualquer processo de software, pois auxilia os interessados a compreender melhor o que está para ser construído.

A prototipação se dá basicamente com a comunicação que ocorre através de uma reunião com todos os envolvidos a fim de definir objetivos gerais do software e identificar quais requisitos já estão bem conhecidos, além de esquematizar as áreas que realmente necessitam de uma definição mais genérica. Uma iteração de prototipação deve ser planejada rapidamente e dessa forma ocorre à modelagem na forma de um projeto rápido que foca na representação dos aspectos do software que serão visíveis aos usuários como layout da interface e os formatos de exibição. Isso leva à construção de um protótipo que será avaliado pelo cliente. O cliente por sua vez retornará com comentários e avaliações para a equipe de software que irá aprimorar os requisitos. A iteração vai ocorrendo conforme vamos ajustando o protótipo às necessidades dos usuários.

Já o famoso modelo espiral foi proposto por Boehm. Esse é um modelo de processo de software evolucionário que também é iterativo como a prototipação, porém com aspectos sistemáticos e controlados do modelo cascata. Esse modelo fornece um grande potencial para que possamos ter rápido desenvolvimento de versão cada vez mais completas em seus ciclos. Sempre iniciamos pelo centro da espiral. Os riscos são considerados à medida que cada evolução é realizada. A primeira atividade se dá com o desenvolvimento de uma especificação de produto, as próximas passagens podem ser usadas para desenvolver um protótipo e, assim sucessivamente vamos evoluindo para versões cada vez mais sofisticadas do software. Cada passagem pela parte de planejamento, por exemplo, resulta em ajustes no planejamento do projeto. O custo e o cronograma são sempre ajustados de acordo com o *feedback* obtido do cliente após uma entrega. Também teremos um ajuste no número de iterações planejadas para completar o software.

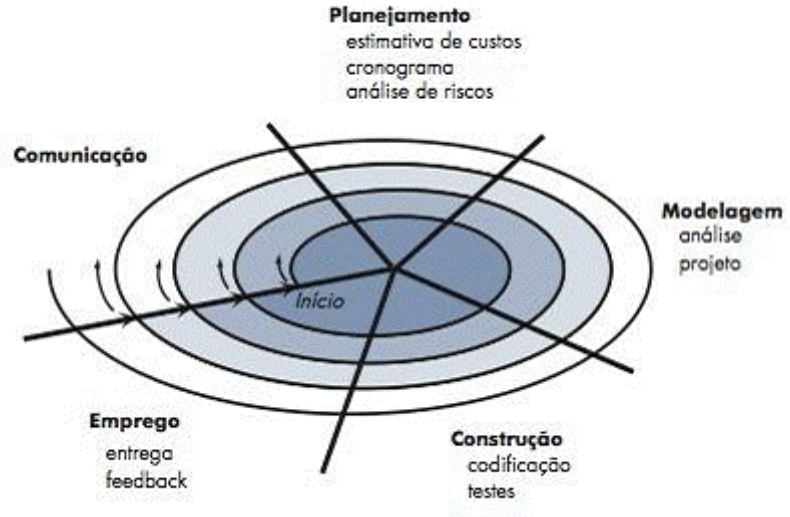


FIGURA 1: MODELO ESPIRAL
FONTE: DEVMEDIA (2015)

Entretanto, nos modelos evolucionários é difícil estimar e planejar a quantidade de iterações necessárias para construir o produto completo, pois a maioria das técnicas de gestão e estimativa de projeto é baseada na disposição linear das atividades (PRESSMAN, 2006).

A estimação e o planejamento em projetos que adotam o modelo evolucionário devem ser constantemente revisados na medida em que mudanças nos requisitos são detectadas. Na medida em que o projeto evolui, se houver gerenciamento adequado, os ajustes nas estimativas e o replanejamento provavelmente irão convergir para resultados mais próximos da realidade.

2.1.4. Modelos Ágeis

A partir da década de 1990 o desenvolvimento de software começou a evoluir como parte de uma reação contra os métodos de pesada regulamentação, forte regimento, lento e contraditório. Em 2001 o termo “Metodologias Ágeis” tornou-se mais popular quando um grupo de dezessete especialistas em processos de desenvolvimento de software decidiu se reunir para discutir maneiras de melhorar o desempenho de seus projetos.

Foi então criado o Manifesto Ágil, contendo os conceitos e princípios comuns compartilhados. Este documento enfatiza alguns princípios já conhecidos com o

objetivo de superar os desafios modernos do desenvolvimento de software (PRESSMAN, 2006).

Segundo Abrahamsson (2002), uma metodologia pode ser dita ágil quando efetua o desenvolvimento de software de forma incremental (liberação de pequenas versões em ciclos), colaborativa (cliente e desenvolvedores trabalhando juntos, em constante comunicação), direta (o método em si é simples de aprender e modificar) e adaptativa (capaz de responder às mudanças). Nesse conceito, inclui como metodologias ágeis: *Extreme Programming* (*XP*), Scrum, Crystal, *Feature Driven Development* (*FDD*) e, com certa ressalva, o *Rational Unified Process* (*RUP*).

Os princípios fundamentais dos modelos ágeis são:

- Indivíduos e interações em vez de processos e ferramentas;
- Software funcionando em vez de documentação abrangente;
- Colaboração com o cliente ao invés de negociação de contratos;
- Adaptação a mudanças em vez de seguir um plano.

Os modelos ágeis procuram estabelecer apenas um conjunto mínimo de conceitos de organização e disciplina, deixando as demais decisões a cargo da equipe. Há uma suposição de que uma equipe com experiência e diversidade de conhecimentos saberia como coordenar o seu trabalho e se auto-organizar, portanto qualquer tipo de burocracia inibiria a plena capacidade dos indivíduos. Os “agilistas” também defendem a tese que não basta apenas incluir alguns ajustes e boas práticas nos modelos tradicionais, é preciso livrar-se da “roupagem velha”.

Embora não seja a solução para todos os problemas, a metodologia ágil mostra uma maneira de trabalhar bastante organizada e iterativa, podendo inclusive contribuir para um ambiente de trabalho mais amigável, portanto é uma boa opção obter os diferenciais desejados. O objetivo dos modelos ágeis em geral não é solucionar definitivamente os desafios da Engenharia de Software, mas prover o ambiente mais adequado para o desenvolvimento de software.

2.1.5. Desenvolvimento Ágil

Os Métodos Ágeis de Desenvolvimento de Software surgiram como uma reação aos métodos clássicos de desenvolvimento e do reconhecimento da necessidade premente de se criar uma alternativa a estes “processos pesados”, caracterizados pelo foco excessivo na criação de uma documentação completa (BECK, et al, 2001). Em meados dos anos 90, integrantes da comunidade de desenvolvimento de software começaram a questionar estes processos, julgando-os pouco efetivos e, muitas vezes, impossíveis de serem colocados em prática (HIGHSMITH, 2002).

Resumindo o pensamento desse grupo, Highsmith menciona que a indústria e a tecnologia sofrem modificações tão aceleradas que acabam por “atropelar” os métodos clássicos. Highsmith (2002) ainda acrescenta que os clientes, na maioria das vezes, são incapazes de definir de forma clara e precisa os requisitos do software, logo no início de um projeto de desenvolvimento, o que inviabiliza a adoção dos métodos clássicos em muitos projetos.

Como resposta a esta situação, muitos especialistas criaram métodos próprios para se adaptar às constantes mudanças exigidas pelo mercado e às indefinições iniciais dos projetos. O agrupamento desses métodos deu origem à família dos Métodos Ágeis de Desenvolvimento de Software. Sendo assim,

> “*[...] os Métodos Ágeis podem ser considerados uma coletânea de diferentes técnicas e métodos, que compartilham os mesmos valores e princípios básicos, alguns dos quais remontam de técnicas introduzidas em meados dos anos 70, como os desenvolvimentos e melhorias iterativos*” (COHEN, 2003).

Cockburn e Highsmith (2001) já haviam afirmado que a maioria das práticas propostas pelos Métodos Ágeis não tem nada de novo e que a diferença recai principalmente sobre o foco e os valores que os sustentam.

Segundo Cohen (2003), um dos primeiros questionamentos aos métodos clássicos de desenvolvimento de software foi feito por Schwaber, criador do Scrum.

Para entender melhor os métodos clássicos de desenvolvimento de software baseados no *SW-CMM*, Schwaber (2002) elaborou um estudo junto aos cientistas da *DuPont*, que tinha por objetivo responder a seguinte pergunta: “Por que os processos definidos e defendidos pelo *SW-CMM* não promovem entregas consistentes?”, após analisarem seus processos de desenvolvimento de software, os cientistas chegaram à conclusão que, apesar do *SW-CMM* buscar a consistência, a previsibilidade e a confiabilidade dos processos de desenvolvimento de software, muitos destes processos ainda eram, de fato, imprevisíveis e impossíveis de serem repetidos. A explicação para tal recaía na complexidade dos processos propostos pelo *SW-CMM*, na consequente dificuldade de aplicação e também na necessidade de mudanças constantes e difíceis de serem antecipadas.

Schwaber percebe que para que o desenvolvimento de software seja realmente ágil, devem-se aceitar as mudanças, ao invés de dar foco extremo à previsibilidade. Quase que simultaneamente, outros especialistas no assunto chegam à conclusão de que métodos que respondam às mudanças, tão rapidamente quanto estas venham a surgir e que incentivem a criatividade, são a única maneira de enfrentar e gerenciar os problemas do desenvolvimento de software em ambientes complexos (COCKBURN; HIGHSMITH, 2001a, SCHWABER, 2002).

Neste mesmo período, modelos de processos aplicados a outras indústrias, começam a ser analisados para servir como fonte de inspiração ao aprimoramento do processo de desenvolvimento de software (POPPENDIECK, 2001). O Modelo Toyota de Produção foi alvo de atenção especial: enquanto as unidades americanas trabalhavam a 100% de sua capacidade e mantinham grandes volumes de inventário de matérias-primas e de produtos acabados, a fábrica da Toyota mantinha o nível de estoque suficiente para um dia de operação e produzia somente o necessário para atender aos pedidos já colocados. Este modelo traduzido no princípio da *Lean Manufacturing* visava à utilização mais eficiente dos recursos e a redução de qualquer tipo de desperdício e estava totalmente alinhado à filosofia da Administração da Qualidade Total, criada pelo Dr. Edwards Deming (POPPENDIECK, 2001; FERREIRA, 2002). Deming (1990) acreditava que as pessoas desejavam fazer um bom trabalho e que os gerentes deveriam permitir que os trabalhadores do chão de fábrica tivessem autonomia para a tomada de decisões e a resolução de problemas. Além disso, estimulava o estabelecimento de uma

relação de confiança com os fornecedores e defendia uma cultura de melhoria contínua dos processos e dos produtos. Enquanto as unidades fabris japonesas geravam produtos cada vez melhores e mais baratos, as fábricas americanas não conseguiam fazer o mesmo.

Com base nesta avaliação, Poppendieck (2001) listou 10 práticas que tornavam a *Lean Manufacturing* tão bem-sucedida e que, em seu entendimento, poderiam ser adaptadas e aplicadas ao desenvolvimento de software:

1. Eliminação de gastos – eliminar ou reduzir diagramas e modelos que não agregam valor ao produto final.
2. Minimização de inventário – minimizar artefatos intermediários, como documentos de requisitos e de desenho.
3. Maximização do fluxo – utilizar o desenvolvimento iterativo para redução do prazo de entrega do software.
4. Atendimento à demanda – atender às mudanças de requisitos.
5. Autonomia aos trabalhadores – compartilhar a documentação e dizer aos programadores “o que” precisa ser feito e não “como” deve ser feito.
6. Atendimento aos requisitos dos clientes – trabalhar perto dos clientes, permitindo que eles mudem suas opiniões ou seus desejos.
7. Fazer certo da primeira vez – testar o quanto antes e refazer o código se necessário.
8. Abolição da otimização local – gerenciar o escopo de forma flexível.
9. Desenvolvimento de parceria com os fornecedores – evitar relações conflitantes, tendo em mente que todos devem trabalhar juntos para gerar o melhor software.
10. Cultura de melhoria contínua – permitir que o processo seja melhorado, que se aprenda com os erros e se alcance o sucesso.

Highsmith (2002) afirma que de forma independente, Kent Beck e Ron Jeffreis percebem a importância dos princípios defendidos por Poppendieck (2001) durante um projeto de desenvolvimento de software na *Chrysler* e criam o projeto *Extreme Programming* (*XP*), um dos Métodos Ágeis de maior expressão atualmente. Simultaneamente, outras histórias começam a ecoar pelo mundo, como a vivenciadas por Alistair Cockburn, que entrevistando profissionais do *IBM Consulting Group*, percebe que equipes de projetos bem-sucedidos se desculpavam por não ter

seguido os processos formais, por não utilizar as ferramentas de alta tecnologia e por ter “simplesmente” trabalhado de forma próxima e integrado, enquanto membros de projetos mal sucedidos afirmavam ter seguido as regras e processos e que não entendiam o que havia dado errado. Com base nesta experiência, Cockburn desenvolveu o *Crystal Method*, outro Método Ágil (HIGHSMITH, 2002).

Assim sendo, percebe-se que o mundo do desenvolvimento de software passa por uma importante transformação: os métodos clássicos são vistos como não adequados a todas as situações e os especialistas reconhecem a necessidade de criação de novas práticas, orientadas a pessoas e flexíveis o suficiente para fazer frente a um ambiente de negócio dinâmico (COCKBURN; HIGHSMITH, 2001a). Os principais desafios enfrentados e que devem ser endereçados pelos novos métodos de desenvolvimento de software são assim sumarizados por Cockburn e Highsmith:

- A satisfação dos clientes passar a ter precedência frente à conformidade aos planos.
- As mudanças sempre ocorrem – o foco deixa de ser como evitá-las e passa a ser como abraçá-las e como minimizar o seu custo ao longo do processo de desenvolvimento.
- A eliminação das mudanças pode significar menosprezar condições importantes do negócio, ou seja, pode levar ao insucesso de uma organização.
- O mercado espera um software inovador, com alta qualidade, que atenda aos requisitos do negócio e que esteja disponível em prazos cada vez menores.

2.1.6. Manifesto Ágil

No início do ano 2001, criadores, especialistas e representantes dos chamados Métodos Ágeis de Desenvolvimento de Software – *Extreme Programming*, Scrum, *Dynamic Systems Development Method*, *Adaptative Software Development*, *Crystal Methods*, *Feature-Driven Development*, *Lean Development*, entre outros – se reuniram para discutir alternativas aos tradicionais processos “pesados” de desenvolvimento de software (BECK, 2001).

Eles já vinham praticando, publicando e divulgando metodologias rotuladas como “leves” há algum tempo. Mas foram enfáticos em dizer que não eram contra métodos, processos ou metodologias, sendo que muitos até mencionaram o desejo de resgatar o verdadeiro significado e a credibilidade destas palavras. Defendiam a modelagem e a documentação, mas não em excesso. Planejavam, mas reconheciam os limites do planejamento e da previsibilidade num ambiente turbulento (BECK, 2001).

A essência deste movimento é a definição de novo enfoque de desenvolvimento de software, calcado na agilidade, na flexibilidade, nas habilidades de comunicação e na capacidade de oferecer novos produtos e serviços de valor ao mercado, em curtos períodos de tempo (HIGHSMITH, 2004). Como agilidade deve-se entender “a habilidade de criar e responder a mudanças, buscando a obtenção de lucro, em um ambiente de negócio turbulento” (HIGHSMITH, 2004); ou ainda, “a capacidade de balancear flexibilidade e estabilidade”. A agilidade não deve ser vista como falta de estrutura, mas está diretamente relacionada à capacidade de adaptação a situações diversas e inesperadas. Highsmith (2004) enfatiza que a ausência de estrutura ou de estabilidade pode levar ao caos, mas que estrutura em demasia gera rigidez.

Como resultado do encontro, foi criada a *Agile Alliance*, sendo publicado o Manifesto para Desenvolvimento Ágil de Software ou o Manifesto for *Agile Software Development* (BECK, 2001), cujo conteúdo é apresentado abaixo:

> *“Nós estamos descobrindo melhores maneiras para desenvolver software, praticando e auxiliando os outros a fazê-lo. Através deste trabalho nós valorizamos: Os indivíduos e suas interações acima de processos e ferramentas; Software em produção acima da documentação exaustiva; Colaboração do cliente acima da negociação de contratos; Respostas às mudanças acima da execução de um plano. Ou seja, embora haja valor nos itens à direita, nós valorizamos mais os itens à esquerda.”* (Manifesto Ágil).

Segundo Cohen (2003), este Manifesto tornou-se a peça-chave do movimento pelo desenvolvimento ágil de software, uma vez que reúne os principais valores dos Métodos Ágeis, que os distingue dos métodos clássicos de desenvolvimento.

Além do Manifesto, foram definidos os princípios que regem a maioria das práticas dos chamados Métodos Ágeis de Desenvolvimento de Software (AGILE ALLIANCE, 2005). Estes princípios são apresentados abaixo, de acordo com a ordem originalmente proposta.

Princípios:

- Nossa maior prioridade é satisfazer o cliente através da entrega rápida e contínua de um software de valor.
- Pessoas de negócio e programadores devem trabalhar juntos, diariamente, ao longo de todo o projeto.
- Aceite as mudanças de requisitos, mesmo que numa etapa avançada do desenvolvimento.
- Entregue novas versões do software frequentemente.
- O software em funcionamento é a medida primária de progresso do projeto.
- Construa projetos com pessoas motivadas. Ofereça a elas o ambiente e todo o apoio necessários e acredite em sua capacidade de realização do trabalho.
- As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.
- O método mais eficiente e efetivo de distribuir a informação para e entre uma equipe de desenvolvimento é a comunicação face a face.
- Processos ágeis promovem desenvolvimento sustentado.
- A atenção contínua na excelência técnica e num bom projeto aprimora a agilidade.
- Simplicidade é essencial.

- Equipes de projeto avaliam seu desempenho em intervalos regulares e ajustam seu comportamento de acordo com os resultados.

2.1.7. Scrum

A metodologia Scrum surgiu de um estudo em 1986 realizado por Takeuchi e Nonaka publicado em *Harvard Business Review*. Em 1997, Jeff Sutherland iniciou discussões com Ken Schwaber, CEO da *Advanced Development Methods*, sobre esta metodologia. Ken concordou que o Scrum era o melhor caminho para desenvolver softwares e então formalizaram este processo em um congresso em OOPSLA'95. (SCHWABER, 1997).

O Scrum orienta-se por três princípios: a visibilidade, a inspeção e a adaptabilidade (SCHWABER, 2004). As coisas devem estar visíveis a todos os envolvidos no desenvolvimento, a inspeção deve ser uma ação corrente e, consequentemente, as ações para adaptação do produto de software devem ser realizadas.

Vale salientar que essa abordagem é semelhante ao ciclo PDCA (Plan, Do, Check, Action), que iremos abordar mais a frente, evidenciando que esse método fundamenta-se em valores já conhecidos e considerados válidos pela comunidade. São consideradas iniciativas essenciais do Scrum a comunicação, o trabalho em equipe, a flexibilidade, as entregas de produtos de software funcionando, sendo que essas entregas se caracterizam por serem entregas que, a cada versão, possuem novas funcionalidades ou alguma melhoria que tenha sido introduzida pela equipe (PRESSMAN, 2006).

A equipe do Scrum é composta por:

- *Team*: Consiste na equipe de desenvolvimento do projeto constituída por até dez pessoas em que cada membro tem uma habilidade em determinada área, mas os membros não são impedidos de executar tarefas em outra área. Assim, além de estarem

integrados, os membros do time conhece o software, o que diminui o impacto da saída de algum membro;

- *Product Owner*: Responsável por especificar a funcionalidade do software e tirar as dúvidas que surgirem durante o desenvolvimento. Ele é o representante do cliente para acompanhar o andamento do projeto de perto e auxiliar na construção de um software que atenda melhor as necessidades do cliente;
- *Scrum Master*: Responsável por conduzir o time e eliminar impedimentos que surgem no decorrer do processo. Impedimento é algo que pode atrapalhar um membro do time na realização de seu trabalho. Por exemplo, solicitações referentes a outras atividades que não digam respeito ao projeto, problemas no servidor de teste, dificuldades com a tecnologia e requisitos não planejados que podem atrapalhar a execução da Sprint.

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de *Sprints*. O *Sprint* representa um *Time Box* dentro do qual um conjunto de atividades deve ser executado. Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações, que são chamadas de *Sprints*.

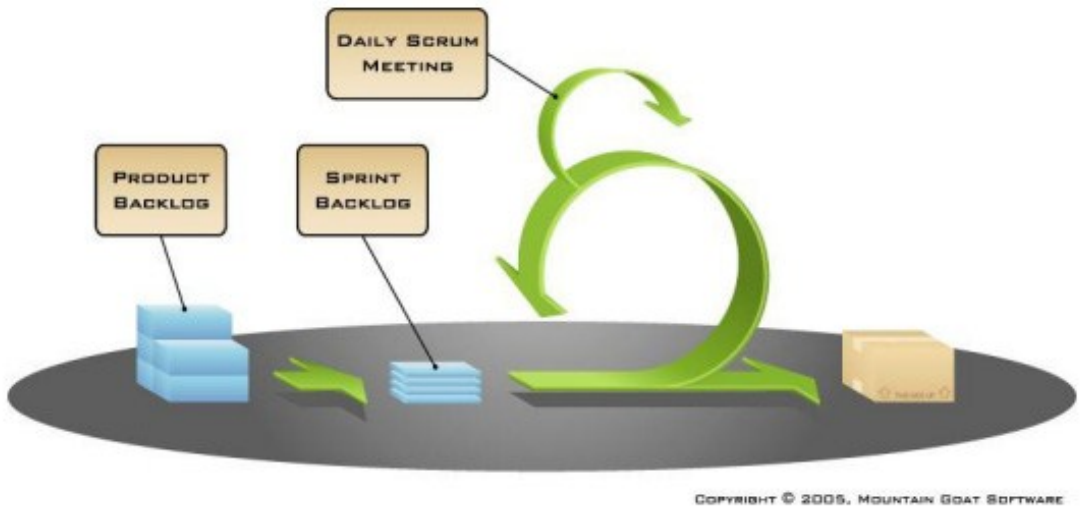


FIGURA 2: SCRUM
FONTE: MOUNTAIN (2011)

As funcionalidades a serem implementadas em um projeto são mantidas em uma lista que é conhecida como *Product Backlog*. No início de cada *Sprint*, faz-se um *Sprint Planning Meeting*, ou seja, uma reunião de planejamento na qual o *Product Owner* prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante o *Sprint* que se inicia. As tarefas alocadas em um Sprint são transferidas do *Product Backlog* para o *Sprint Backlog*.

A cada dia de uma *Sprint*, a equipe faz uma breve reunião (normalmente de manhã), chamada *Daily* Scrum. O objetivo é disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia.

Ao final de um *Sprint*, a equipe apresenta as funcionalidades implementadas em uma *Sprint Review Meeting*. Finalmente, faz-se uma *Sprint Retrospective* e a equipe parte para o planejamento do próximo *Sprint*. Assim reinicia-se o ciclo.

2.1.7.1. Daily Meeting

A cada dia do *Sprint* a equipe faz uma reunião diária, chamada *Daily Meeting*. Ela tem como objetivo disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho a ser realizado no dia que se inicia.

Os *Daily Meetings* normalmente são realizados no mesmo lugar, na mesma hora do dia e não devem durar mais que 15 minutos. Idealmente são realizados na parte da manhã, para ajudar a estabelecer as prioridades do novo dia de trabalho.

Todos os membros da equipe devem participar do *Daily Meeting*. Outras pessoas também podem estar presentes, mas só poderão escutar. Isso torna os *Daily Meetings* uma excelente forma para uma equipe disseminar informações sobre o estado do projeto.

O *Daily Meeting* não deve ser usado como uma reunião para resolução de problemas. Questões levantadas devem ser levadas para fora da reunião e normalmente tratadas por um grupo menor de pessoas que tenham a ver diretamente com o problema ou possam contribuir para solucioná-lo. Durante o *Daily*

Meeting, cada membro da equipe provê respostas para cada uma destas três perguntas:

- O que você fez ontem?
- O que você fará hoje?
- Há algum impedimento no seu caminho?

Concentrando-se no que cada pessoa fez ontem e no que ela irá fazer hoje, a equipe ganha uma excelente compreensão sobre que trabalho foi feito e que trabalho ainda precisa ser feito. O *Daily* Scrum não é uma reunião de *status report* na qual um chefe fica coletando informações sobre quem está atrasado. Ao invés disso, é uma reunião no qual membros da equipe assumem compromissos perante os demais.

Os impedimentos identificados no *Daily Meeting* devem ser tratados pelo *Scrum Master* o mais rápido possível.

2.1.7.2. Product Backlog

O *Product Backlog* é uma lista contendo todas as funcionalidades desejadas para um produto. O conteúdo desta lista é definido pelo *Product Owner*. O *Product Backlog* não precisa estar completo no início de um projeto. Pode-se começar com tudo aquilo que é mais óbvio em um primeiro momento. Com o tempo, o *Product Backlog* cresce e muda à medida que se aprende mais sobre o produto e seus usuários.

Durante o *Sprint Planning Meeting*, o *Product Owner* prioriza os itens do *Product Backlog* e os descreve para a equipe. A equipe então determina quais itens serão capazes de completar durante a *Sprint* que está por começar. Tais itens são, então, transferidos do *Product Backlog* para o *Sprint Backlog*. Ao fazer isso, a equipe quebra cada item do *Product Backlog* em uma ou mais tarefas do *Sprint Backlog*. Isso ajuda a dividir o trabalho entre os membros da equipe. Podem fazer parte do *Product Backlog* tarefas técnicas ou atividades diretamente relacionadas às funcionalidades solicitadas.

2.1.7.3. Product Owner

O *Product Owner* é a pessoa que define os itens que compõem o *Product Backlog* e os prioriza nas *Sprint Planning Meetings*.

O time olha para o *Product Backlog* priorizado, seleciona os itens mais prioritários e se compromete a entregá-los ao final de um *Sprint* (iteração). Estes itens transformam-se no *Sprint Backlog*.

A equipe se compromete a executar um conjunto de atividades no Sprint e o *Product Owner* se compromete a não trazer novos requisitos para a equipe durante o *Sprint*. Requisitos podem mudar (e mudanças são encorajadas), mas apenas fora do *Sprint*. Uma vez que a equipe comece a trabalhar em um Sprint, ela permanece concentrada no objetivo traçado para o *Sprint* e novos requisitos não são aceitos.

2.1.7.4. Scrum Master

O Scrum *Master* procura assegurar que a equipe respeite e siga os valores e as práticas do Scrum. Ele também protege a equipe assegurando que ela não se comprometa excessivamente com relação àquilo que é capaz de realizar durante um Sprint.

O *Scrum Master* atua como facilitador do *Daily Scrum* e torna-se responsável por remover quaisquer obstáculos que sejam levantados pela equipe durante essas reuniões.

O papel de *Scrum Master* é tipicamente exercido por um gerente de projeto ou um líder técnico, mas em princípio pode ser qualquer pessoa da equipe.

2.1.7.5. Sprint backlog

O *Sprint Backlog* é uma lista de tarefas que o time se compromete a fazer em um *Sprint*. Os itens do *Sprint Backlog* são extraídos do *Product Backlog*, pela

equipe, com base nas prioridades definidas pelo *Product Owner* e a percepção da equipe sobre o tempo que será necessário para completar as várias funcionalidades.

Cabe à equipe determinar a quantidade de itens do *Product Backlog* que serão trazidos para o *Sprint Backlog*, já que é ela quem irá se comprometer a implementá-los.

Durante um *Sprint*, o *Scrum Master* mantém o *Sprint Backlog* atualizando-o para refletir que tarefas são completadas e quanto tempo à equipe acredita que será necessário para completar aquelas que ainda não estão prontas. A estimativa do trabalho que ainda resta a ser feito no *Sprint* é calculada diariamente e colocada em um gráfico, resultando em um *Sprint Burndown Chart*.

2.1.7.6. Sprint Meetings

O *Sprint Planning Meeting* é uma reunião na qual estão presentes o *Product Owner*, o *Scrum Master* e todo o time, bem como qualquer pessoa interessada que esteja representando a gerência ou o cliente.

Durante o *Sprint Planning Meeting*, o Product Owner descreve as funcionalidades de maior prioridade para a equipe. A equipe faz perguntas durante a reunião de modo que seja capaz de quebrar as funcionalidades em tarefas técnicas, após a reunião. Essas tarefas irão dar origem ao *Sprint Backlog*.

O *Product Owner* não precisa descrever todos os itens que estão no *Product Backlog*. Dependendo do tamanho do *Product Backlog* e da velocidade da equipe, pode ser suficiente descrever apenas os itens de maior prioridade, deixando a discussão dos itens de menor prioridade para o próximo *Sprint Planning Meeting*.

Coletivamente, o time e o *Product Owner* definem um objetivo para o *Sprint*, que é uma breve descrição daquilo que se tentará alcançar no *Sprint*. O sucesso do Sprint será avaliado mais adiante no *Sprint Review Meeting* em relação ao objetivo traçado para o *Sprint*.

Depois do *Sprint Planning Meeting*, a equipe Scrum se encontra separadamente para conversar sobre o que eles escutaram e decidir quanto eles podem se comprometer a fazer no *Sprint* que será iniciado. Em alguns casos, haverá

negociação com o *Product Owner*, mas será sempre responsabilidade da equipe determinar o quanto ela será capaz de se comprometer a fazer.

Ao final de cada *Sprint* é feito um *Sprint Review Meeting*. Durante esta reunião, o time mostra o que foi alcançado durante o *Sprint*. Tipicamente, isso tem o formato de um demo das novas funcionalidades.

Os participantes do *Sprint Review* tipicamente incluem o *Product Owner*, o time, o *Scrum Master*, gerência, clientes e engenheiros de outros projetos.

Durante o *Sprint Review*, o projeto é avaliado em relação aos objetivos do *Sprint*, determinados durante o *Sprint Planning Meeting*. Idealmente, a equipe completou cada um dos itens do *Product Backlog* trazidos para fazer parte do *Sprint*, mas o importante mesmo é que a equipe atinja o objetivo geral do *Sprint*.

E finalmente quando acaba é necessário fazer a última reunião e olhar para trás e repensar o que deu certo, o que deu errado e planejar o que pode ser melhorado no futuro.

Assim, o que a metodologia ágil Scrum traz é um processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software sustentado por quatro pilares:

- Os indivíduos e suas interações acima de procedimentos e ferramentas.
- O funcionamento do software acima de documentação abrangente.
- A colaboração dos clientes acima da negociação de contratos.
- A capacidade de resposta às mudanças acima de um plano pré-estabelecido.

2.2. MÉTODOS DE GESTÃO

2.2.1. PDCA

O ciclo PDCA é um método gerencial de tomada de decisões para garantir o alcance das metas necessárias à sobrevivência de uma organização. Segundo Ishikawa (1989, 1993) e Campos (1992, 1994) o ciclo PDCA (*Plan*, *Do*, *Check*, *Action*) é composto das seguintes etapas:

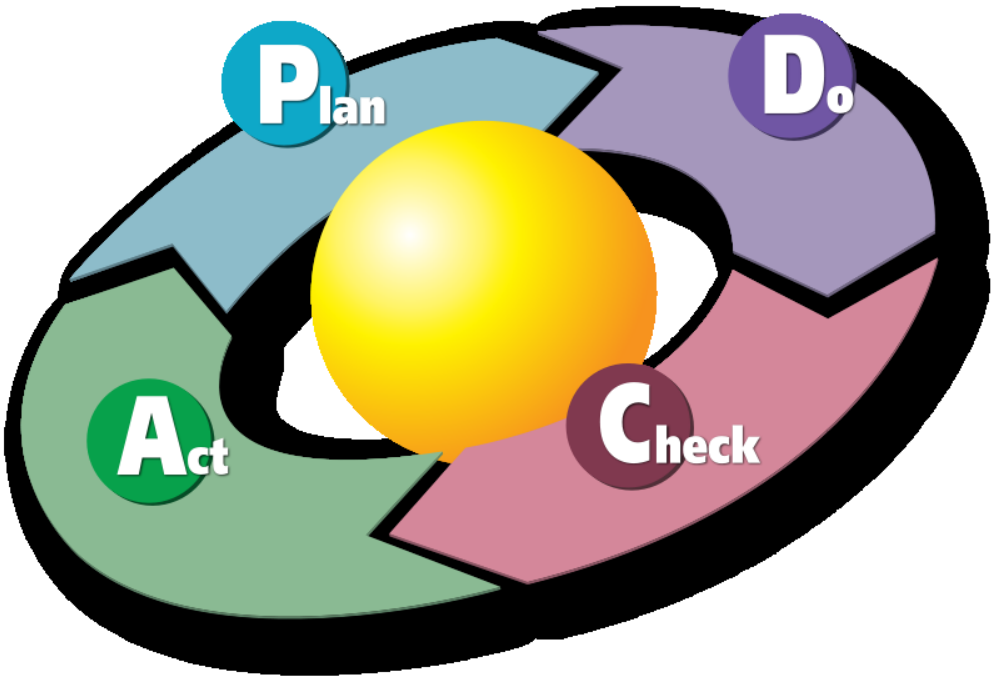


FIGURA 3: PDCA
FONTE: BULSUK (2008)

- Planejamento (P) Essa etapa consiste em estabelecer metas e estabelecer o método para alcançar as metas propostas.
- Execução (D) Executar as tarefas exatamente como foi previsto na etapa de planejamento e coletar dados que serão utilizados na próxima etapa de verificação do processo. Na etapa de execução são essenciais educação e treinamento no trabalho.
- Verificação (C) A partir dos dados coletados na execução compararem o resultado alcançado com a meta planejada.

- Atuação Corretiva (A) Etapa que consiste em atuar no processo em função dos resultados obtidos, adotando como padrão o plano proposto, caso a meta tenha sido atingida ou agindo sobre as causas do não atingimento da meta, caso o plano não tenha sido efetivo. De acordo com Campos (1994), na utilização do método poderá ser preciso empregar várias ferramentas para a coleta, o processamento e a disposição das informações necessárias à condução das etapas do PDCA. Estas ferramentas serão denominadas ferramentas da qualidade. Entre as ferramentas da qualidade, as técnicas estatísticas são de especial importância.

As Sete Ferramentas da Qualidade (Estratificação, Folha de Verificação, Gráfico de Pareto, Diagrama de Causa e Efeito, Histograma, Diagrama de Dispersão, Gráfico de Controle).

- Amostragem
- Análise de Variância
- Análise de Regressão
- Planejamento de Experimentos
- Otimização de Processos
- Análise Multivariada
- Confiabilidade

Para entender o papel das ferramentas da qualidade dentro do ciclo do PDCA, devemos novamente destacar que a meta (resultado) é alcançada por meio do método (PDCA). Quanto mais informações (fatos, dados, conhecimentos) forem agregadas ao método, maiores serão as chances de alcance da meta e maior será a necessidade da utilização de ferramentas apropriadas para coletar, processar e dispor estas informações durante o giro do PDCA. Vale destacar que o aumento da sofisticação das ferramentas empregadas deverá ocorrer em função do aumento da capacidade de alcance das metas.

Segundo Werkema (1995b), as metas para melhorar, ou metas de melhoria, surgem do fato de que o mercado (clientes) sempre deseja um produto cada vez

melhor, a um custo cada vez mais baixo e com entrega cada vez mais precisa. A entrada de novos concorrentes no mercado e o surgimento de novos materiais e novas tecnologias também levam à necessidade do estabelecimento de metas de melhoria. As metas de melhoria são metas que devem ser atingidas e para que isso seja possível será necessário modificar a forma atual de trabalho. As expressões "reduzir em 30% a variação da dureza das peças de aço fabricadas pela empresa até o final do ano" e "reduzir o prazo máximo de entrega do produto ao cliente de dois dias para um dia até o final do ano" são exemplos de metas para melhorar. Cada meta de melhoria gera um problema que deverá ser "atacado" pela empresa.

Segundo Werkema (1995b), para atingir metas de melhoria de resultados utilizamos o ciclo PDCA, que nesse caso também é denominado Método de Solução de Problemas, já que como visto anteriormente, cada meta de melhoria gera um problema que a empresa deverá solucionar.

Em relação ao ciclo PDCA de melhorias, ainda segundo Werkema (1995b) deve-se fazer as seguintes observações:

> • Planejamento (P) O problema identificado na fase um da etapa P do PDCA é gerado a partir da meta de melhoria (estabelecida sobre os fins), a qual pode pertencer a uma das duas categorias relacionadas a seguir: Meta "Boa" – É aquela que surge a partir do plano estratégico, sendo baseada nas exigências do mercado e na necessidade de sobrevivência da empresa. Meta "Ruim" – É aquela proveniente de anomalias crônicas. O trabalho que objetiva o alcance das metas ruins não agrega valor, já que apenas corrige algo que anteriormente foi mal feito. Após o estabelecimento da meta e a identificação do problema, deve ser feita uma análise do fenômeno ou análise do problema (observação), para que as características do problema possam ser reconhecidas. A análise do fenômeno, realizada sobre os fins, consiste em investigar as características específicas do problema, com uma visão ampla e sob vários pontos de vista. Esta análise permite a localização do foco do problema. A próxima fase da etapa P é a análise do processo (análise), realizada sobre os meios, que tem por objetivo a descoberta das causas fundamentais do problema. Na análise de processo devemos investigar o

relacionamento existente entre o fenômeno, concentrando nossa atenção no foco do problema identificado na fase anterior, e quaisquer deficiências que possam existir no processo (meios). Após a condução da análise do processo, deve ser estabelecido o plano de ação (sobre os meios), que é um conjunto de contramedidas com o objetivo de bloquear as causas fundamentais. Para cada contramedida constante do plano de ação, deverá ser definido o "5W2H". O QUE ("*WHAT*") será feito, QUANDO ("*WHEN*") será feito, QUEM ("*WHO*") fará, ONDE ("*WHERE*") será feito, POR QUE ("*WHY*") será feito e COMO ("*HOW*") será feito e qual será o CUSTO ("*HOW MUCH*"). A etapa de planejamento do ciclo PDCA de melhorias consiste então no estabelecimento de metas sobre os fins e na definição das ações que deverão ser executadas sobre os meios para que a meta possa será atingida. Esta é a etapa mais difícil do PDCA. No entanto quanto mais informações forem agregadas ao planejamento, maiores serão as possibilidades de que cada meta seja alcançada. Além disto, quanto maior for o volume de informações utilizadas, maior será a necessidade do emprego de ferramentas apropriadas para coletar, processar e dispor estas informações. Também é importante destacar que a quantidade de informações e o grau de sofisticação das ferramentas necessárias à etapa P variam de acordo com o tipo de atividade no qual o giro do PDCA está inserido, ou seja, dependem da complexidade do problema sob consideração.

- Execução (D) A etapa de execução do PDCA de melhorias consiste no treinamento nas tarefas estabelecidas no plano de ação, na execução destas tarefas e na coleta de dados que serão utilizados na etapa seguinte, de confirmação da efetividade da ação adotada.

- Verificação (C) Na verificação do ciclo PDCA de melhorias será feita a confirmação da efetividade da ação de bloqueio adotada. Se o bloqueio não foi efetivo e a meta de melhoria não foi atingida, devemos retornar à fase de observação, fazer uma nova análise, elaborar um novo plano de ação e emitir o chamado "Relatório de Três Gerações", que é o documento que relata o esforço de se atingir a meta por meio do giro do PDCA. O Relatório de Três Gerações

deve mostrar: O que foi planejado (passado). O que foi executado (presente). Os resultados obtidos (presente). Os pontos problemáticos, responsáveis pelo não atingimento da meta (presente). A proposição (plano) para resolver os pontos problemáticos (futuro). Caso o bloqueio tenha sido efetivo, resultando no alcance da meta, devemos passar à etapa A do PDCA de melhorias.

- Atuação Corretiva (A) A fase de padronização da etapa A consiste em adotar como padrão as ações que “deram certo”, isto é, as ações cuja implementação permitiu o alcance da meta. Observe que, para que a consolidação do alcance da meta de melhoria possa ocorrer, a nova maneira de trabalhar definida a partir do giro do PDCA de melhorias deverá ser utilizada no dia a dia, passando então a constituir o novo patamar que será adotado como padrão. Após a padronização vem à fase de conclusão, na qual deve ser feita uma revisão das atividades realizadas e o planejamento para o trabalho futuro.

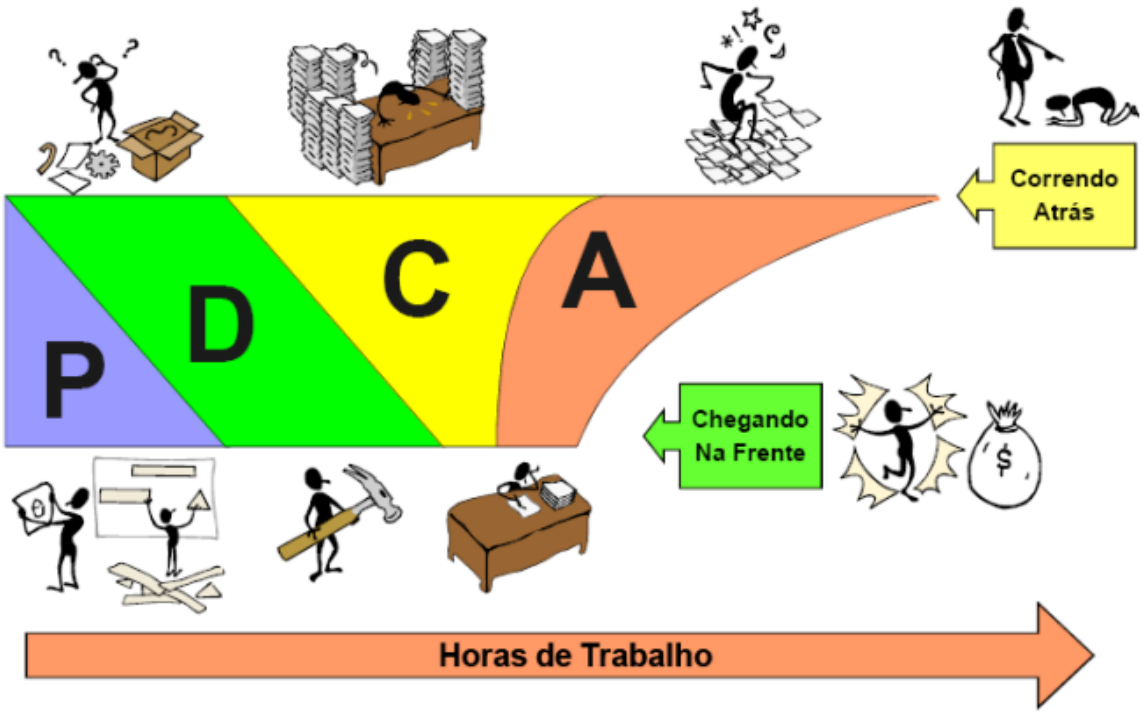


FIGURA 4: PDCA AO LONGO DO TEMPO
FONTE: CONSTRUIR (2015)

Para encerrar as observações relativas à atuação para o alcance das metas de melhoria, é importante ressaltar que existem duas maneiras pelas quais estas metas podem ser atingidas por meio do giro do ciclo PDCA.

Melhorando-se continuamente os processos existentes. Nesse tipo de atuação são feitas sucessivas modificações nos processos existentes na empresa, tais como dar mais treinamento aos operadores, empregar matérias primas de qualidade mais uniforme e aperfeiçoar a forma de utilização de equipamentos e ferramentas. Estas modificações geralmente conduzem a ganhos sucessivos obtidos sem nenhum investimento ou com pequenos investimentos.

Projetando-se um novo processo ou fazendo-se modificações substanciais nos processos existentes. O projeto de um novo processo ou a realização de grandes modificações no processo existente são ações necessárias quando as metas colocadas pelo mercado são tão desafiadoras que não podem ser atingidas pelo processo existente. Geralmente este procedimento resulta em grandes avanços para a empresa, mas também implica na realização de investimentos elevados. A implantação de um processo totalmente informatizado, visando o alcance das metas de melhoria, é um exemplo deste modo de atuação.

2.3. SISTEMAS

2.3.1. Teoria Sistêmica

De acordo com Alvarez (1990), um sistema pode ser definido como um conjunto de elementos interdependentes que interagem com objetivos comuns formando um todo, e onde cada um dos elementos componentes comporta-se, por sua vez, como um sistema cujo resultado é maior do que o resultado que as unidades poderiam ter se funcionassem independentemente. Qualquer conjunto de partes unidas entre si pode ser considerado um sistema, desde que as relações entre as partes e o comportamento do todo sejam o foco de atenção.

Visto que nosso objetivo é identificar causas problemáticas e não resolver problemas em si, teremos que realizar uma análise minuciosa sobre todos os

elementos que afetam direta ou indiretamente a ocorrência do problema, ou seja, para identificarmos um problema que ocorre em determinada parte do sistema, precisamos ter uma visão analítica sobre o sistema como um todo.

Mas para identificar as correlações do problema, precisamos identificar as correlações das variáveis, pois de acordo com Kleiner (1997), Um sistema é um todo percebido, cujos elementos mantêm-se juntos porque afetam continuamente uns aos outros ao longo do tempo, e atuam para um propósito comum.

Vale ressaltar que são partes independentes, porém que atuam juntas em um propósito comum, assim como afirma Oliveira (2002), onde descreve o sistema como um conjunto de partes interagentes e interdependentes que, conjuntamente, formam um todo unitário com determinado objetivo e efetuam determinada função.

Levando em consideração essas teorias, seremos capazes de analisar o fluxo de influência entre as variáveis do sistema, assim podemos gerar estratégias de atuação sobre uma variável específica. A interferência no sistema através de uma variável pode mudar a estrutura do sistema como um todo, porém a cadeia de influências deve ser rigorosamente analisada. Pois uma interferência positiva em uma variável pode em primeira instância acarretar em uma mudança positiva, mas ao decorrer do tempo o impacto dessa mudança pode desencadear uma sequência de outras mudanças que se não forem previamente estudadas, podem gerar uma resultado negativo sobre o sistema.

Assim como afirma Senge (1990), onde diz que existem razões pelas quais as explicações estruturais serem tão importantes, reside no fato delas serem as únicas a lidarem com as causas do comportamento num nível em que os padrões de comportamento podem ser alterados. A estrutura gera comportamento e mudando-se a estrutura, podem-se gerar diferentes padrões de comportamento.

Mas é possível prever essas mudanças de comportamento, pois segundo o mesmo autor, Senge (2000), os sistemas enviam sinais continuamente para si mesmo através de ciclos fechados circulares de relações de causa e efeito. Onde isso é chamado de “retroalimentação” (*feedback*), pois o efeito do sistema realimenta as influências em si mesmo. O resultado de estudos sobre esse efeito gerou um conjunto de ferramentas para mapear e desenhar sistemas, e o ato de se familiarizar com esses conceitos vai proporcionar a capacidade de compreender uma linguagem que torna viável discutir sobre eventos complexos.

É comum acreditar que algo que está certo não necessite atenção, como o famoso dito popular “time que está ganhando, não se mexe”, porém é uma crença totalmente errônea quando estamos tratando de sistemas, pois um dos conceitos fundamentais da teoria de sistemas é que todo sistema sofre entropia, ou seja, todo sistema está constantemente em deterioração.

A Física e a Química fazem uso da palavra entropia há algum tempo. A etimologia da palavra entropia é decorrente do radical grego *em* (dentro) e *tropee* (mudança, troca, alternativa), ou seja, mudança interior, troca interna ou alternativa de dentro.

Conforme Máximo e Alvarenga (1993), a entropia é a perda de uma parte da energia disponível que seria usada para movimentar outras máquinas. Uma vez perdida essa parte de energia, o processo se torna irreversível e a desordem se instala de tal forma que o sistema não consegue mais reverter o processo de entropia.

O processo de deterioração é inevitável, mas as praticas que aqui serão discutidas resultam em uma homeostase sistêmica, ou seja, manter a evolução constante para que haja ao menos um equilíbrio interno, pois assim como descreve Chiavenatto (1999), um sistema pode sofrer todo tipo de influências, mas a sua concepção, o seu caráter mantém-se intacto.

2.3.2. Introdução à Abordagem Sistêmica

Segundo Senge (1990), uma pessoa que desenvolve sua proficiência, desenvolve sua própria capacidade de percepção e assim gradativamente passa a se render a novos modos de olhar o mundo.

Uma pessoa fechada aos fatos ocorrentes a sua volta, restringe seu potencial de crescimento perante o sistema e ao não absorver fatos do sistema a qual pertence, acaba por restringir seu potencial de crescimento pessoal. O ciclo não possui brechas e nem exceções, quando alguma situação foge da nossa compreensão, facilmente nos deixamos iludir por alternativas que desvirtuam ao pressuposto desse ciclo. Ou seja, só porque não conseguimos enxergar o vinculo da situação com o sistema, não significa que o vinculo não exista.

É fácil atribuir uma grande ideia a “genialidade” de um individuo, pois é fato que algumas pessoas possuem um dom inato para uma ou mais disciplinas. Assim, “talento” passa ser a resposta fácil de aceitar a execução de obras as quais não compreendemos a viabilidade.

Considere uma pessoa extremante dotada de certo talento artístico, a título de exemplo vamos nomeá-lo de “João”, suas obras são únicas e extremamente valorizadas pela sociedade, tornado assim o artista que as criou um ícone na área de atuação.

Mas será que essas obras artísticas são frutos de puro talento?

Agora considere um segundo artista, a titulo de exemplo vamos nomeá-lo de “Marcos”, embora seja impossível ponderar, esse artista tem um talento natural para a arte muito superior ao talento do artista citado no primeiro exemplo, e atua no mesmo nicho de João e suas obras também são únicas, porém não tem a mesma expressividade do primeiro artista, logo seu trabalho não tem o mesmo reconhecimento da sociedade.

Qual seria a diferença entre os dois artistas?

Para responder essa pergunta, temos um terceiro individuo que compara as duas obras, cada uma pertencente a um dos artistas citados. A resposta mais intuitiva para qualquer pessoa seria que João possui mais talento do que Marcos, porém foi afirmado na instancia do problema que Marcos possui mais talento do que João.

Como chegamos a esse empasse?

A comparação foi realizada a partir das obras desses dois artistas distintos, tentando chegar a uma conclusão sobre a qualidade do produto final, levando em consideração apenas o talento dos criadores. Porém algumas variáveis desse problema não foram levadas em consideração, assim foram tiradas conclusões sem qualquer análise do sistema como um todo. Se observarmos a situação com uma visão mais ampla do sistema, somos capazes de identificar e levantar questões sobre o sistema para facilitar nossa própria compreensão, porém as questões apenas ajudam a estruturar nossos modelos mentais, mas são suas respostas que podem nos levar a conclusões mais apuradas sobre o sistema analisado.

De acordo com Wind, Crook e Gunther (2004), modelos mentais moldam nossa capacidade de identificar oportunidades e ameaças, bem como podem limitar ou melhorar a criatividade.

A primeira vista, parece que estamos complicando mais o estudo ao invés de simplificar esse processo, porém o levantamento de novas questões não é um empecilho se tivermos a capacidade de identificar rapidamente as respostas, as quais geralmente estão presentes na instância do problema, ou são tão intuitivas que praticamente já estão respondidas na própria questão. Reconhecer essas respostas não é um passo difícil, o verdadeiro trabalho se da ao transformar essas respostas em variáveis do nosso estudo, e somente a partir do momento em que enxergamos essas variáveis como métricas ponderáveis é que conseguiremos analisá-las para chegar a conclusões mais racionais.

Algumas perguntas de nosso exemplo, geradas através de uma visão sistêmica:

1- Por que as obras têm reconhecimentos distintos da sociedade?

2- Por que Mario tem menos expressividade?

3- Se ambos atuam no mesmo nicho, porque somente João é um ícone na área de atuação?

A primeira pergunta é o nosso padrão de qualidade, pois o “reconhecimento” nesse exemplo caracteriza o valor do produto. E a resposta para essa pergunta está literalmente disposta dentro da própria questão, pois quem gera o reconhecimento da sociedade se não a própria sociedade. A resposta para qualificar a qualidade dos produtos desses artistas, não estava nos artistas, mas sim nos padrões impostos pela sociedade que os julgam. Encarando esse fato como verdade, se torna possível enxergar uma resposta para a pergunta de numero dois, uma resposta que está implícita quando assumimos que a premissa da primeira questão é verdadeira. Se a falta de expressividade de Marcos acarreta na falta de reconhecimento pela sociedade, sendo que nosso padrão de qualidade é o “reconhecimento”, por fugir do padrão de qualidade, a falta de reconhecimento de Marcos perante a sociedade tornam suas obras menos expressivas. Por fim, a terceira pergunta vai nos ajudar a compreender melhor como João é mais bem sucedido do que Marcos, pois para João tornar-se um ícone nessa área de atuação seu produto teve que ser reconhecido, e para ser reconhecido João precisou desenvolvê-lo sob os padrões impostos por esse sistema, assim podemos concluir que mesmo não possuindo o

mesmo talento de Marcos, João obteve maior sucesso por entender e se adequar ao sistema que o cerca.

Com os resultados obtidos, podemos levantar conclusões não só apenas sobre o problema, mas sim conclusões sobre o sistema. Exemplificando sobre nosso caso estudado, sabemos que nesse sistema o sucesso não está diretamente relacionado ao talento, a qualidade é subjetiva e totalmente imposta pelo sistema e etc.

Essas conclusões podem auxiliar a solucionar outras situações problemáticas, distintas, geradas pelo próprio sistema. Ou seja, um sistema constantemente estudado é um sistema com capacidade evolutiva, viável através de uma base de conhecimento gerada pela abordagem sistêmica. A análise sistêmica é uma forte aliada na solução de problemas, torna o individuo mais dinâmico a encarar as situações desafiadoras que irão surgir no meio em que vive. Porém o verdadeiro aprendizado não está na solução do problema, mais sim no estudo de como essa solução foi encontrada. Quando passamos a gerar um ciclo de aprendizado através dos resultados, aprendemos a não só solucionar problemas, mas acabamos desenvolvendo a capacidade de identificar um fator critico antes mesmo que ele venha a se tornar um verdadeiro problema.

O exemplo utilizado é simples, porém extremamente didático e suficiente para entender os princípios da abordagem sistêmica. Contudo, não nos torna aptos a encarar de forma mais profunda o assunto, a abordagem sistêmica consiste em aplicar dois princípios fundamentais, que são o Pensamento Sistêmico e a Dinâmica de Sistemas, e só através deles é possível obter o esperado resultado final, que é o aprendizado. Muitas vezes trabalhamos exaustivamente em um desenvolvimento, dominando o processo aplicado, mas ao chegarmos ao final nos deparamos com uma situação curiosa: “Obtemos o resultado esperado, sabemos que está certo, mas não sabemos como compreender esse produto”. A alienação a informações básicas do produto prejudicam no que chamaremos de “valor agregado”, ou seja, o valor obtido para crescimento pessoal após a elaboração do produto. Antes de nos aprofundarmos na abordagem sistêmica, sugiro em aprimorarmos nossos conhecimentos sobre o nosso resultado, o aprendizado.

2.3.3. Aprendizado

É característica do ser humano acreditar que domina tudo aquilo que considera simples, mas existe uma diferença entre considerarmos algo simples e desconhecermos a complexidade daquilo que consideramos simples. O aprendizado não é diferente, para a grande maioria das pessoas é um tema considerado extremamente simples, pois utilizamos o conceito gradativamente ao longo da nossa vida, mas assim como a grande maioria dos conceitos, é extremamente complexo. Somos capazes de aprender devido a diversas medidas que foram expostas ao longo de nossa formação, principalmente durante nossa educação no estágio estudantil, mas muitas dessas medidas aplicadas não são explicitas ou discutidas nos tornando assim cegos ao verdadeiro valor de aprendizado que temos em nossa experiência, de forma simplificada, temos a capacidade de aprender, mas realmente temos a capacidade de ensinar outros a aprender?

Parece totalmente fora de contexto, a primeira vista, pois aparenta que objetivo aqui seria um embalsamento teórico para formar um professor ou alguém capaz de ministrar o assunto desse trabalho. Isso não constitui o objetivo desse trabalho, porém, o domínio em relação ao aprendizado está diretamente relacionado com o objetivo proposto por esse trabalho, pois para o sucesso das práticas que serão aqui abordadas, teremos de ser capazes de ensinar pelo menos uma pessoa a aprender. Nós mesmos.

A abordagem sistêmica que será abordada irá gerar um acervo de conhecimento sobre seu sistema empregado, onde o produto final será a aprendizagem. E nesse trabalho não podemos fazer mais do que auxiliar a interpretar os resultados, porém somente o analista é capaz de transformar esses resultados em valor agregado para o seu próprio desenvolvimento.

Não se engane, pois independente do seu nível de conhecimento sobre um tema, de alguma forma você ainda é um aprendiz, assim como é dito por Senge (1990) em seu caderno de campo, “Praticar uma disciplina é ser um aprendiz perpétuo que trilha um caminho de desenvolvimento sem fim”.

Vamos observar alguns conceitos da teoria de aprendizagem proposta por Vygotsky (1934), embora essas teorias sejam voltadas para um indivíduo em

desenvolvimento de aprendizado inicial, ou seja, em geral tratam de fundamentos para compreender a evolução de crianças em estágios de aprendizagem. As teorias de Lev Vygotsky são comuns em ambientes pedagógicos, mas não há restrições para as áreas de influência dessas teorias, e como estamos tratando do principio básico de aprendizagem e supondo que iremos nos confrontar a um processo de desenvolvimento de aprendizado, independente de qual seja, os conceitos abordados por essas teorias contribuíram muito com a compreensão dos nossos resultados.

Pois de acordo com Vygotsky (1934), uma pessoa em processo de aprendizado pleno, tem seu potencial estimulado quando conclui um ciclo de aprendizagem, o qual ele dividiu em quatro estágios: Interação, Mediação, Internalização e Zona de Desenvolvimento Proximal.

Primeiramente o indivíduo necessita interagir, ou seja, todo sujeito adquiri seu conhecimentos através de relações interpessoais de troca com o meio, assim tudo aquilo que parece individual, na verdade é um produto construído pelo coletivo. Essa troca de informação com o coletivo, simplificadamente é uma negociação de modelos mentais onde individuo se torna capaz de internalizar suas ideias. Mas para realizar tal negociação o individuo necessita se comunicar, onde tal ato engloba uma infinidade de recursos no qual o individuo faz uso para expor os seus pensamentos, assim qualquer forma capaz de transmitir ideias ou sentimentos pode ser interpretada como linguagem, e somente através da linguagem podemos compreender aquilo que nos foi oferecido, a linguagem se torna então o fator mediador dessa troca.

Para Seleme (2005) A linguagem é uma construção simbólica (ou de símbolos), organizada de tal forma que permite ao usuário tanto construir mentalmente uma representação da realidade quanto expressar objetivamente esta realidade.

Esse processo descreve a obtenção do conhecimento, porém o individuo necessita analisar essas novas informações obtidas para realmente transformá-las em aprendizado. Esse processo de maturar internamente as ideias é conhecido como internalização, onde o individuo passa a vincular a informação com o conhecimento previamente obtido e criar novas associações para a informação desconhecida, assim expandindo sua conectividade de informações. Por fim, esse processo não é perfeito, por isso a zona de desenvolvimento proximal é fundamental

para que o processo de aprendizado seja completo, ela consiste em unir aquilo que o individuo já possui, com aquilo que ele ainda não sabe, mas tem pleno potencial para aprender, ou seja, é quando o indivíduo consegue maturar suas ideias, porém não consegue as entender, assim necessita de auxilio para compreendê-las. Essa ajuda vem do coletivo, que é capaz de fornecer os recursos necessários para a viabilização desse processo de interpretação da ideia já elaborada, onde geralmente o recurso necessário é fornecido por um elemento mais experiente do próprio sistema.

Como dito anteriormente, as teorias de aprendizagem não são exclusividades da área pedagógica, pois enquanto o seu processo de desenvolvimento de software envolver mais de uma pessoa, irá caracterizar um sistema com relações interpessoais, assim qualquer teoria sistêmica que visa o aprendizado estará sujeita a levar em consideração as capacidades de aprendizado dos seres humanos.

2.3.4. Pensamento Sistêmico

Apesar da proximidade dos conceitos entre dinâmica de sistemas e pensamento sistêmicos, mesmo ambos as disciplinas constituírem as partes de um mesmo produto, abordagem sistêmica, os conceitos são diferentes. Porém o resultado não pode ser encarado como uma soma de processos diferentes, pois ambos são conceitos teóricos que só se transformam em um processo quando aplicadas em conjunto. Porém, a fins didáticos, iremos considerar que os conceitos aqui apresentados podem ser considerados como etapas do processo.

Diante do sistema que iremos analisar, vamos levantar suas variáveis e estudar suas ligações, pois de acordo com (BELLINGER, 1996), um sistema segundo a lógica do Pensamento Sistêmico, pode ser definido como uma entidade que mantém sua existência através da interação mútua entre suas partes.

Ou seja, um sistema é um emaranhado de elementos interligados, assim os elementos mesmo independentes estão sujeitos a influências devido as suas ligações. Cada influência tem conotações e pesos distintos para cada ponto de vista, ponderar esses valores constitui dinamizar as variáveis, tais conceitos serão

explicados em dinâmicas de sistemas, porém para ponderar valores entre essas ligações, primeiro precisamos identificar quais são essas conexões.

As conexões são toda e qualquer ligação entre conceitos do sistema, o modo mais fácil de identifica-las é levantando todos os conceitos do sistema e os confrontando um a um, nesse processo identificaremos quais são as variáveis do nosso sistema. Os conceitos que possuírem uma relação com outro conceito constituem uma conexão. Um conceito pode constituir inúmeras conexões, como também pode não constituir conexões, neste caso dizemos que a variável é irrelevante.

Uma relação é definida quando um conceito possui influência sobre outro conceito. O conceito influente é definido como variável independente e um conceito que sofre influência é caracterizado como variável dependente.

FIGURA 5: INFLUÊNCIA ENTRE CONCEITOS
FONTE: OS AUTORES (2015)

Pois de acordo White e Gunstone (1997), a influência pode ser identificada através da aplicação de um verbo, ou seja, existe influência quando existe ação entre conceitos.

Exemplo:

FIGURA 6: INFLUÊNCIA ENTRE CONCEITOS (EXEMPLO)
FONTE: OS AUTORES (2015)

De acordo com o exemplo, “Evaporação” provoca “Chuva”, assim o conceito A tem ação sobre o conceito B, o que caracteriza A como variável independente e B Como variável dependente do sistema. Porém um sistema não é constituído de conexões isoladas, logo, quando inserimos mais conceitos a nosso sistema, definiremos um fluxo de influência. É intuitivo acreditar que o fluxo possui um único

sentido, porém só representamos o sentido forte, pois também pode existir uma influência da variável dependente para a independente que deve ser representada caso tenha teor equivalente à dependência forte, caso contraio apenas a desconsideramos para titulo de análise.

Esse teor recíproco é descrito por Senge (1990), onde tais fluxos de influências tem um caráter recíproco, uma vez que toda e qualquer influência é, ao mesmo tempo, causa e efeito – a influência jamais tem um único sentido, sendo denominado de enlace ou *feedback*.

Assim o sistema necessita ser analisado como um todo, logo, quando começamos a realizar novas conexões para conceitos já ligados em um sistema, construímos algo que é conhecido como modelo conceitual.

Exemplo:

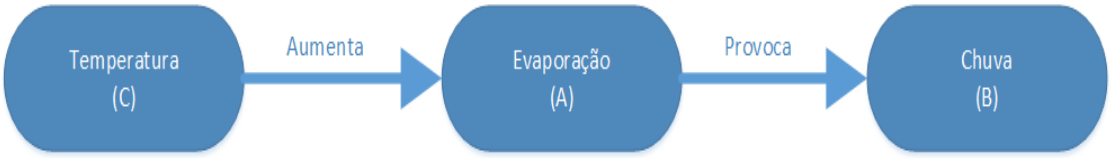


FIGURA 7: CADEIA DE INFLUÊNCIAS
FONTE: OS AUTORES (2015)

Nesse exemplo, inserimos o conceito C em nosso sistema, assim a variável A que era independente passa a ser dependente, pois é influenciada por uma variável do sistema. Conforme vamos adicionando conceitos em nosso modelo, vamos identificando novas conexões e assim conseguimos “Mapear” o fluxo de influência, essa ferramenta será largamente utilizada nas práticas que iremos adotar nesse trabalho.

Andrade (2006) define algumas características para o pensamento sistêmico tais como, mudança de partes para todo, mudança de objetos para relacionamentos, mudança de causalidade linear para circularidade dos fluxos e relações, mudança de estrutura para processos e mudança de quantidade para qualidade.

Segundo o mesmo autor, a característica de mudança de causalidade linear para circularidade dos fluxos e relações, dentro do pensamento cartesiano se justifica, uma vez que o entendimento da realidade científica passa do levantamento de dados para o reconhecimento de padrões, e somente aí passa para explicações causais dos fenômenos.

Porém, a tradição de estudos de causalidade sempre esteve limitada ao reconhecimento dos padrões lineares de conexão, por considerar implicitamente as relações de causalidade circular, anômalas ou de exceção, ficando elas em segundo plano.

Modelos conceituais são fortes aliados na interpretação do sistema, assim tornando-se uma ferramenta eficaz na identificação de problemas. Porém modelos conceituais podem ser difíceis de ser interpretados, mas é possível deixar o modelo mais compreensível através do teor da influência que pode ser representado da seguinte forma:

FIGURA 8: INFLUÊNCIA INSTANTÂNEA
FONTE: OS AUTORES (2015)

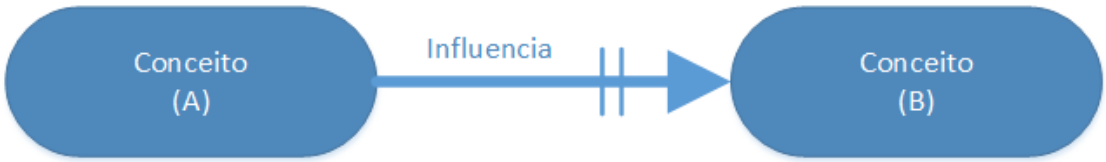


FIGURA 9: INFLUÊNCIA COM ATRASO
FONTE: OS AUTORES (2015)

A primeira imagem é a influência instantânea, já aborda anteriormente. A segunda imagem é referente à influência com atraso, ou seja, uma influência que tem causa e efeito, porém não de forma imediata, mas sim em longo prazo (período estipulado no modelo conceitual). Com essa distinção, somos capazes de diferenciar facilmente o grau de importância de cada influência. Porém, as influências ainda podem gerar confusão, pois a aplicação do verbo pode ter conotação negativa, para facilitar a identificação das influências iremos utilizar os seguintes padrões:

FIGURA 10: INFLUÊNCIA DIRETA – 1
FONTE: OS AUTORES (2015)

FIGURA 11: INFLUÊNCIA DIRETA – 2
FONTE: OS AUTORES (2015)

FIGURA 12: INFLUÊNCIA INVERSA – 1
FONTE: OS AUTORES (2015)

FIGURA 13: INFLUÊNCIA INVERSA – 2
FONTE: OS AUTORES (2015)

As imagens (FIGURA 10) e (FIGURA 11) representam uma influência direta, ou seja, quando o verbo aplicado e seu significado tem teor positivo para o sistema. As imagens (FIGURA 12) e (FIGURA 13) representam uma influência inversa, ou seja, quando o verbo aplicado e seu significado tem teor negativo para o sistema. O cabeçalho em cada conceito representa o direcional do fluxo do próprio conceito, assim como o direcional esperado de acordo com o verbo aplicado.

Com os conceitos apresentados até então, somos capazes de construir elementos que representam nossas ideias, assim facilitando a discussão sobre o sistema analisado. A representação física vai nos ajudar a entender o fluxo do sistema, assim como também torna viável utilizar conceitos de dinâmica de sistemas

para encontrar pontos que podem ser alterados para melhor o sistema como um todo.

2.3.5. Dinâmica de Sistemas

A análise realizada até o momento tem caráter teórico, assim não poderia ser descrita como uma ferramenta de aprimoramento de software, porém a partir do momento em que aplicarmos conceitos de dinâmica de sistemas, tornamos o nosso estudo um material com fundamentação em dados, assim, baseando-se em valores podemos tomar decisões não arbitrárias.

Para fundamentarmos nossa análise com valores concretos, utilizaremos as ferramentas de estruturação já apresentadas, porém realizando uma nova análise sobre as influências.

Pois de acordo com Goodman (1989), o foco principal da dinâmica de sistemas é a busca da compreensão da estrutura e do comportamento dos sistemas compostos por enlaces de *feedback* interagentes, utilizando-se diagramas de enlace causal e os diagramas de fluxo.

As técnicas apresentadas na sessão de pensamento sistêmico são fundamentais para construirmos nosso digrama de enlace causal, através dele podemos começar o processo de identificação dos fluxos das variáveis do sistema.

De acordo com Okada (2008), O mapeamento conceitual é uma técnica que estabelece relações entre conceitos e sistematiza o conhecimento significativo.

Logo, mapas conceituais são capazes de ampliar nossa visão sobre o sistema, mas não são suficientes para definirmos os valores de causa e efeito. As influências construídas anteriormente precisam ser ponderadas e analisadas tanto isoladamente como em conjunto, assim saberemos o fator de relevância de cada variável, tal quais as estratégias que podemos adotar para melhorar esse sistema através da manipulação dos valores das variáveis.

A ideia de mudanças através da manipulação é difundida por Senge (1990), onde aponta como a essência da mudança de mentalidade que o pensamento sistêmico gera duas questões: a primeira uma nova forma de pensar sobre as inter-relações dos elementos da realidade, passando a perceber cadeias circulares ao

invés de cadeias lineares de causa-efeito e a segunda é passar a ver processos de mudanças ao invés de mudanças instantâneas.

Assim, mudanças calculadas sobre uma variável do sistema pode gerar uma mudança instantânea, mas o real valor dessa estratégia é visar às mudanças em longo prazo, as quais geralmente tem um valor mais significativo do que as mudanças imediatas.

Dentro deste contexto, Senge et. al. (1996) sugerem o uso de diagramas de enlace causal como instrumento de linguagem. Partindo do fato de que a linguagem molda o pensamento, uma linguagem que trate mais adequadamente as complexidades dinâmicas da realidade poderá trazer uma forma de pensamento mais efetiva. Este diagrama enfatiza a simplicidade da representação do comportamento de um sistema, através do mapeamento dos seus elementos formadores e dos relacionamentos entre eles, isto é, de que forma um elemento influencia o comportamento de outro.

Até então, sabíamos identificar a influência entre dois elementos do sistema, porém para identificar o teor dessas influências, ou seja, a importância real que uma variável possui sobre a outra e consecutivamente sobre o sistema, precisamos identificar os valores dessa variável. Algumas vezes esses valores não são evidentes ou ponderáveis, mas esse problema pode ser contornado utilizando valores estipulados para essa variável de acordo com valores reais de outras variáveis do sistema.

Para fins didáticos, vamos observar um exemplo de um contexto diferente da análise desse trabalho: Imagine um laboratório de informática como um sistema a ser observado, onde a temperatura é nossa variável dependente e importante para o sistema, o valor dessa temperatura pode ser controlado através de um equipamento de ar condicionado, valores baixos são extremamente benéficos para a integridade dos computadores, porém deve se levar em consideração o bem estar dos usuários e também o consumo de energia. O limiar de ambiente para as máquinas é com a temperatura inferior a 15 graus, a temperatura ambiente é de 25 graus, o laboratório comporta até 40 pessoas, quanto maior a quantidade de usuários mais se elevará a temperatura do ar e por fim, o ar condicionado gasta mais energia conforme a temperatura desejada seja mais distante da temperatura ambiente.

Primeiramente precisamos identificar as variáveis que compõe o sistema, a partir da instância do problema pode-se concluir que a Temperatura, Consumo e

Satisfação dos Usuários são variáveis claras do sistema. Porém a maioria dos sistemas é composta por altas quantidades de variáveis, as quais são encontradas quando vamos além da instância do problema. Olhando com atenção para o sistema como um todo, poderíamos inferir que o número de usuários, a temperatura ambiente, a umidade do ar e etc. são variáveis presentes no sistema e para análises precisas necessitaríamos utiliza-las, caso essas variáveis fossem comprovadas como relevantes. Somente a título didático, utilizaremos o número de usuários como variável relevante que ajudará a compor o nosso mapa.

Logo, iremos compor uma análise baseada em:

- Temperatura
- Consumo de Energia
- Satisfação
- Número de Usuários

Com as variáveis definidas, precisamos confrontá-las uma a uma para descobrimos suas conexões, algumas ferramentas como, por exemplo, uma tabela de influência pode nos ajudar a realizar essa função.

Tabela 1: INFLUÊNCIA ENTRE CONCEITOS (LABORATÓRIO)

	Temperatura	Satisfação	# Usuários	Consumo
Temperatura		+	-	+
Satisfação	-		+	-
# Usuários	+	-		-
Consumo	-	-	-	

FONTE: OS AUTORES (2015)

Os sinais de “+” indicam algum tipo de influência considerável, os sinais de “-” indicam que não existe influência considerável. Mas a pergunta mais intuitiva e fundamental para a dinâmica de sistemas é: “Como identificamos essas influências?”. Apesar de parecer muito simples e pertinente, a resposta é um pouco mais complexa; O mapa conceitual é o resultado da visão do analista que o desenvolve, ou seja, fruto do seu ponto de vista e experiência.

Pois de acordo com Novak (1999), não pode existir mapas conceituais “perfeitos”, no máximo o que nos esforçamos em representar são maneiras corretas de representar hierarquias de relações, em torno das quais giram os conceitos e mapas conceituais.

Ou seja, existem milhares de formas de construir um mapa conceitual sobre determinado sistema. Mapas conceituais diferentes podem culminar no mesmo resultado dependendo das análises aplicadas, assim como mapas conceituais iguais podem apresentar resultados diferentes dependendo do ponto de vista aplicado pelo analista. O que define se o resultado é realmente um bom resultado é apenas um fator, a experiência.

Um analista experiente na construção de modelos conceituais tem maior facilidade na construção de modelos claros e objetivos, assim como também estão mais aptos em retirar bons resultados de um mapa conceitual confuso ou complexo.

Voltando ao nosso exemplo, de acordo com os dados dispostos (Tabela 1), possuímos as seguintes influências:

1- Temperatura - Satisfação
2- Temperatura - Consumo de Energia
3- Satisfação - Número de Usuários
4- Número de Usuários - Temperatura

Sabemos que existe influência entre os conceitos, porém ainda não identificamos o tipo dessa influência. Se utilizarmos as técnicas apresentadas na sessão de pensamento sistêmico (2.3.4), podemos definir esses valores através da aplicação de ação entre conceitos, ou seja, utilizando um verbo pertinente para conectar os dois elementos. Para melhor entendimento, as ações estarão representadas em maiúsculo:

No caso de número (1), pode-se dizer que aumentar a temperatura, OCASIONA no aumento da satisfação do usuário. Perceba que a variável independente sofreu acréscimo, assim como a variável dependente, logo essa é uma conexão instantânea direta. A análise é fundamentada no valor considerado normal para a temperatura corporal de um ser humano, e os efeitos negativos que o mesmo tem se exposto a temperaturas discrepantes a sua própria.

No caso de número (2), é possível afirmar que a diminuição da temperatura AUMENTA o consumo de energia. Note-se que a variável independente sofreu redução, mesmo com um verbo com sinônimo a acréscimo, já a variável dependente, embora não explicita recebe acréscimo, esse definido pelo próprio verbo. Logo, essa é uma conexão instantânea inversa. Essa análise leva em consideração as reações mecânicas comprovadas cientificamente, onde para um

aparelho de ar condicionado tende a gastar mais energia para manter temperaturas distantes da temperatura ambiente.

No Caso de número (3), foi considerado que caso seja alta a satisfação dos usuários, pode ACARRETAR em um número maior de usuários. É possível retirar varias conclusões dessa passagem, em nossa análise foi considerado que uma satisfação alta pode influenciar no número de usuários, esse tipo de conexão incerta é extremamente importante, porém não é instantâneo, o resultado será obtido em longo prazo. Como houve um acréscimo em ambas as variáveis, o tipo de conexão é direta. Ficando então caracterizada como conexão direta com atraso. Essa conclusão tem como base o nosso comportamento social, onde frequentamos locais de acordo com a satisfação que ele pode nos proporcionar.

No último caso, de número (4), é possível afirmar que o número elevado de usuários AUMENTA a temperatura do ambiente, assim, um caso fácil de identificar, pois ambas as variáveis tem direção positiva e o feito é imediato, logo a conexão é instantânea e direta. Essa decisão foi tomada baseada em leis da termodinâmica, pois um elevado número de indivíduos altera a temperatura do ambiente de acordo com sua própria temperatura corporal.

Em todos os casos foram tomadas decisões pessoais em relação a cada análise, porém é possível notar que nenhuma delas foi arbitrária ou aleatória, pois um mapa conceitual consistente deve ter suas conexões com fundamentação teórica.

Assim, de acordo com os conceitos levantados e suas conexões, nosso mapa conceitual pode ser representado graficamente como (FIGURA 14):

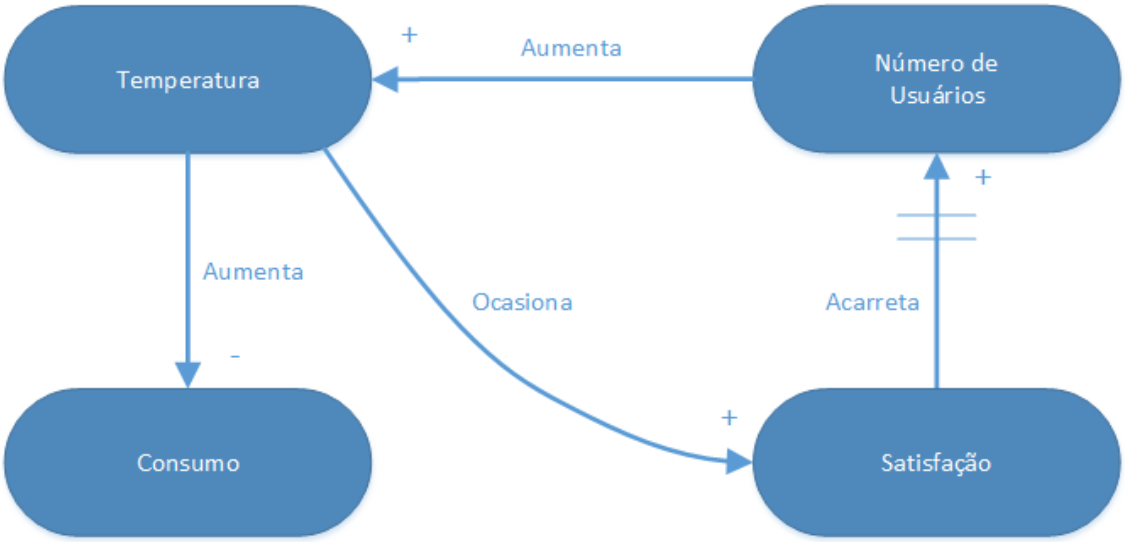


FIGURA 14: MODELO CONCEITUAL (LABORATÓRIO)
FONTE: OS AUTORES (2015)

O mapa conceitual apresentado (FIGURA 14) nos permite observar o sistema com mais clareza, a partir desse material somos capazes de visualizar as possíveis mudanças de comportamento nos conceitos quando um evento ocorre no sistema. Os impactos causados por uma alteração podem ser provisionados, porém a previsão ainda é teórica e totalmente relativa à experiência do analista.

Para atingirmos nosso objetivo, que é ter opções de tomada de decisões sólidas, precisamos trabalhar com valores a serem calculados. Para isso vamos adotar outra técnica de dinâmica de sistema, que consiste estipular valor para as variáveis imponderáveis através de causa e efeito com as variaríeis que realmente possuem valores reais.

Existem dois tipos de valores reais, o valor real concreto e o relativo. O valor real concreto trata-se do valor que possuímos controle, ou seja, podemos alterá-lo sempre que for necessário. Um valor real relativo é um valor ponderável, porem seu valor é alterado automaticamente conforme as alterações no sistema, ou seja, não é possível alterá-lo diretamente. Apesar de não termos controle direto sobre um valor relativo, podemos definir os seus limites ou alterá-lo através de manipulação de valores concretos.

Em nosso exemplo possuímos quatro variáveis: Temperatura, Consumo de Energia, Satisfação e Número de usuários. Vamos observar cuidadosamente cada elemento.

- Temperatura: Conforme dito na instância do problema, podemos alterar a temperatura a qualquer momento através de um equipamento de ar condicionado, logo essa variável do nosso modelo conceitual é caracterizada como um valor real e concreto. Também podemos chamá-lo de fonte, devido ao fato que seus valores serem independente e também pelo fato que os seus valores alteram o fluxo do sistema.
- Consumo de Energia: Essa variável deve ser observada com atenção, pois é intuitivo acreditarmos que possuímos total controle sobre esse elemento. “Pois se controlamos a temperatura do ar, logo controlamos o consumo de energia”, nessa passagem que o equivoco fica evidente, pois controlamos a Temperatura, a qual é uma variável do nosso sistema e de maneira alguma podemos sintetizar duas variáveis em um único evento, pois a temperatura tem impactos diferentes no sistema no qual o consumo de energia não está diretamente relacionado. Assim o consumo de energia é uma variável real e relativa.
- Satisfação: Esse é um caso onde não é possível atribuir valores a variável, podemos observar o comportamento do sistema e estipular alterações sobre esse elemento, para fins de análise podemos estipular valores fictícios para definir esse comportamento. Como a variável não tem caráter concreto, esse elemento é caracterizado como uma variável abstrata, ou seja, não real.
- Número de usuários: Esse é um valor real, porém não pode ser definido como concreto. Pois a quantidade de usuários que frequentam a estação não pode ser controlada, assim os valores nas observações não podem ser manipulados, porém podem ser ponderados através da análise do comportamento do sistema. Assim, apesar de ser uma variável relativa, podemos controlar o seus limites conforme dito anteriormente. Esse fator pode ser importante para definir estratégias.

Até então definimos quais os tipos desses valores, mas de fato ainda não atribuímos valor a nenhuma das variáveis. Esse tipo de análise pode ser feito de

diversas formas, como: observação do sistema por períodos, enquete com os agentes do sistema, simulação computacional e etc. Para esse exemplo iremos adotar uma simulação da observação do sistema num período de 15 dias, onde variamos diariamente o valor da temperatura para podermos estudar o impacto dessa mudança sobre o sistema.

Tabela 2: ANÁLISE INICIAL (LABORATÓRIO)

ID	Temperatura	Satisfação	#Usuários	Consumo
1	15	?	?	?
2	14	?	?	?
3	13	?	?	?
4	12	?	?	?
5	11	?	?	?
6	10	?	?	?
7	9	?	?	?
8	8	?	?	?
9	7	?	?	?
10	6	?	?	?
11	5	?	?	?
12	4	?	?	?
13	3	?	?	?
14	2	?	?	?
15	1	?	?	?

FONTE: OS AUTORES (2015)

Através da estipulação da variável fonte (temperatura) nos dados dispostos (Tabela 2), podemos analisar no sistema qual foi o impacto dessa variação, para assim podemos definir estratégias referentes a objetivos diversos. A tabela tem um plano de análise que consiste em descobrir os valores para as interrogações através dos métodos citados, ou seja, o número de usuário e o consumo serão valores reais que aparecerão conforme a análise transgredir, já a satisfação pode ser oriundo de uma pesquisa com os usuários.

Como o nosso limite superior para a temperatura foi definido como 15 graus, ao decorrer dos dias (representados por ID) vamos verificar diversos comportamentos do sistema em relação à temperatura. Na seguinte análise (Tabela 3):

Tabela 3: ANÁLISE EM RELAÇÃO À TEMPERATURA (LABORATÓRIO)

ID	Temperatura	Satisfação	#Usuários	Consumo
1	15	10	40	Baixo
2	14	10	37	Baixo
3	13	10	38	Baixo
4	12	9	36	Médio
5	11	9	28	Baixo
6	10	8	35	Médio
7	9	7	36	Médio
8	8	7	36	Médio
9	7	6	37	Alto
10	6	5	34	Alto
11	5	3	33	Alto
12	4	1	35	Alto
13	3	1	39	Muito alto
14	2	1	29	Muito alto
15	1	1	31	Muito alto

FONTE: OS AUTORES (2015)

A estruturação de um modelo conceitual e a ponderação dos conceitos nos possibilita extrair diversas informações em relação ao sistema, porém somente quando traçamos um objetivo de comportamento é que conseguimos visualizar as estratégias. Imagine que em nosso exemplo, toda nossa análise é voltada em relação à economia no consumo de energia, levando os fatos de que é de interesse deixar a temperatura o mais baixo possível, porém sem ser extremamente prejudicial aos usuários. Nesse caso o conceito (Consumo de Energia) torna-se nosso conceito resultado, ou seja, toda manipulação no sistema tem como objetivo ajustar esse conceito de acordo com as necessidades do problema.

Para identificar como a manipulação de uma variável altera o sistema, precisamos confrontar os dados da tabela com o modelo conceitual em busca de padrões, esses comportamentos definidos são conhecidos como arquétipos, existem diversos casos estudados de arquétipos comuns, porém cada sistema é único, podemos utilizar um arquétipo como base da análise, mas as estratégias a serem definidas devem levar em consideração que o padrão seguido pelo sistema é único e pertencente somente a esse sistema.

Com base nos dados (Tabela 3), podemos verificar que conforme diminuímos a temperatura, maior é o consumo e menor é a satisfação do usuário. Ou seja, não conseguiremos ter uma temperatura perfeita para as máquinas ao mesmo tempo em que essa temperatura seja perfeita à satisfação do usuário, assim

teremos que encontrar um meio termo onde o sistema possa continuar operando em harmonia, o que é o objetivo desta análise.

Observe que temos uma variação incomum na faixa de consumo entre as linhas de identificação (3), (4) e (5) (Tabela 3). Na linha de identificação de número 3, o consumo permanece baixo e a satisfação do usuário em valor elevado, ambos a uma temperatura de 13 graus. Já na linha de identificação de número (4), o consumo sofre uma alteração e passa a ser médio quando a temperatura é de 12 graus. O consumo volta a ser baixo quando a temperatura é definida como 11 graus, na linha de identificação de número (5).

Se o nosso objetivo é manter o consumo como baixo, no melhor cenário possível, podemos afirmar que nossa melhor análise em relação à temperatura foi observada na linha de identificação de número (5). Mas como foi possível uma temperatura de 12 graus, ter um efeito inferior à temperatura de 11 graus? A resposta só pode ser observada quando levamos em consideração o sistema como um todo. É possível notar que entre as linhas (4) e (5) (Tabela 3), ocorre uma queda consideravelmente no número de usuários, assim isso pode ser utilizado como uma técnica para limiar o sistema. Para isso precisamos fazer uma nova análise, agora em relação ao número de usuários, sendo que o valor da temperatura será próximo ao que foi encontrado no melhor caso.

Tabela 4: ANÁLISE EM RELAÇÃO AO NÚMERO DE USUÁRIOS (LABORATÓRIO)

ID	Temperatura	Satisfação	#Usuários	Consumo
1	11	9	28	Baixo
2	11	9	29	Baixo
3	11	9	30	Baixo
4	11	9	31	Médio
5	10	9	31	Médio
6	10	9	30	Baixo

FONTE: OS AUTORES (2015)

Nessa segunda etapa, verificamos que é possível manter o consumo em um estado baixo, limitando o número de usuários (Tabela 4). As primeiras quatro análises definiram o valor limite para o número de usuários, as últimas duas análises são uma tentativa bem sucedida de abaixar a temperatura de acordo com o número de usuário previamente estabelecido.

Logo, podemos concluir que uma possível estratégia para manter o consumo em um estado considerado como baixo, seria limitar o número de usuários para 30 pessoas, com isso poderíamos manter a temperatura em 10 graus, trazendo uma estabilidade para todo o sistema.

Nossa análise foi realizada de acordo com o padrão observado no modelo conceitual. Embora o padrão do nosso sistema seja único, podemos classificá-lo de acordo com outra escala de padrões definidas para a dinâmica de sistemas. Como dito anteriormente, esses padrões são conhecidos como arquétipos.

De acordo com Senge (2000), arquétipos sistêmicos são estruturas sistêmicas genéricas compostas por relações de causa-efeito cíclicas que se repetem em diferentes contextos, geralmente, sem que as pessoas tenham consciência de seus efeitos na situação em questão. Por terem um comportamento previsível, a revelação destas estruturas pode inspirar estratégias de ação eficazes para as situações problemáticas que elas representam.

Entender os arquétipos pode ser uma maneira de facilitar a nossa análise, assim como certamente é uma maneira de se especializar em dinâmica de sistemas. A quantidade de arquétipos existentes é incalculável, visto que visões diferenciadas podem resultar em padrões diferentes, assim sendo incontável o número de possíveis resultados. Entretanto, existe um número finito de arquétipos que são denominados recorrentes.

Para compreender como os arquétipos foram classificados, primeiramente precisamos nos situar sobre alguns conceitos básicos de padrões em dinâmica de sistemas.

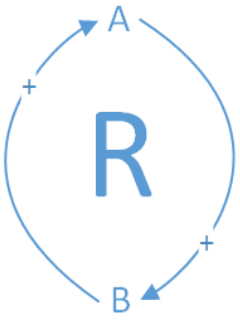


FIGURA 15: ENLACE DE REFORÇO
FONTE: OS AUTORES (2015)

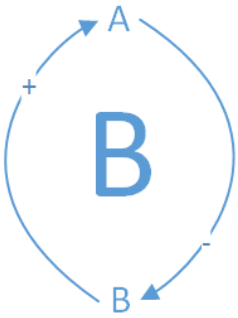


FIGURA 16: ENLACE DE BALANÇO
FONTE: OS AUTORES (2015)

A primeira estrutura (FIGURA 15) refere-se a um enlace de reforço, facilmente reconhecido em um sistema devido ao fato de ser um subsistema com um número par de conexões inversas. Um enlace de reforço recebe esse nome devido ao fato de que um conceito alimenta o outro, e no fim das contas o conceito também recebe agregação, assim teoricamente o sistema se “reforça”. A segunda estrutura (FIGURA 16) também pode ser facilmente reconhecida em um subsistema, pois é constituída de um número impar de conexões inversas. O enlace recebe o nome de balanço, devido a existir uma unidade de oposição. Assim mesmo que um conceito receba acréscimos, ao longo do sistema teremos um decréscimo em algum outro conceito, o que faz o sistema entrar em equilíbrio.

2.3.5.1. Arquétipos

Senge (1990) sugere padrões de comportamento sistêmico baseado em arquétipos a partir da construção e uso de modelos causais das relações que ocorrem no interior de qualquer sistema.

A ideia não é construir modelos conceituais baseados nos arquétipos, porém é possível identificá-los no interior do seu modelo previamente definido, isso pode ajudar a identificar padrões de comportamento. Atualmente temos 11 arquétipos considerados como “comuns”, os quais foram nomeados de acordo com suas características:

1. Limite ao crescimento
2. Principio da atratividade
3. Solução quebra galho
4. Transferência de responsabilidade
5. Escalada
6. Derivada de metas
7. Sucesso para os bens sucedidos
8. Equilíbrio com defasagem
9. Tragédia dos comuns
10. Adversários acidentais
11. Crescimento e subinvestimento

2.3.5.1.1. Limite ao crescimento

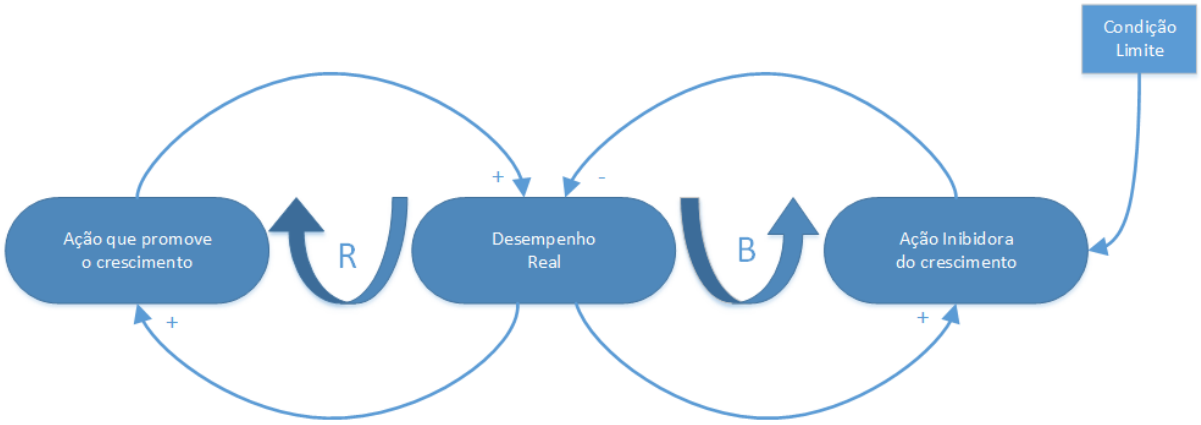


FIGURA 17: LIMITES AO CRESCIMENTO
FONTE: OS AUTORES (2015)

Consiste em um ciclo virtuoso onde inicialmente o crescimento é exponencial, porém possui fatores limitantes. Conforme o passar do tempo o balanceamento é expressivo, principalmente quando o crescimento já atingiu o seu auge, dando inicio a uma frenagem rápida que pode até culminar em reversão.

2.3.5.1.2. Principio da atratividade

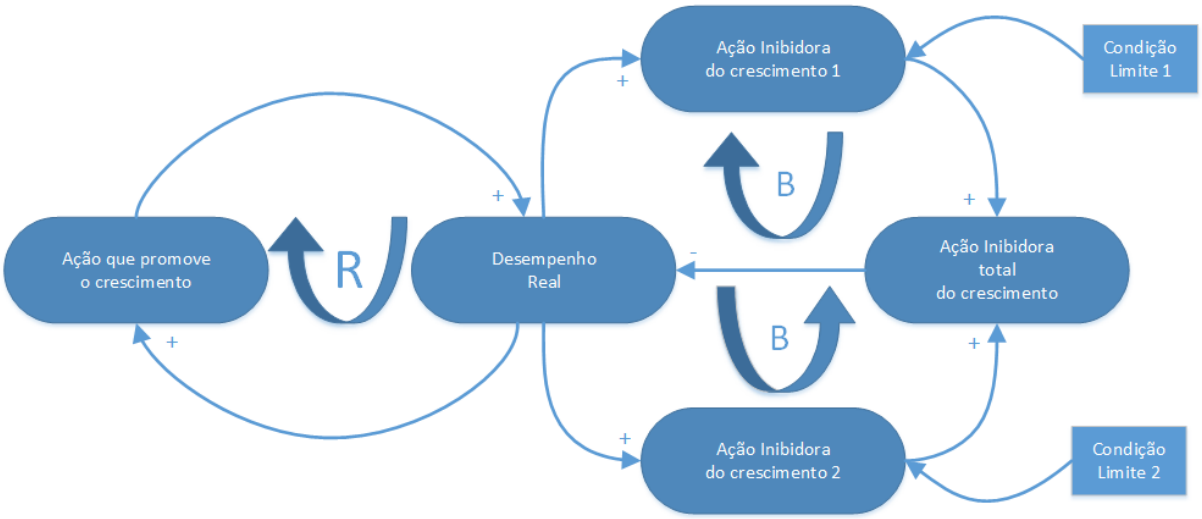


FIGURA 18: PRINCÍPIO DA ATRATIVIDADE
FONTE: OS AUTORES (2015)

É um caso especial de limites ao crescimento (FIGURA 17), onde o crescimento não possui limites, porém existem ações inibidoras a evolução. O principio é o mesmo, constituído de um crescimento exponencial, porém tem seu balanço controlado pelo reforço que é constante, pouco percebido no inicio do ciclo, porém fundamental para sua frenagem.

2.3.5.1.3. Solução quebra galho

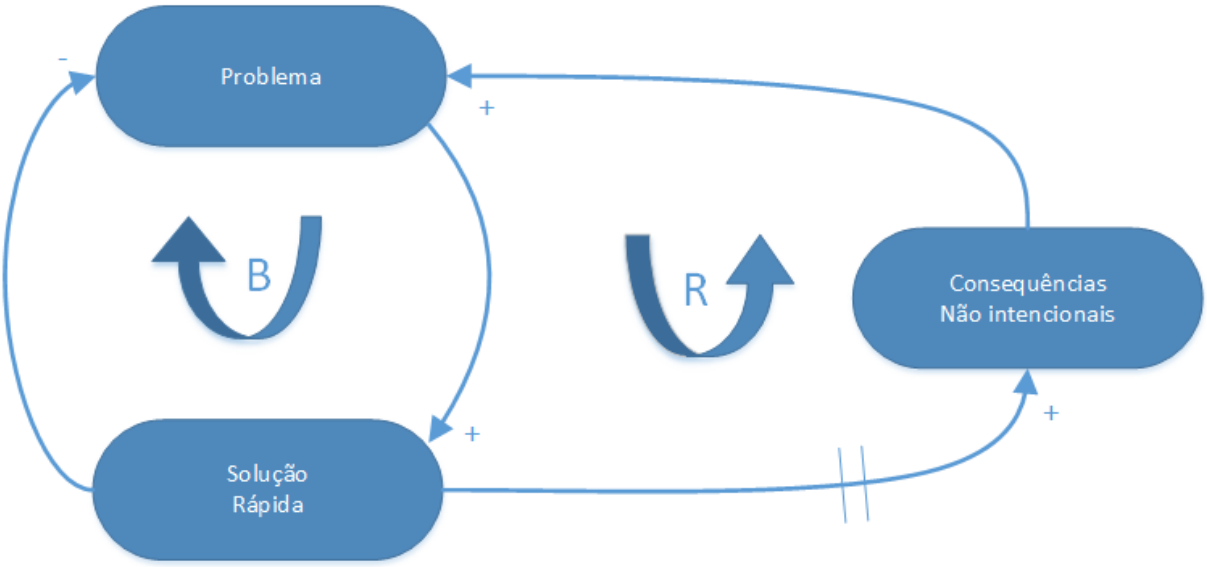


FIGURA 19: SOLUÇÃO QUEBRA GALHO
FONTE: OS AUTORES (2015)

Seu nome é reflexo de uma expressão popular, onde a solução é encontrada rapidamente, porém é apenas temporária. Produz não intencionalmente consequências em longo prazo, fazendo surgir diversos problemas, inclusive o ressurgimento da causa.

2.3.5.1.4. Transferência de responsabilidade

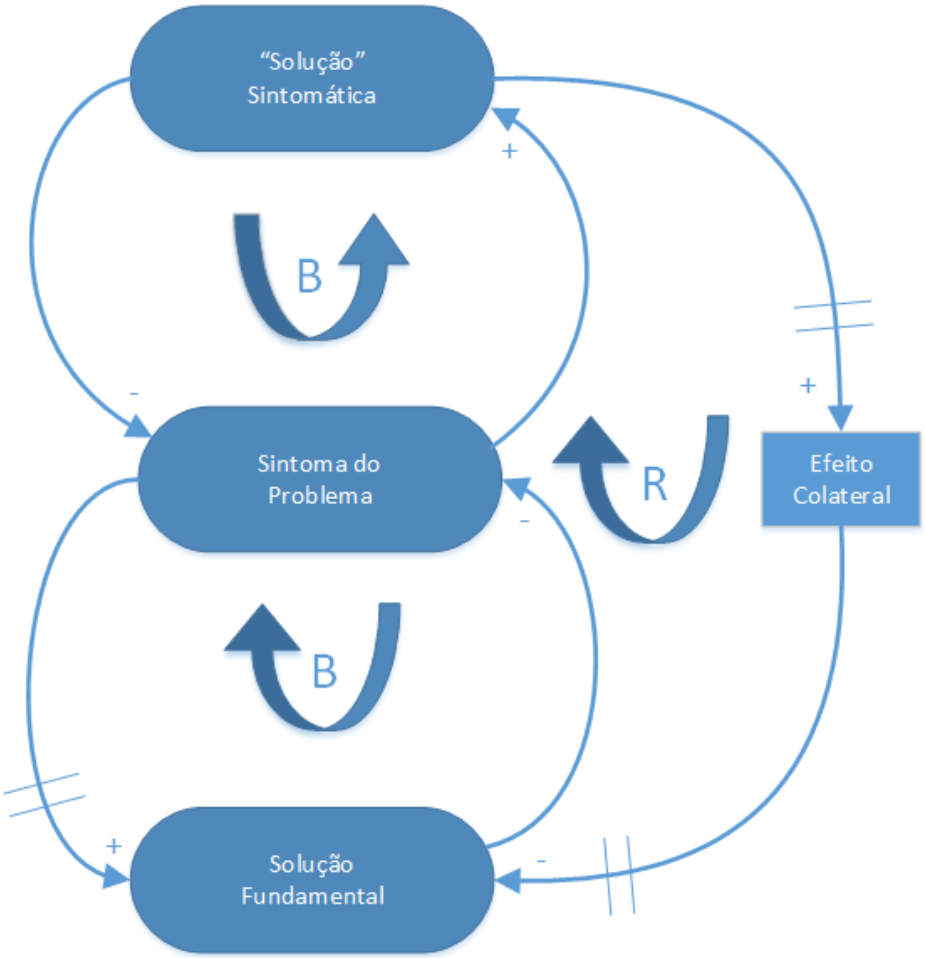


FIGURA 20: TRANSFERÊNCIA DE RESPONSABILIDADE
FONTE: OS AUTORES (2015)

É uma solução imediata, onde os resultados positivos podem ser observados logo nos primeiros ciclos. Porém, à medida que esse enlace é aplicado, inviabiliza medidas corretivas de longo prazo podendo causar graves danos ao sistema. Logo, deve ser considera como uma estratégia temporária quando a necessidade de ação rápida, mas não aconselhada a ser utilizada em um sistema que necessita sobreviver.

2.3.5.1.5. Escalada

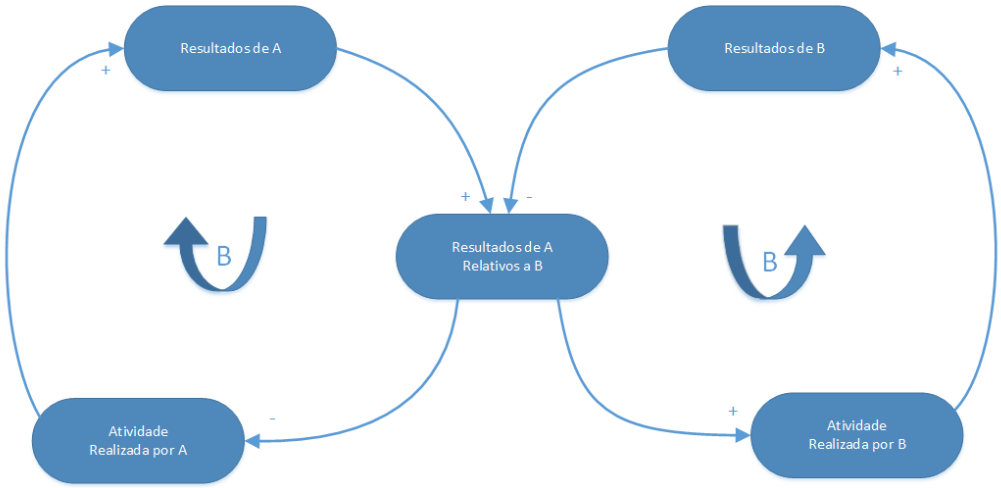


FIGURA 21: ESCALADA
FONTE: OS AUTORES (2015)

Uma solução considerada explosiva, pois as conexões forçam um conceito a tirar vantagem sobre o outro, sendo assim, tem um poder agressivo de solução. Mas como os enlaces geram esforços demasiados, esse tipo de solução desgasta o sistema ao longo do tempo.

2.3.5.1.6. Derivada de metas

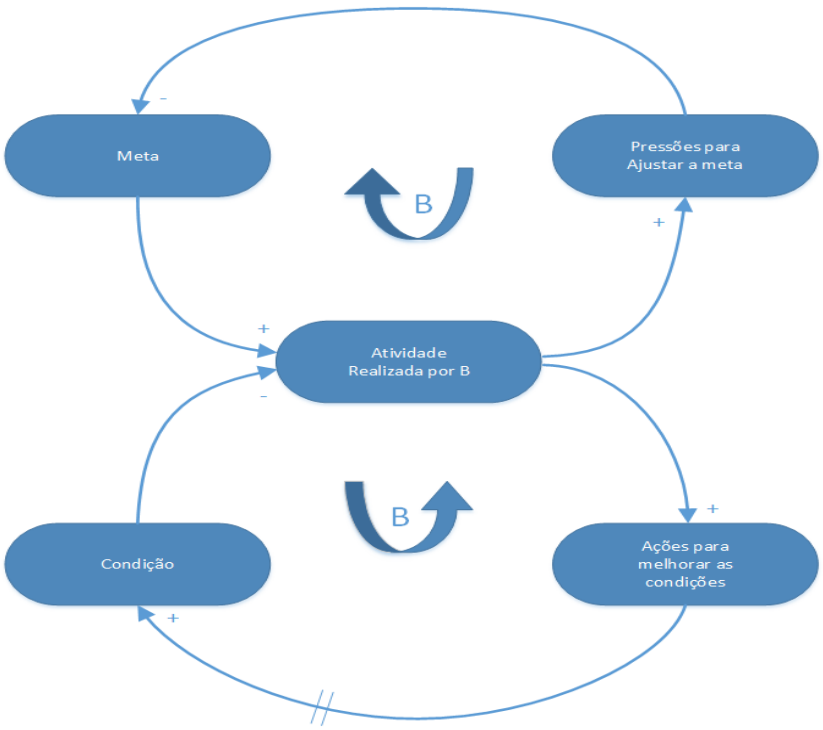


FIGURA 22: DERIVA DE METAS
FONTE: OS AUTORES (2015)

É um caso diferenciado de transferência de responsabilidade (FIGURA 20), onde a solução também é obtida em curto prazo, porém sacrificando um objetivo de longo prazo. É caracterizada por manter certo nível de pressão entre a meta desejada e a condição atual, causando uma tensão ao sistema. O aumento da tensão dá origem a esforços para melhorar a condição atual, porém o balanço em longo prazo é mais incisivo.

2.3.5.1.7. Sucesso para os bens sucedidos

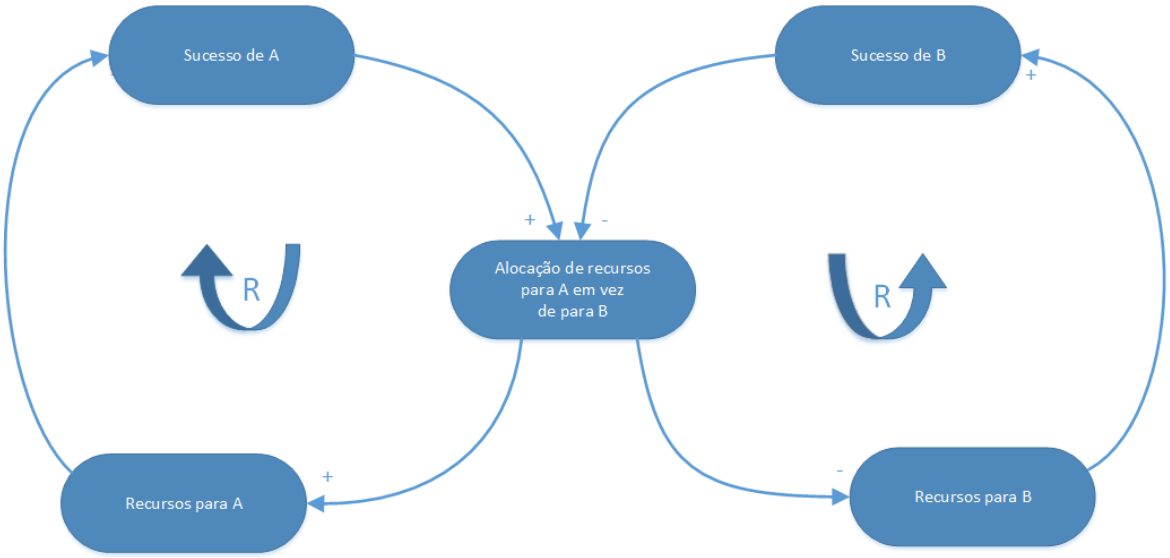


FIGURA 23: SUCESSO PARA OS BENS SUCEDIDOS
FONTE: OS AUTORES (2015)

Significa tomar uma estratégia de prioridade, onde uma das partes é priorizada e utiliza os recursos da outra. Dependendo do seu objetivo e não tendo relevância o declínio explicitado da parte desfavorecida, é uma ótima solução para manter um objetivo especifico.

2.3.5.1.8. Equilíbrio com defasagem

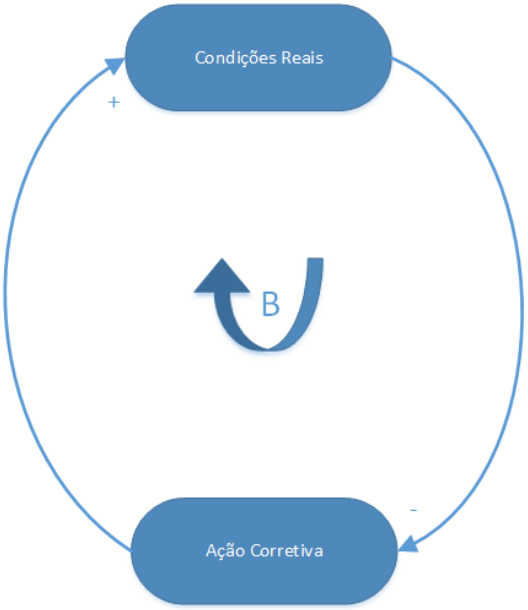


FIGURA 24: EQUILÍBRIO COM DEFASAGEM
FONTE: OS AUTORES (2015)

Trata-se de um *feedback* com atraso, tornando assim a depreciação do sistema menos acentuada, os impactos são mais imperceptíveis, porém prejudica os reforços e melhorias do sistema. Considerada uma estratégia de manutenção de um sistema com prazo de validade definido.

2.3.5.1.9. Tragédia dos comuns

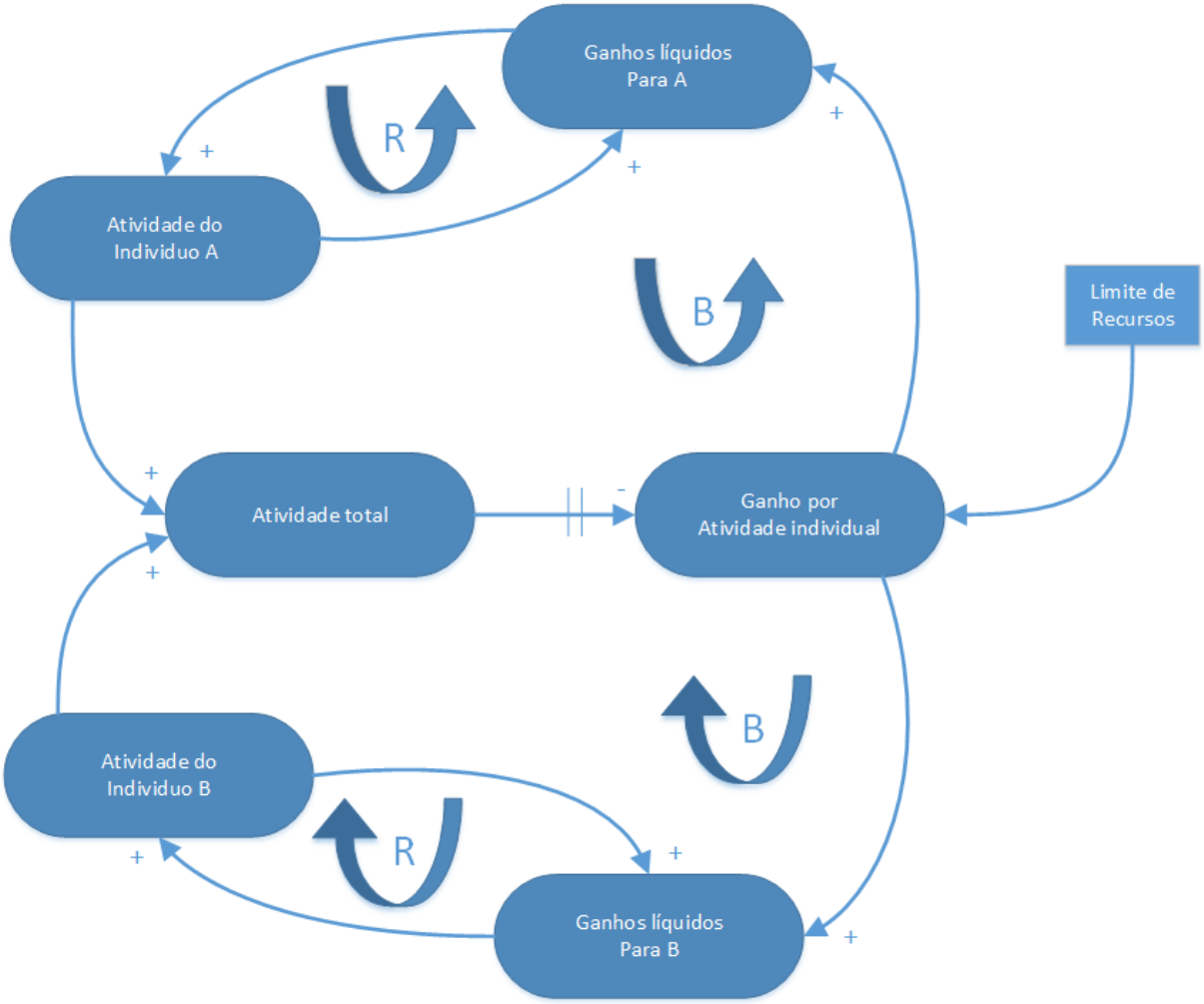


FIGURA 25: TRAGÉDIA DOS COMUNS
FONTE: OS AUTORES (2015)

Caracterizado pelo compartilhamento de um recurso entre enlaces distintos, o qual é abundante inicialmente. Esse recurso em comum ajuda a manter o sistema em equilíbrio, pois com o *feedback* os esforços são intensificados, porém a queda é progressiva e pode tornar o uso do recurso inviável.

2.3.5.1.10. Adversários acidentais

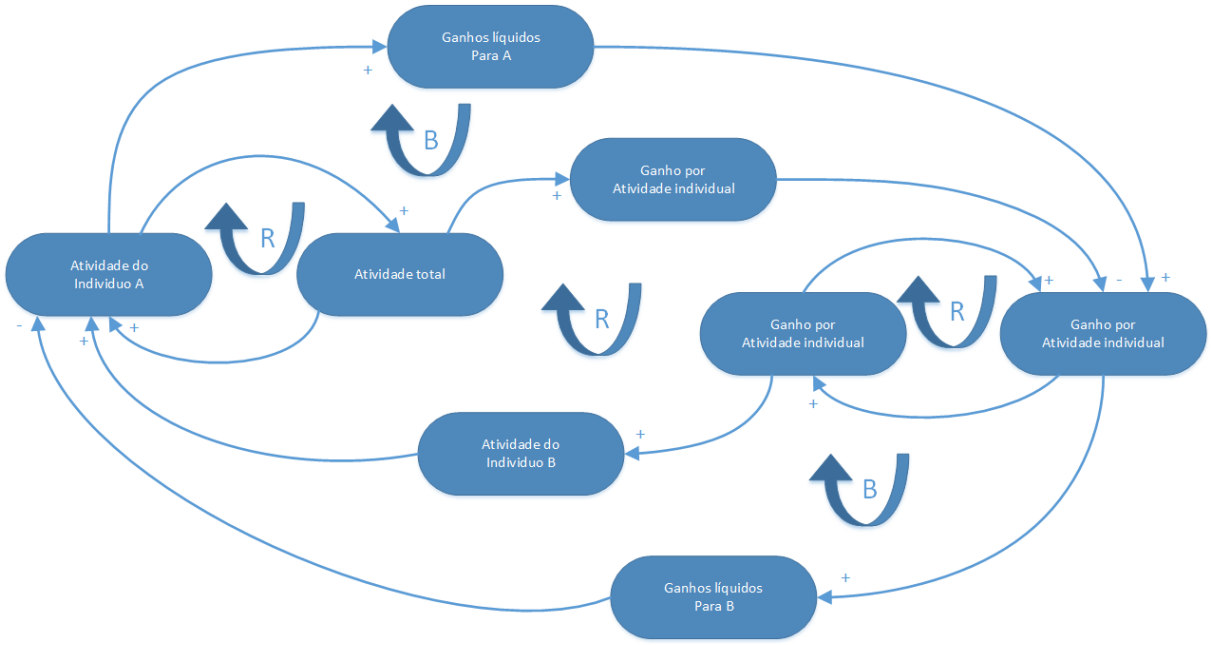


FIGURA 26: ADVERSÁRIOS ACIDENTAIS
FONTE: OS AUTORES (2015)

Estratégia que utiliza um reforço cooperativo entre os enlaces, onde o crescimento é obtido em paralelo, porém quando um enlace trabalha em favorecimento próprio, reduz o sucesso do outro e por consequência a influência positiva que recebia pela cooperação. Uma estratégia que deve ser utilizada somente quando há uma necessidade real de estabilidade entre enlaces.

2.3.5.1.11. Crescimento e subinvestimento

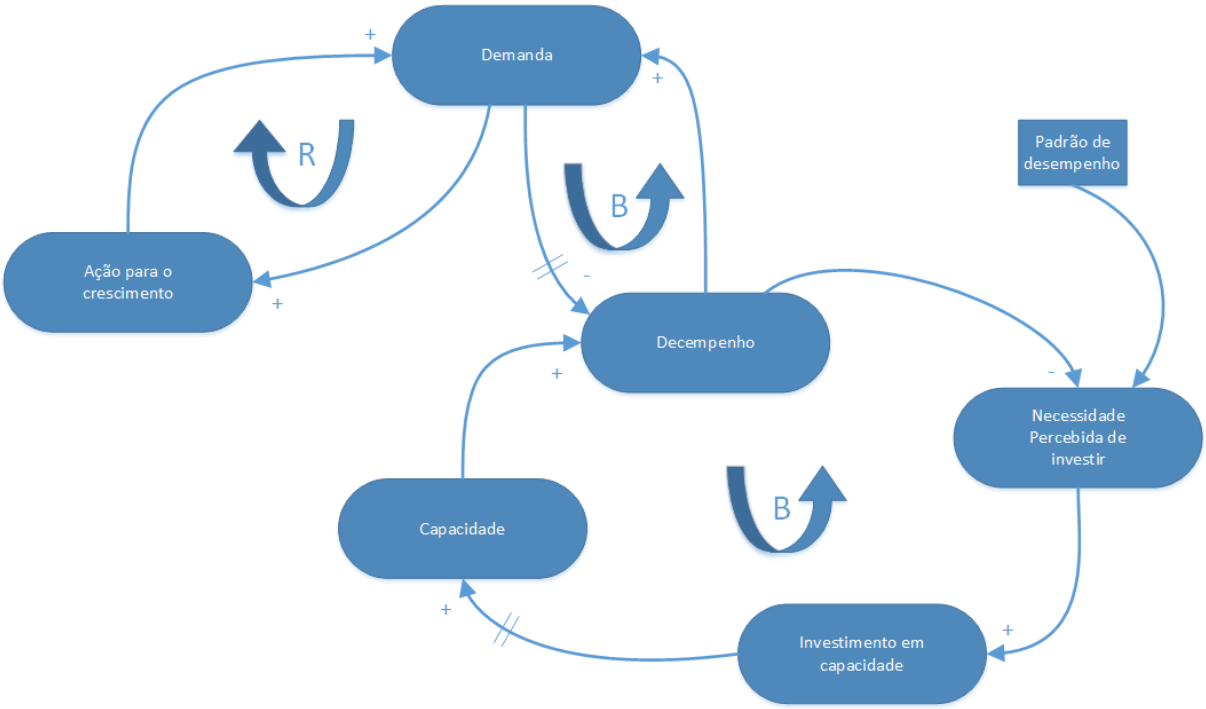


FIGURA 27: CRESCIMENTO E SUBINVESTIMENTO
FONTE: OS AUTORES (2015)

Estratégia para quando nos deparamos com uma estagnação de crescimento, onde uma interferência externa força a evolução, de moto imediato e agressivo. Essa interferência está sujeito ao ambiente e momento em que o sistema atravessa, ou seja, não é viável a todo o momento. Devido à inserção de novos fatores, o sistema necessitará de tempo para se reajustar, podendo até causar impactos negativos nos reforços de longo prazo.

Os arquétipos apresentados não devem ser encarados como padrões para construção de um modelo conceitual, são padrões que devem ser observados em subsistemas do seu modelo conceitual já montado. Identificar esses arquétipos pode facilitar na definição de estratégias, assim tornando suas decisões mais precisas.

3. REALIZANDO A ANÁLISE

Nesse capítulo será apresentada uma análise sobre um sistema em desenvolvimento que emprega a modelagem SCRUM. As técnicas aqui utilizadas serão detalhadamente apresentadas e descritas. Tudo que será efetuado tem base na fundamentação teórica fornecida no capitulo (2).

3.1. ANALISANDO A MODELAGEM UTILIZADA

É possível realizar uma análise sistêmica para qualquer uma das modelagens existentes. Pois enquanto houver um modelo de desenvolvimento, haverá padrões, os quais podem ser identificados e estudados através da dinâmica de sistemas. Porém, uma análise genérica sobre modelagens em geral seria pouco eficiente devido a cada modelagem ter suas próprias peculiaridades, assim como uma análise genérica de apenas uma modelagem não pode ser definida como uma ferramenta para a melhoria dessa modelagem. Pois sistemas distintos que utilizam a mesma modelagem podem possuir comportamentos diferentes, assim é necessário realizar análises individualmente para cada sistema. Logo, os exemplos que serão abordados nesse trabalho, têm caráter didático, não será apresentada uma análise perfeita sobre a modelagem, mas sim técnicas úteis de como realizar essa análise.

Para realizar a análise sobre um ambiente de desenvolvimento de software, primeiramente precisamos conhecer alguns detalhes básicos sobre o ambiente, não precisamos nos aprofundar nos objetivos da organização em relação à análise. Pois, nosso resultado será um mapa de estratégias a seguir, um modelo conceitual completo envolvendo todos os potenciais problemas, logo, é possível extrair todas as potenciais soluções. Desde que as informações pertinentes sejam utilizadas na construção do modelo.

Visto a importância de levantar os dados do sistema, vamos observar atenciosamente o ambiente. Nossa análise é voltada para um ambiente que utiliza à metodologia SCRUM, assim além das informações que vamos coletar do sistema, não podemos esquecer-nos das definições da modelagem vistas na sessão (2.1.7).

A organização observada desenvolve softwares para outras empresas, no formato de terceirização, onde a empresa contratante gera documentação contendo suas necessidades e solicita desenvolvimento para encontrar soluções em software através dos requisitos apresentados. O que simplifica nosso cliente final, pois independente de quantos usuários utilizam a aplicação desenvolvida, nosso cliente final é apenas a empresa contratante. Definir as responsabilidades é importante para definir os limites do nosso sistema, pois não somos capazes de realizar análises precisas de sistemas que não fazem parte do nosso ambiente. A empresa contratante, a qual será identificada pelo nome de Alpha, será tratada como uma caixa preta, a qual não temos conhecimento dos subsistemas que a compõe, porém podemos observar as entradas e saídas de informações na qual ela interage com o nosso sistema, o qual é um subsistema da organização que iremos observar, vamos chamar nossa organização de Beta.

A empresa Beta possui diversos times com diversas finalidades, cada time é um subsistema da própria empresa, a qual seria nosso sistema completo. Como não iremos analisar um sistema de gerência de global, mas sim modelagens de gerência de desenvolvimento de software, vamos focar em apenas um time especifico da empresa Beta. Esse time é responsável pelo desenvolvimento de aplicações para sistemas locais para a empresa contratante, assim como também é responsável pela parte web que tem a mesma finalidade da aplicação, ambas utilizando o mesmo banco de dados. O time é constituído por oito pessoas, dentre elas:

- Um *Team Leader*
- Dois Analistas de sistemas
- Quatro programadores
- Um *Trainee*

As qualificações do time:

Team Leader:

- *DBA*
- Gestão de projetos
- Noções de lógica computacional

Analista-1:

- Conhecimento avançado de banco de dados
- Especialista em linguagens *web*
- Conhecimento avançado em programação

Analista-2:

- Conhecimento avançado de banco de dados
- Especialista em linguagens para construções de aplicações
- Conhecimento avançado em programação

Programadores:

- Conhecimento avançado em programação
- Conhecimento razoável em banco de dados

Trainee:

- Conhecimento razoável em programação
- Conhecimento básico em banco de dados

O time Beta seguiu a rigor os fundamentos da modelagem SCRUM, pois ao início do projeto foi feito um estudo e definido que essa seria a melhor modelagem para esse tipo de projeto. O desenvolvimento funcionava bem e a equipe entregava os produtos no prazo, porém sempre encontrando dificuldades para fechar tudo dentro do tempo estipulado. Após várias interações, um dos ciclos apresentou atraso devido a eventos não programados, o que gerou preocupação. O problema se repetiu no ciclo seguinte, revelando um problema nesse processo, assim o *Team Leader* constatou que sua equipe apesar de ser eficiente para concluir as tarefas, não estava evoluindo, pois com o decorrer de vários ciclos era esperado que com a experiência adquirida, o processo de desenvolvimento fosse fácil.

Ficou evidente que o sistema estava se deteriorando, o *Team Leader* passou a acompanhar sistemicamente o seu time, pois sabia que mudanças eram necessárias. Visto isso ele partiu para uma análise sistêmica que é proposta por esse trabalho.

A análise construída objetiva de imediato resolver o problema de atraso nos prazos de entrega do produto, porém é de interesse aumentar o valor agregado para diminuir a deterioração do processo, assim evitando que o problema ocorra novamente. A corporação tem condições de disponibilizar mais recursos para o time, porém nenhum recurso deve ser gasto desnecessariamente.

3.2. IDENTIFICANDO AS RELAÇÕES

Como vimos anteriormente no capítulo (2.3.4), um passo importante para começar uma análise sistêmica é o levantamento de variáveis. Para isso, qualquer conceito que possa gerar influência sobre o sistema deve ser considerado. Sabemos que o sistema utiliza a modelagem SCRUM a rigor, mas caso não fizesse uso de todas as funcionalidades descritas no modelo, elas mesmo não presentes deveriam estar presentes na análise. Sendo assim, um bom ponto de partida para o levantamento de variáveis é destacar os conceitos pré-definidos na modelagem utilizada.

Dentre os inúmeros conceitos do SCRUM vistos em (2.1.7), os mais evidentes que causam impacto no sistema são as *Daily Meetings*, Tempos de *Sprint*, Tamanho do time, *Product Backlog* e *Sprint Backlog*. Sabemos que o padrão ideal do modelo traz alguns valores, como: *Daily Meetings* entre 8 minutos e 15 minutos, reuniões do *Sprint* de duas a quatro semanas, tamanhos dos times de aproximadamente nove pessoas, incluindo um *Scrum Master* e um *Team Leader*.

Sintetizando esses conceitos para facilitar e encaixar em nosso modelo conceitual, utilizaremos as reuniões de *Sprint* como “Tempos de *Sprint*”, onde os tempos de *Sprint* somados devem ser inferiores ao prazo de entrega do produto. O *Product BackLog* e o *Sprint Backlog*, geram um material que deverá ser trabalhado em cada ciclo. Para nosso modelo iremos unir esses dois conceitos e chamá-los de “Quantidade de Material de *Sprint*”. Os demais conceitos, como *Daily Meetings* e Tamanho dos times, serão usados na integra.

Nosso projeto está sendo aprimorado através de uma técnica diferenciada do modelo PDCA, o qual visa diminuir o tempo necessário de ação, assim como visto anteriormente na sessão (2.2.1). Também vimos que a situação que desencadeou essa análise requeria alterações no sistema para suprir os prazos estipulados. Por esses fatores a variável “Tempo de desenvolvimento” fará parte no nosso modelo. As demais variáveis são frutos de observações do ambiente atual e do ambiente desejado. E um componente que é comum em qualquer área de desenvolvimento, o qual define um valor ao produto final, é a qualidade, que também

será utilizada no nosso modelo conceitual. Por fim, concluímos o levantamento de variáveis com os conceitos esperados pelo analista, que iniciou esse estudo objetivando melhorias para o seu time. Essa evolução interna será definida como "Valor agregado".

Nosso conjunto de variáveis será constituído por:

- Daily Meeting
- Tempo de Sprint
- Quantidade de material do Sprint
- Tamanho dos times
- Valor Agregado
- Qualidade do produto
- Tempo de desenvolvimento

Existem diversas outras variáveis que compõe esse sistema, quanto maior o número de variáveis pertinentes, mais precisa será nossa análise. Entretanto, as variáveis escolhidas foram definidas como fundamentais e suficientes para construir exemplos didáticos das teorias aqui apresentadas.

Agora com as variáveis já levantadas, torna-se possível realizar as conexões de causas e efeitos. Assim como vimos anteriormente (2.3.5), vamos utilizar o recurso de uma tabela (Tabela 5) para verificar a relação entre os conceitos:

Tabela 5: INFLUÊNCIAS DO SCRUM

	Daily Meeting	#Pessoas	Valor agregado	Tempo de *Sprint*	Qualidade	Tempo de desenvolvimento	Material *Sprint*
Daily Meeting		-	+	-	+	-	-
#Pessoas	+		-	+	-	+	+
Valor agregado	-	+		-	-	+	+
Tempo de *Sprint*	-	-	-		+	-	+
Qualidade	-	-	+	-		-	-
Tempo de desenvolvimento	-	-	-	-	+		-
Material *Sprint*	-	-	-	-	-	+	

FONTE: OS AUTORES (2015)

As ralações obtidas foram:

1. *Daily Meeting* - Valor Agregado
2. *Daily Meeting* – Qualidade

3. Número de pessoas - *Daily Meeting*
4. Valor Agregado – Número de pessoas
5. Número de pessoas – Tempo de *Sprint*
6. Número de pessoas – Tempo de desenvolvimento
7. Número de pessoas – Quantidade de material de *Sprint*
8. Qualidade - Valor agregado
9. Valor agregado – Tempo de desenvolvimento
10. Valor agregado – Quantidade de material para *Sprint*
11. Tempo de *Sprint* – Qualidade
12. Tempo de *Sprint* – Material *Sprint*
13. Tempo de desenvolvimento – Qualidade
14. Material *Sprint* – Tempo de Desenvolvimento

Para que possamos construir o nosso modelo conceitual e elaborar a grandeza das variáveis, primeiramente devemos identificar a polaridade das relações. Então, uma a uma, vamos analisar o contexto das conexões, atribuindo ação entre as duas variáveis.

3.3. MAPEANDO AS RELAÇÕES

Vamos estabelecer qual o tipo de relação entre conceitos, a partir das relações já definidas na (Tabela 5). Para melhor compreensão, as ações encontram-se destacadas em maiúsculo.

Na relação de número (1), foi considerado que a duração da reunião diária ACRESCENTA benefícios ao valor agregado, pelo fato da comunicação entre as pessoas. Esse compartilhamento de ideias é tratado como um compartilhamento de experiências do estado atual do projeto.

Na relação de número (2), foi considerado que um projeto cujo conteúdo é amplamente discutido por seus integrantes é consequentemente desenvolvido por pessoas mais cientes da atual situação do projeto. Caso semelhante ao exemplo de número (1), assim é possível afirmar que as reuniões diárias AGREGAM qualidade ao produto final.

Na conexão de identificação (3), é possível identificar claramente a relação entre os conceitos. Pois quanto maior o número de pessoas, mais tempo irá demorar a reunião diária, esse fato é explicado devido às especificações da modelagem SCRUM (2.1.7), no ponto em que se refere às perguntas que todos os integrantes devem responder, ou seja, todos têm voz e devem falar durante a reunião diária. Mesmo não sabendo os efeitos de uma reunião mais longa, é fato em que aumentar o número de pessoas irá PROLONGAR a reunião diária.

O valor da conexão número (4) segue os princípios da relação de número (3), mas olhando isoladamente é possível verificar que um maior valor agregado, irá gerar um maior número de especializações. Logo, elevando o número de valor agregado iremos necessitar de um menor número de pessoas para a equipe. Pessoas possuem conhecimentos distintos e, conforme vimos na sessão de aprendizagem (2.3.3), a interação traz efeitos positivos do coletivo para o individual. Mas esse efeito positivo é limitado, causando uma estabilidade na relação, portanto, aumentar o valor agregado pode em longo prazo DIMINUIR o número de pessoas.

A conexão número (5) é totalmente relativa e depende muito do critério utilizado pelo analista, nessa análise foi considerado que um maior número de pessoas é capaz de concluir as tarefas de um *Sprint* mais rapidamente, assim um elevado número de pessoas REDUZ o tempo necessário para um *Sprint*. O fato que torna essa variável relativa, é que não necessariamente precisamos alterar o tempo de *Sprint*, soluções como aumentar a quantidade de material para o mesmo período seriam mais indicadas. Entretanto nesse ponto da análise, só é necessário julgar a relação entre dois conceitos sem envolver outras partes do sistema.

Para a afirmação (6), foi considerado um fato simples e intuitivo, um número maior de pessoas realiza uma atividade em menos tempo. Para o desenvolvimento de software o critério também é válido, assim um número maior de pessoas DIMINUI o tempo de desenvolvimento.

Na colocação (7) encontramos mais uma relação que fica a critério do analista, nesse exemplo foi levado em consideração que aumentar o número de pessoas torna o time capaz de assimilar mais atividades a serem realizadas em um mesmo período de tempo. Portanto, aumentar o número de pessoas torna viável AUMENTAR a quantidade de material para o *Sprint*.

Na colocação (8) usamos conceitos gerais de linhas de produção, pois um time mais bem qualificado tem capacidade de desenvolver produtos de maior

qualidade. Logo, o ato de qualificar a linha de montagem influencia no produto final, assim como produtos melhores produzidos geram experiências mais produtivas, portanto, quanto maior for a qualidade do produto produzido mais irá AUMENTAR o valor agregado obtido do software desenvolvido.

Na relação de número (9) é incontestável dizer que quanto mais experiente for um indivíduo, maior facilidade ele encontrará para realizar as suas atividades. Com base nessa afirmação, podemos dizer que quanto maior o valor agregado do coletivo, o tempo de desenvolvimento será REDUZIDO.

Na colocação (10) usaremos a mesma premissa da afirmação (9), onde afirmava que um indivíduo experiente possui maior facilidade em realizar as suas atividades. Assim, um indivíduo que possui mais facilidade em realizar tarefas, está apto a executar um número maior de tarefas do que um indivíduo inexperiente. Logo, aumentar o valor agregado ACARRETA em uma maior capacidade em comportar material para o *Sprint*.

Na conexão de identificação (11) usamos conceitos do nosso cotidiano, onde realizar uma tarefa em tempo demasiadamente acelerado pode acarretar em uma má execução da mesma. Assim, um tempo reduzido de *Sprint* pode DIMINUIR a qualidade do produto.

Na conexão (12) foi utilizado um pensamento simples, onde se inferirmos um maior tempo para realizar atividades, mais atividades conseguiremos realizar. Assim, se aumentarmos o tempo de *Sprint*, iremos AUMENTAR a capacidade de comportar uma quantidade maior de material para esse *Sprint*.

A conexão (13) possui uma afirmação já vista anteriormente, já foi esclarecido que realizar uma atividade em tempo demasiadamente acelerado, pode não conduzir a bons resultados, assim como realizar um projeto demasiadamente rápido pode resultar em um produto de má qualidade. Logo, podemos afirmar que reduzir o tempo de desenvolvimento, pode DIMINUIR a qualidade do produto a ser entregue.

A colocação de número (14) trata do fato de que quanto mais atividades realizarmos, antes alcançaremos nosso objetivo final. Assim, desenvolver uma maior quantidade de material em um período de *Sprint* vai reduzir a quantidade de *Sprints* necessários para finalizar o projeto. Fica possível afirmar que quanto maior for a quantidade de material por *Sprint,* mais iremos REDUZIR o tempo de desenvolvimento.

Com as conexões esclarecidas temos condições de criar um modelo conceitual para dar andamento à análise. Porém, antes de iniciarmos esse passo, olhemos com atenção para as colocações de número (8), (9) e (10). O conceito valor agregado causa influência de diversas maneiras diferentes, esse fato ocorre por o conceito ser complexo e assim seria de interesse conhecer a natureza desse conceito. Ou seja, entender o significado das variáveis, mesmo que isso já tenha sido feito previamente para que elas tenham sido escolhidas. O fato é que sabemos o que cada variável representa, mas precisamos saber do que cada variável é constituída. Caso uma variável seja constituída por mais de um conceito, essa pode ser um subsistema que tem seu próprio comportamento, o qual pode causar influência nos nossos resultados, caso não analisado isoladamente. Dentre as variáveis apresentadas, apenas o Valor Agregado possui uma natureza complexa, a qual deve ser definida pelo próprio analista, de acordo com o ambiente em que ele observa.

Nesse caso o valor agregado refere-se ao desenvolvimento da equipe. Então os fatores influentes vão do ambiente em que a equipe atua até as relações interpessoais dos seus integrantes. Para que nossos resultados do modelo conceitual sejam satisfatórios, precisamos entender o sistema chamado valor agregado.

3.4. VALOR AGREGADO

Para melhor compreensão do subsistema chamado valor agregado, vamos compreender como funciona um subsistema independente. Para não causar confusão, o modelo conceitual debatido nas sessões anteriores (3.2 e 3.3) será chamado de sistema principal.

Para facilitar a explicação, suponhamos que o sistema valor agregado possui apenas três variáveis. Essas variáveis formam um sistema entre si e são influentes umas com as outras, porém inferir que uma dessas variáveis causa influência direta no sistema principal seria realizar uma afirmação incompleta. Apenas o conjunto operante de todas simultaneamente tem impacto no sistema

principal, portanto o funcionamento do conceito valor agregado é um sistema com entrada e saída para o sistema principal, mas internamente tem o seu próprio mecanismo sistêmico.

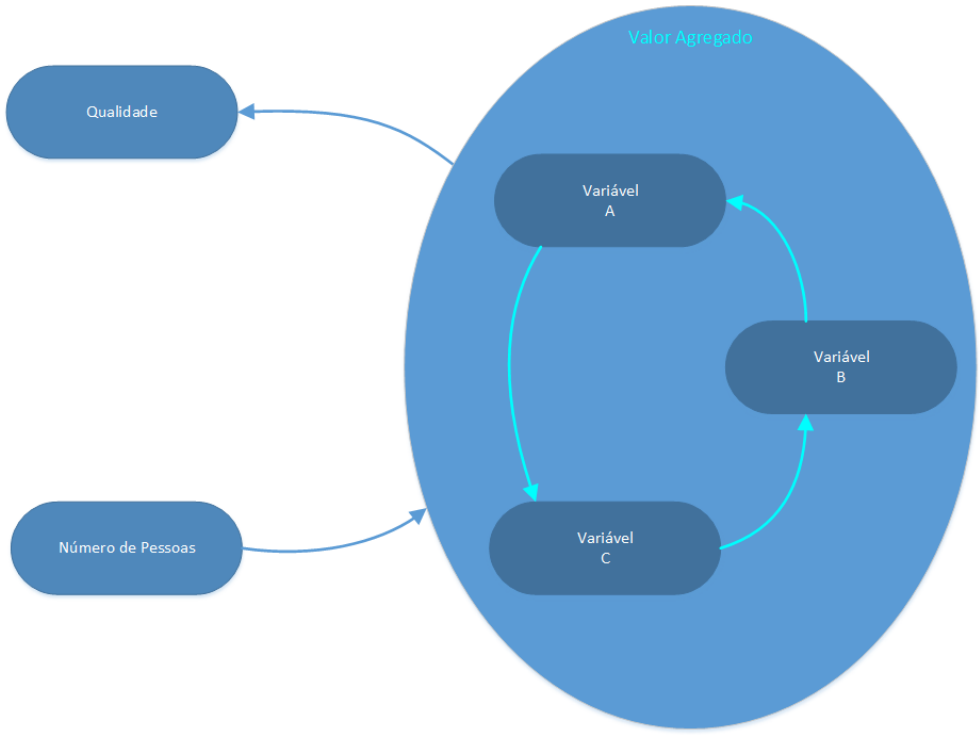


FIGURA 28: SUBSISTEMAS
FONTE: OS AUTORES (2015)

Conforme visto (Tabela 5), o valor agregado tem influência na variável Qualidade, pertencente ao sistema principal. Já no sistema principal o conceito número de pessoas é influente no conjunto chamado valor agregado. Caso queiramos entender nosso sistema por completo, vamos analisar o sistema valor agregado como sendo um sistema independente, pois seu mecanismo pode ser interessante para o nosso sistema principal.

Apesar de ser um subsistema, está sujeito a todas as técnicas apresentadas nesse trabalho. Logo, já possuímos todas as ferramentas para realizar essa nova análise, a qual deve ser iniciada com o levantamento de variáveis, as quais foram definidas como:

- Especialização
- Integração
- Entretenimento
- Autoridade

- Comunicação verbal
- Comunicação virtual direta
- Comunicação virtual indireta
- Ambiente

Para essa análise foram consideradas as relações interpessoais entre os integrantes da equipe, assim como fatores do ambiente que influenciam essas relações. A especialização refere-se ao nível de instrução médio da equipe, o qual é calculado de acordo com os níveis de instrução individuais.

Como dito na sessão de aprendizagem (2.3.3), o fator de interação é um importante ponto para a evolução do individuo, mas para que isso possa ocorrer a interação deve se fazer presente, pois a evolução individual ocasiona a evolução do coletivo. Entretenimento e autoridade são fatores; que influenciam no ambiente e esse impacto pode causar inferência nos demais fatores, o entretenimento foi levado como a liberdade em que a equipe tem para se expressar dentro do ambiente de trabalho, assim como a autoridade é referente ao rigor que o líder atua sobre o seu time, que por sua vez também reflete no ambiente. Mais uma vez recorremos à sessão de aprendizagem (2.3.4) para debatermos a comunicação, pois vimos que foi acentuada a importância da linguagem, logo, suas formas de propagação também são extremamente relevantes. Para contextualização, consideramos que comunicação verbal é a comunicação que envolve oratória direta entre dois ou mais indivíduos, a comunicação virtual direta é similar, também é instantânea, mas utilizando recursos tecnológicos como, por exemplo, um *Chat*. Já a comunicação virtual indireta é aquela comunicação que não mantém diálogo, comunicação realizada através de mensagens do tipo lembre, por exemplo, um *e-mail*.

O ambiente é um agregado de valores que envolvem a localidade, mais uma vez poderia ser mais bem detalhado ou até considerado como um subsistema, mas em virtude de não prolongar demasiadamente essa análise, vamos considerar apenas como uma variável.

Como todo sistema, o primeiro passo é identificar as influências:

Tabela 6: INFLUÊNCIAS DO VALOR AGREGADO

	Especialização	Integração	Entretenimento	Autoridade	Comunicação Verbal	Comunicação Virtual (Direta)	Comunicação Virtual (Indireta)	Ambiente
Especialização		-	-	-	-	-	-	-
Integração	+		+	-	-	-	-	-
Entretenimento	+	-		-	-	-	-	-
Autoridade	-	+	-		+	-	-	+
Comunicação Verbal	-	+	-	-		-	-	+
Comunicação Virtual (Direta)	-	+	-	-	+		+	-
Comunicação Virtual (Indireta)	-	-	-	-	-	-		-
Ambiente	-	-	+	-	-	+	-	

FONTE: OS AUTORES (2015)

As relações encontradas (Tabela 6) levaram em conta as seguintes considerações:

1. Integração – Especialização:

O aumento da integração GERA um maior nível de especialização.

2. Integração – Entretenimento:

Quanto maior a integração entre os integrantes AUMENTARÁ a descontração entre os mesmos.

3. Entretenimento – Especialização:

Quanto mais descontraído for a interação entre os integrantes, AUMENTARÃO as chances de compartilhamento de conhecimentos.

4. Autoridade – Integração:

Quanto maior for o nível de autoridade exercido pelo líder, mais irá DIMINUIR o nível de integração de sua equipe.

5. Autoridade – Comunicação Verbal:

Quando maior for o nível de autoridade exercido pelo líder, menor será a liberdade de comunicação direta, logo irá DIMINUIR o volume de comunicações verbais.

6. Autoridade – Ambiente:

Quanto maior for o nível de autoridade exercido pelas lideranças, mais tenso se torna o ambiente. Portando aumentar a autoridade gera uma REDUÇÃO no bem estar do local.

7. Comunicação Verbal – Integração:

Quanto maior for o uso desse tipo de comunicação, mais facilmente as pessoas irão interagir diretamente. Ou seja, aumentar a comunicação verbal também AUMENTA o nível de integração da equipe.

8. Comunicação Verbal – Ambiente:

Um ambiente com maior liberdade de comunicação é um ambiente mais saudável, assim, aumentando o nível de comunicação verbal iremos AUMENTAR a qualidade do ambiente.

9. Comunicação Virtual (Direta) – Integração:

Aumentar o nível de diálogo, mesmo que não seja de forma verbal, tende a favorecer em manter as relações pessoais. Em longo prazo, aumentar a comunicação virtual direta implica em AUMENTAR a integração entre os integrantes da equipe.

10. Comunicação Virtual (Direta) – Comunicação Verbal:

Aumentar o nível de comunicação virtual torna a reduzir os diálogos verbais, assim um aumento na comunicação virtual direta irá REDUZIR a quantidade de comunicações verbais.

11. Comunicação Virtual (Direta) – Comunicação Virtual (Indireta):

Quanto maior o nível de comunicação virtual, mais tendência de que os diálogos não sejam mais diretos. Assim com o passar do tempo irá AUMENTAR a comunicação virtual indireta em virtude do aumento da comunicação virtual direta.

12. Ambiente – Entretenimento:

Um ambiente saudável é um ambiente que gera bem estar nos usuários, logo um aumento na qualidade do ambiente com certeza irá AUMENTAR o entretenimento entre os integrantes do time.

13. Ambiente - Comunicação Virtual (Direta):

Um ambiente favorável a diálogos, tende a AUMENTAR o nível de comunicações virtuais diretas.

Com as relações detalhadas podemos montar um modelo o conceitual para o subsistema, assim esclarecer os componentes e o fluxo interno do valor agregado.

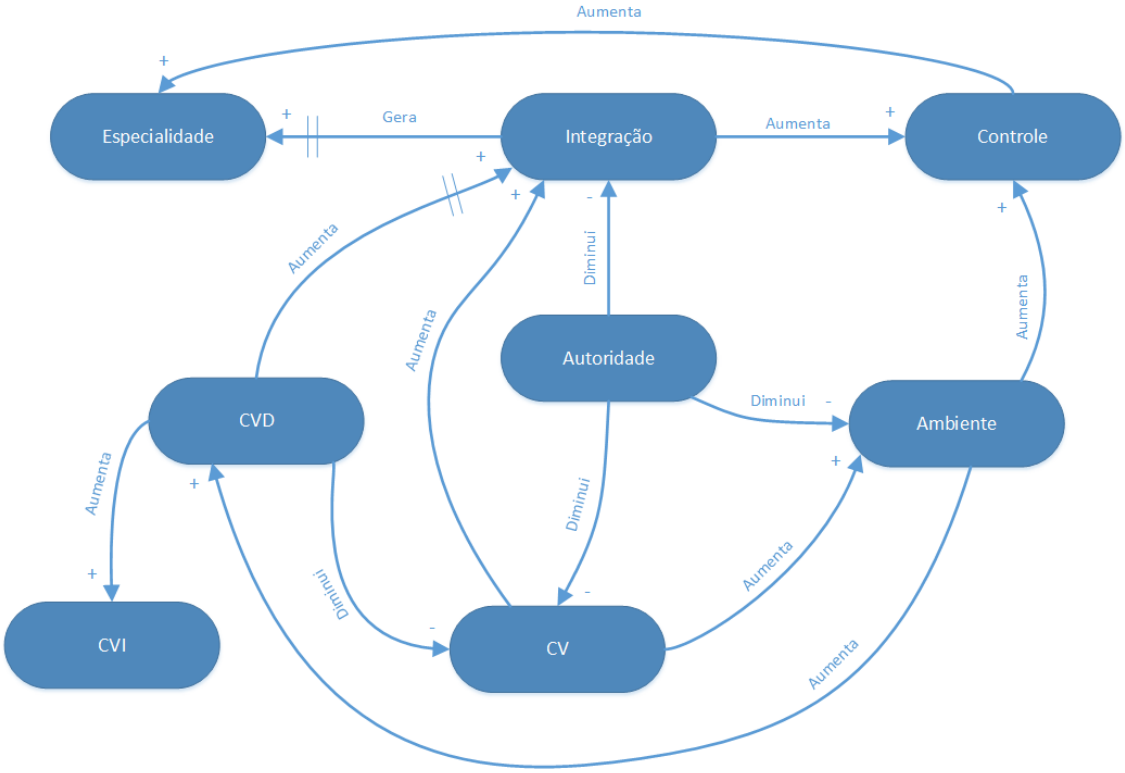


FIGURA 29: MAPA SISTÊMICO - VALOR AGREGADO
FONTE: OS AUTORES (2015)

Com o modelo conceitual definido (FIGURA 29), podemos saber os impactos e reações que o sistema provoca no subsistema e assim como também a influência do subsistema no sistema geral. Para isso podemos realizar a análise estipulada para valores do subsistema de acordo com o estado atual do sistema geral.

A partir disso podemos montar estratégias internas caso queiramos que o valor agregado ajude a contribuir para o sistema. Embora uma análise calculada em relação aos valores dos conceitos fosse necessária, preliminarmente podemos afirmar olhando para o modelo, que a autoridade é uma fonte e está diretamente ligada com qualquer que seja o resultado esperado para esse subsistema. Logo, realizar análises baseando-se nesse fator irá gerar boas estratégias de como conduzir esse subsistema.

Os valores definidos pelos conceitos reais do sistema geral irão causar influências no subsistema “valor agregado”, o que por sua vez terá reflexo e influência no sistema principal. Porém, para que possamos mapear as duas vertentes de influências entre os conjuntos, primeiramente precisaremos terminar a definição do sistema primário.

3.5. CONSTRUINDO O MODELO CONCEITUAL

Um modelo conceitual consciente segue uma sequência de diretrizes que já foram apresentadas nesse trabalho, mas como dito anteriormente, um modelo perfeito é inviável. As melhores estratégias não são fornecidas pelo modelo conceitual em si, mas sim pela análise de quem o utiliza. Em outras palavras, o modelo conceitual é uma mera ferramenta, que até o presente momento ainda tem necessidade de intervenção humana para ser operada.

Com esses fatos esclarecidos, vamos elaborar o modelo conceitual de acordo com as conexões estabelecidas na (Tabela 5), sem esquecer as definições estabelecidas na sessão de (2.3.5).

Com os dados obtidos até o momento, foi possível chegar ao seguinte modelo conceitual:

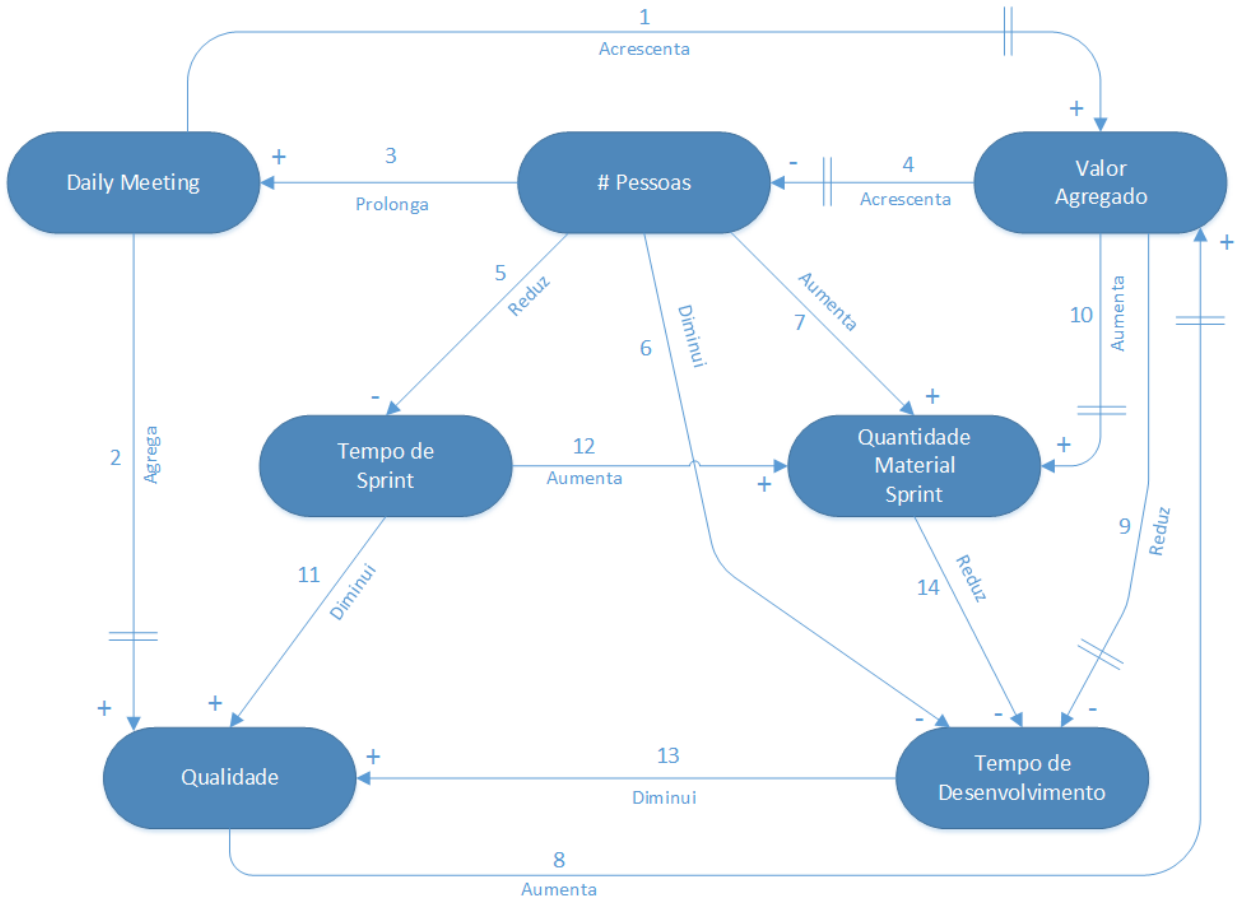


FIGURA 30: MAPA SISTÊMICO
FONTE: OS AUTORES (2015)

A (FIGURA 20) representa o modelo conceitual do sistema, sem esquecer que o campo valor agregado é um subsistema apresentado na (FIGURA 29), já visto anteriormente. Seria possível desenhar por completo o sistema utilizando o recurso de conjuntos assim como na (FIGURA 28), mas para sintetizar e facilitar as explicações a seguir, o valor agregado foi comprimido como sendo um único conceito.

Com essa ferramenta poderosa é possível definir estratégias para a atuação sobre a modelagem analisada, assim mantendo o controle sobre o sistema. Uma boa estratégia consiste em remodelar os valores reais de acordo com o fluxo da variável que está sendo manipulada. Esse fluxo não é único e também não são facilmente interpretados, logo, efeitos colaterais podem ser apresentados. Quanto pior for a interpretação do fluxo, maiores serão os danos. O passo a seguir demonstra como foi identificado um fluxo em relação a duas variáveis de valores reais.

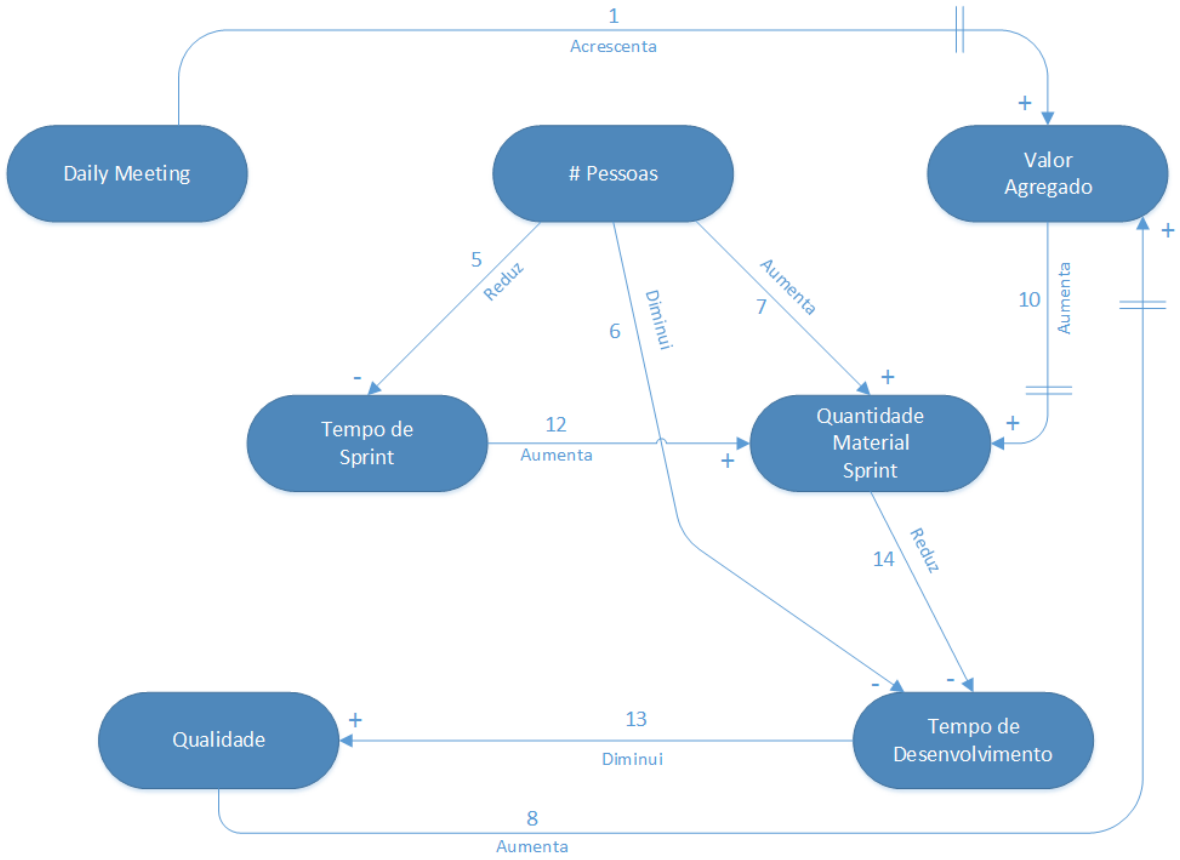


FIGURA 31: FLUXOS DO MAPA SISTÊMICO
FONTE: OS AUTORES (2015)

Para observar um fluxo, precisamos estabelecer o ponto de partida e o ponto final. É recomendável iniciar por uma variável de valor real, pois são as únicas sobre os quais possuímos poder de alteração. Para a variável de destino do nosso fluxo (FIGURA 31), determinar os conceitos resultados, ou seja, os objetivos que nossa análise pretende alcançar.

As variáveis *Daily Meeting* e Número de pessoas, por serem valores reais, foram consideradas as fontes dessa simplificação. Nota-se que é uma simplificação interpretativa, pois o modelo conceitual não deve ser simplificado. Nessa visão estamos observando somente a relação de causa das variáveis fontes.

As variáveis Qualidade, Tempo de Desenvolvimento e Valor Agregado foram consideradas nossos pontos de destino, pois toda nossa análise está voltada para a melhoria do sistema em função desses conceitos.

Nota-se que as relações de *Feedback* não estão presentes para as variáveis de partida, pois nosso intuito é descobrir que tipo de impacto essas variáveis têm no sistema, menosprezando o impacto que o sistema tem sobre essas variáveis.

Para realizar um fluxo linear de uma variável, devemos eliminar as conexões necessárias para que ela deixe de fazer referência circular. Por exemplo, na (FIGURA 31), se tivermos início em *Daily Meeting*, por nenhum caminho do grafo é possível retornar à própria variável de partida *Daily Meeting*. O mesmo acontece para a variável “Número de Pessoas”.

A variável *Daily Meeting* tem um fluxo de fácil interpretação, pois é linear e sem muitas ramificações. Assim, a partir das alterações nessa variável, o sistema sentirá impacto primeiramente em valor agregado e o impacto se propagará para o sistema de acordo com as relações do próprio valor agregado. Já na variável número de pessoas, existem diversas arestas pra iniciarmos o fluxo, assim dando a possibilidade de diversos fluxos distintos. Porém, é de interesse identificar um único fluxo que represente todos os demais. Mesmo no caso de haver mais de um com essa propriedade, vamos focar somente no que for considerado o mais relevante. Ao eliminarmos algumas arestas conforme a lógica que será explicada a seguir, encontramos o seguinte fluxo:

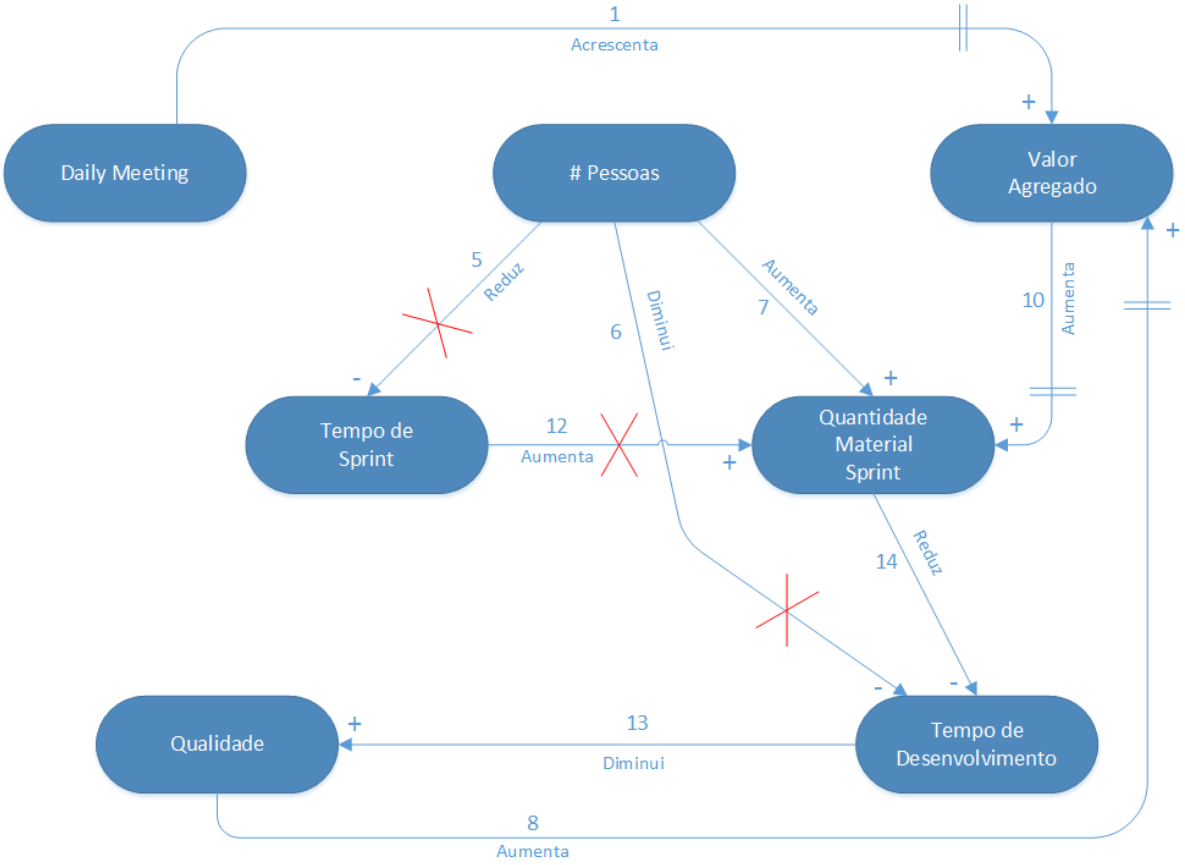


FIGURA 32: SIMPLIFICAÇÃO DOS FLUXOS DO MAPA SISTÊMICO
FONTE: OS AUTORES (2015)

As arestas de número (5) e (12) fazem a conexão entre os conceitos Número de Pessoas e Quantidade de Material para *Sprint*, utilizando o conceito Tempo de *Sprint* como fator mediador dessa conexão. Porém nota-se que a aresta identificada pelo número (7) realiza essa mesma função sem necessidade de um intermediador. Logo, a variável Tempo de *Sprint* pode ser descartada na identificação desses fluxos.

Porém é utilizada a lógica inversamente ao eliminar a aresta de número (6), pois intuitivamente se seguíssemos os mesmos princípios deveríamos eliminar as arestas de número (7) e (14). Entretanto ao eliminar a aresta de número (14), estamos comprometendo o fluxo inicial realizado para *Daily Meeting*. Como estamos analisando ambas as fontes, o conceito Quantidade de Material para *Sprint* deve ser mantido. Logo, o conjunto de arestas menos relevante torna-se a aresta de número (6).

A seguir temos um exemplo (FIGURA 33) de como foi estruturado o modelo mental para a compreensão do fluxo de valores, lembrando novamente que essa não é uma simplificação do modelo conceitual.

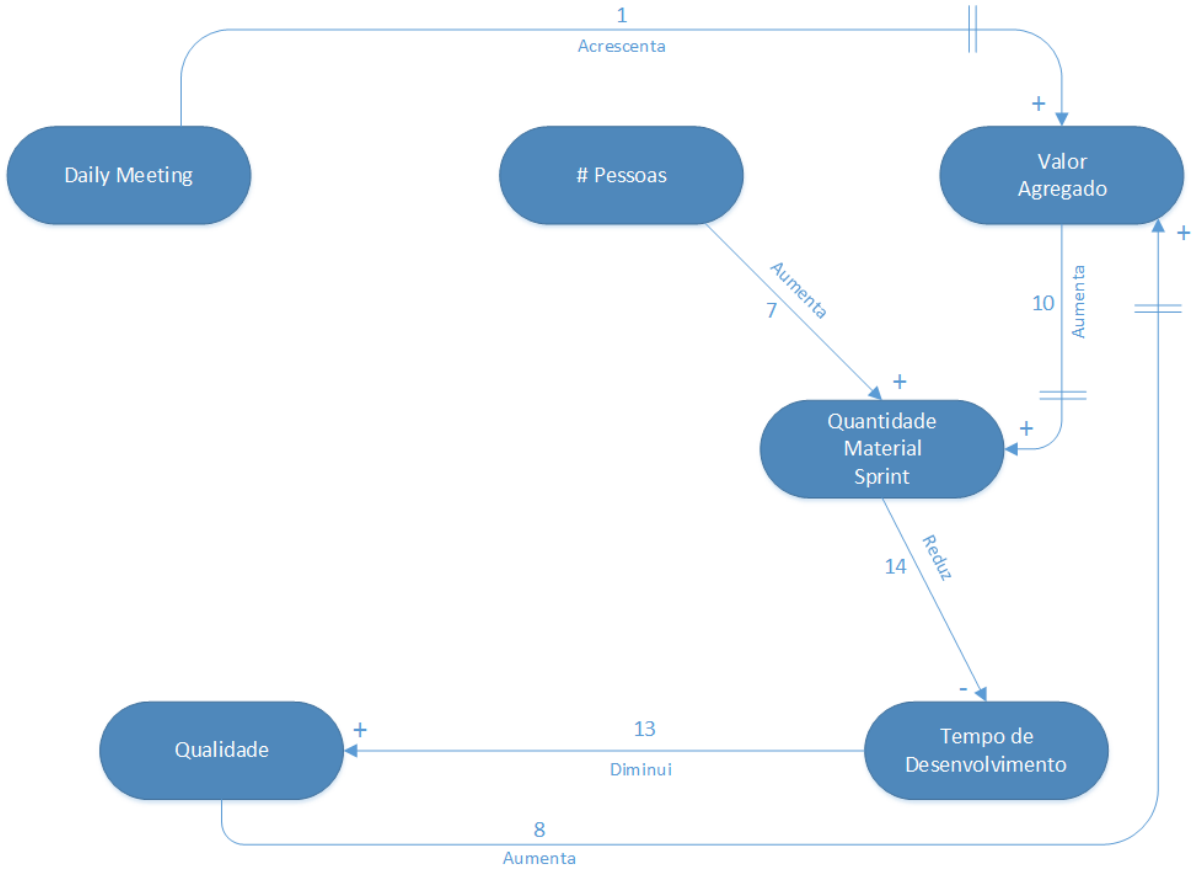


FIGURA 33: FLUXOS PRINCIPAIS DO MAPA SISTÊMICO
FONTE: OS AUTORES (2015)

Desse fluxo é possível retirar algumas estratégias de atuação como, por exemplo, se for de interesse somente aumentar o valor agregado com o mínimo de impacto para o sistema, perceba-se que manipulando o valor da *Daily Meeting* chegaríamos a esse resultado sem possuir intermédios. Caso queiramos melhorar a nossa qualidade, também podemos agir através da variável *Daily Meeting*, pois o conceito Valor Agregado irá propagar a mudança para o sistema, contudo essa alteração embora seja pouco impactante, ela só ocorre em longo prazo. Assim, se a medida for emergencial, é aconselhado procurar outra estratégia, como por exemplo, partir do Número de Pessoas, a qual tem uma sequência de eventos de curto prazo que alteram a qualidade.

Sabemos como o fluxo irá funcionar, mas para analisarmos a intensidade desse fluxo será necessário recorrer a dados concretos, por isso um levantamento de dados, conforme estipulado por esse trabalho, será necessário.

3.6. LEVANTANDO OS DADOS DO MODELO CONCEITUAL

Conforme dito anteriormente, começamos com um novo ciclo *SCRUM* com atraso de entrega (observado no ciclo anterior). A partir do mapa sistêmico (FIGURA 30) e o fluxo de dados estabelecido (FIGURA 33), vamos tomar uma estratégia de mudança para corrigir as deficiências do sistema. Como a instância do nosso problema requeria uma ação rápida para reduzir o Tempo de Desenvolvimento, buscando em paralelo o Valor Agregado em longo prazo, a estratégia a ser seguida tem que ter propriedades influentes nas causas do problema.

Uma das estratégias sugeridas é adequar o Número de Pessoas a um modo que altera o Tempo de Desenvolvimento em curtos períodos de tempo, mas como não queremos gastar recursos desnecessariamente, devemos elaborar um número de maneira cautelosa e gradativa, pois inserir números altamente discrepantes de imediato poderá resolver o problema mais rapidamente, porém torna irreversível analisar o número exato de pessoas necessárias. Por isso vamos adotar mudanças pequenas e gradativas e observar o comportamento do sistema.

O panorama inicial foi observado em um caso real, partindo do cenário:

Tabela 7: PANORAMA INICIAL DA ORGANIZAÇÃO

Daily Meeting	#Pessoas	Valor Agregado	Tempo de Sprint	Qualidade	Tempo de desenvolvimento	Quantidade de Material Sprint
15	8	?	2 semanas	?	20 semanas	7%

FONTE: OS AUTORES (2015)

O ciclo de desenvolvimento com o atual estado (Tabela 7) está fadado ao fracasso, pois sem levar em conta a deterioração do sistema, o *Sprint* de duas semanas resulta em 7% de evolução no projeto; como faltam 20 semanas para a entrega do mesmo e caso o ritmo de desenvolvimento seja mantido, ao final do prazo teremos percorrido 10 *Sprints*, resultando em 70% do projeto. Uma margem

de conclusão baixa, o que permite afirmar que o projeto irá ser finalizado com grande atraso.

Precisamos de uma solução referente a ganho de tempo, logo, uma solução imediata é requerida. Como vimos anteriormente, o fluxo referente ao número de pessoas (FIGURA 33) está altamente ligado ao tempo de desenvolvimento e pode ser uma boa alternativa, desde que seja devidamente acompanhada. Sendo assim foi aplicada uma análise em função da variável número de pessoas durante o ciclo SCRUM para esse produto, todos os dados a partir desse ponto são simulações computacionais utilizando funções:

Tabela 8: CICLO SCRUM COMPLETO EM RELAÇÃO AO NÚMERO DE PESSOAS

ID	Daily Meeting	#Pessoas	Valor Agregado	Tempo de Sprint	Qualidade	Tempo de desenvolvimento	Quantidade de Material Sprint
1	15	8	5	2 semanas	5	20 semanas	7%
2	15	8	5	2 semanas	5	18 semanas	7%
3	15	9	5	2 semanas	5	16 semanas	8%
4	15	9	5	2 semanas	5	14 semanas	9%
5	15	11	5	2 semanas	4	12 semanas	10%
6	15	11	5	2 semanas	4	10 semanas	10%
7	15	12	5	2 semanas	3	8 semanas	12%
8	15	12	5	2 semanas	3	6 semanas	12%
9	15	13	5	2 semanas	3	4 semanas	12%
10	15	14	6	2 semanas	3	2 semanas	12%

FONTE: OS AUTORES (2015)

Fica perceptível ver que o número de pessoas foi sendo aplicado conforme demanda. Foi comprovado que o problema foi solucionado através da alocação de mais pessoas para o projeto. Porém sistemicamente com a redução do tempo de desenvolvimento, também tivemos como consequência a perda na qualidade do produto. É evidente que o sistema SCRUM empregado precisa ser alterado, ou até substituído, mas definir o valor exato das variáveis para não haver perda de recursos é um passo que ainda necessita ser analisado.

Partimos de uma análise baseada no número de pessoas, onde os melhores resultados em fator de tempo de desenvolvimento foram obtidos nas linhas de identificação de número (7) e (8) da (Tabela 8). Porém foi nas linhas de número (5) e (6) que identificamos um valor suficiente para o tempo de desenvolvimento com menos avarias na qualidade do produto. Ou seja, agora temos definidos nosso chão

e teto de atuação; encontrar um valor médio satisfatório é a nossa próxima tarefa. Com base nessas informações, um novo ciclo de SCRUM se inicia com o seguinte panorama:

Tabela 9: PANORAMA APÓS UM CICLO COM ALTERAÇÕES

Daily Meeting	#Pessoas	Valor Agregado	Tempo de Sprint	Qualidade	Tempo de desenvolvimento	Quantidade de Material Sprint
15	12	6	2 semanas	3	20 semanas	12%

FONTE: OS AUTORES (2015)

Perceba-se que o ciclo se inicia com o Número de Pessoas em (12) na (Tabela 9), considerado valor que trouxe benefícios para o sistema. Porém os valores de Valor Agregado e Qualidade são os valores obtidos no ultimo *Sprint* do projeto recém-finalizado. Isso foi disposto dessa maneira para que fosse possível analisar a evolução da segunda estratégia, a qual vai consistir manipular nossa outra variável de valor real, o tempo de duração da reunião diária. O valor de número de pessoas será mantido como fixo nesse ciclo com a finalidade de estudarmos isoladamente o impacto que a Daily Meeting tem sobre o sistema. Essa análise pode ser visualizada na (Tabela 10):

Tabela 10: CICLO SCRUM COMPLETO EM RELAÇÃO AO TEMPO DE REUNIÃO DIÁRIA

ID	Daily Meeting	#Pessoas	Valor Agregado	Tempo de Sprint	Qualidade	Tempo de desenvolvimento	Quantidade de Material Sprint
1	15	12	6	2 semanas	3	20 semanas	12%
2	30	12	6	2 semanas	3	18 semanas	10%
3	30	12	6	2 semanas	3	16 semanas	7%
4	22	12	6	2 semanas	3	14 semanas	10%
5	18	12	6	2 semanas	3	12 semanas	11%
6	18	12	6	2 semanas	4	10 semanas	11%
7	19	12	6	2 semanas	4	8 semanas	10%
8	19	12	7	2 semanas	4	6 semanas	10%
9	20	12	7	2 semanas	4	4 semanas	9%
10	19	12	7	2 semanas	4	2 semanas	10%

FONTE: OS AUTORES (2015)

Note-se que a estratégia de alteração de valores foi diferente nessa segunda análise, pois como o tempo é um valor menos custoso de ser alterado, nesse caso, assim foi aplicado uma atribuição à duração das reuniões no formato de divisão de intervalos. Onde os limites iniciais eram 15 e 30, após interações não satisfatórias foi

atribuído um valor médio estimado de acordo com as análises dos limites, o processo foi replicado até se encontrar um valor adequado. É possível observar que foi possível manter e aumentar o valor agregado mesmo em um número de pessoas que anteriormente não condizia com tais resultados, assim como uma evolução na qualidade do produto. Os melhores resultados para a variável *Daily Meeting* foram observados nas linhas de identificação de número (6), (7) e (8) da (Tabela 10). As quais permitem definir estratégias para objetivos distintos; caso seja de interesse manter um tempo de desenvolvimento com folgas no prazo, é aconselhado manter reuniões em 18 min, pois foram os valores em que se observaram grandes evoluções no sistema, conseguindo manter uma quantidade de material desenvolvido em 11%.

Caso seja de interesse evoluir a infraestrutura do sistema através do time, é aconselhado reuniões de 19 minutos, pois entre as linhas de número (6) e (7) da (Tabela 10), foi possível observar uma evolução no valor agregado mantendo-se a quantidade de material desenvolvido no período em valores suficientes. Vale notar que realizar alterações conforme demanda ficou mais fácil, pois caso necessitemos diminuir o tempo de desenvolvimento do projeto sem realizar grandes alterações no sistema, sabemos que é possível alcançar esse objetivo, a partir de agora, somente reduzindo em alguns minutos a reunião diária.

Assim como verificamos que um número de pessoas igual a 11, é suficiente para atingir as metas. Assim se for necessário diminuir os recursos, é viável reduzir o número de pessoas em uma unidade e manejar novamente o tempo das reuniões diárias.

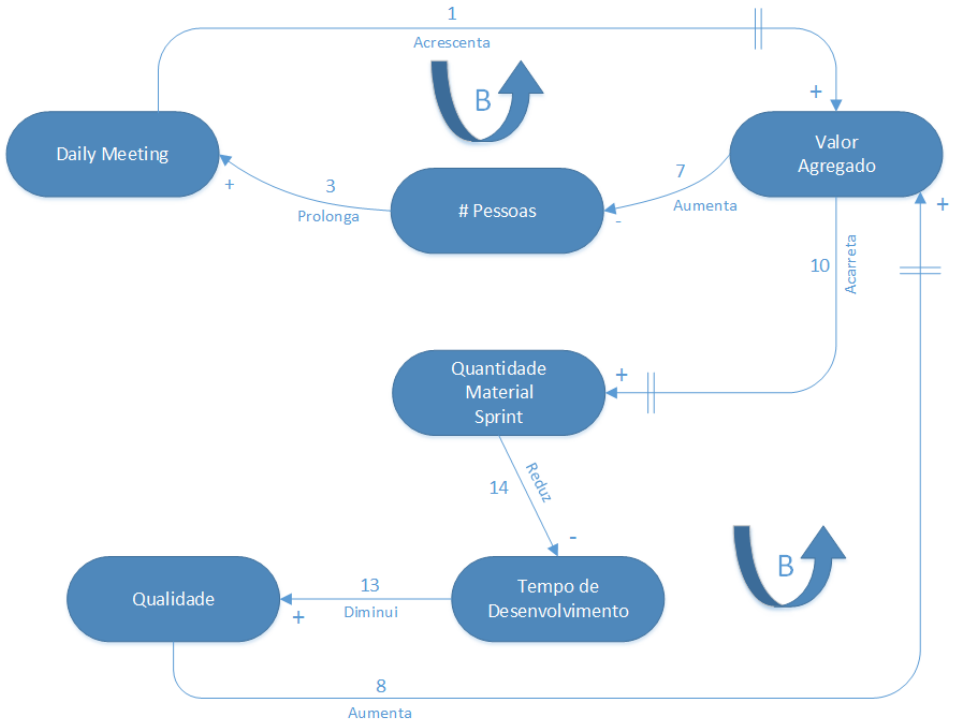
3.7. RESULTADOS DA ANÁLISE

Os fatores problemáticos do sistema foram aparentemente resolvidos, seguimos estratégias corretivas e evolutivas. Mas o sistema tende a estar em constante mudança, assim é possível que sem nenhuma intervenção o sistema que atualmente é adequado passe a se tornar ruim. Para que isso não ocorra, novas análises devem ser efetuadas constantemente, para verificar para qual vertente o sistema está se encaminhando. Algo que já foi discutido nesse trabalho é que não é

possível construir um sistema a partir de um padrão, mas é sim possível identificar padrões em um sistema já construído. Realizar esse tipo de estudo pode ajudar, a saber, segundo qual caminho que o sistema está mudando.

Em nosso exemplo, se olharmos com atenção o modelo conceitual da (FIGURA 30), podemos ver um conceito central cercado por dois enlaces de balanço. Conforme vimos na sessão de dinâmica de sistemas, esse padrão é comum e referente à vantagem relativa, descrita no padrão “Escalada” na (FIGURA 21). Com um padrão de comportamento identificado, sabemos que nosso sistema vai ter um crescimento agressivo, ou seja, irá apresentar resultados positivos já nos primeiros ciclos e continuará a evoluir por certo período de tempo. Porém esse padrão é desgastante e se não for acompanhado através de novas análises pode entrar em declínio repentino. Os enlaces de nosso modelo conceitual que apresentam um comportamento de Escalada podem ser vistos na seguinte imagem:

FIGURA 34: ARQUÉTIPO DO MAPA SISTÊMICO



FONTE: OS AUTORES (2015)

As melhorias encontradas só foram possíveis devido a análises efetuadas, mas esse processo de análise também é sujeito a um padrão de funcionamento. Sendo assim, caso o analista queira manter seu processo de análise eficiente é necessário que elas sejam feitas rigorosamente a cada ciclo.

4. CONSIDERAÇÕES FINAIS

Nesse capítulo será explicado como utilizar as técnicas apresentadas até o momento em uma organização. Também será esclarecido o real valor da análise efetuada e do estudo apresentado, assim como sugestões para futuros estudos com base nesse trabalho.

4.1. OPDCA

O modelo SCRUM analisado é relativo ao projeto em que é utilizado, assim como o projeto também deve ser parte integrante em uma gestão de projetos. Como vimos anteriormente o método PDCA é eficiente para o desenvolvimento desse produto.

Incorporar as técnicas aqui apresentadas em um método existente é impossível, pois os métodos conhecidos não foram desenvolvidos levando em conta as propostas aqui sugeridas.

Porém é possível adaptar qualquer método para acoplar o estudo desse trabalho, assim a seguir é sugerido um método PDCA diferenciado. O método não é uma criação desse trabalho, porém ainda não possui referências sólidas e estudos definidos. O método é difundido informalmente.

O método a ser proposto é conhecido como OPDCA, visto como uma adaptação do PDCA que visa à melhoria da linha de desenvolvimento se baseando em criar uma fase de observação para as experiências passadas, essa fase pode ser alocada como uma nova fase, ou uma divisão dentro do templo de planejamento. Um esboço baseando-se na (FIGURA 4) pode ser observado na seguinte imagem (FIGURA 35):

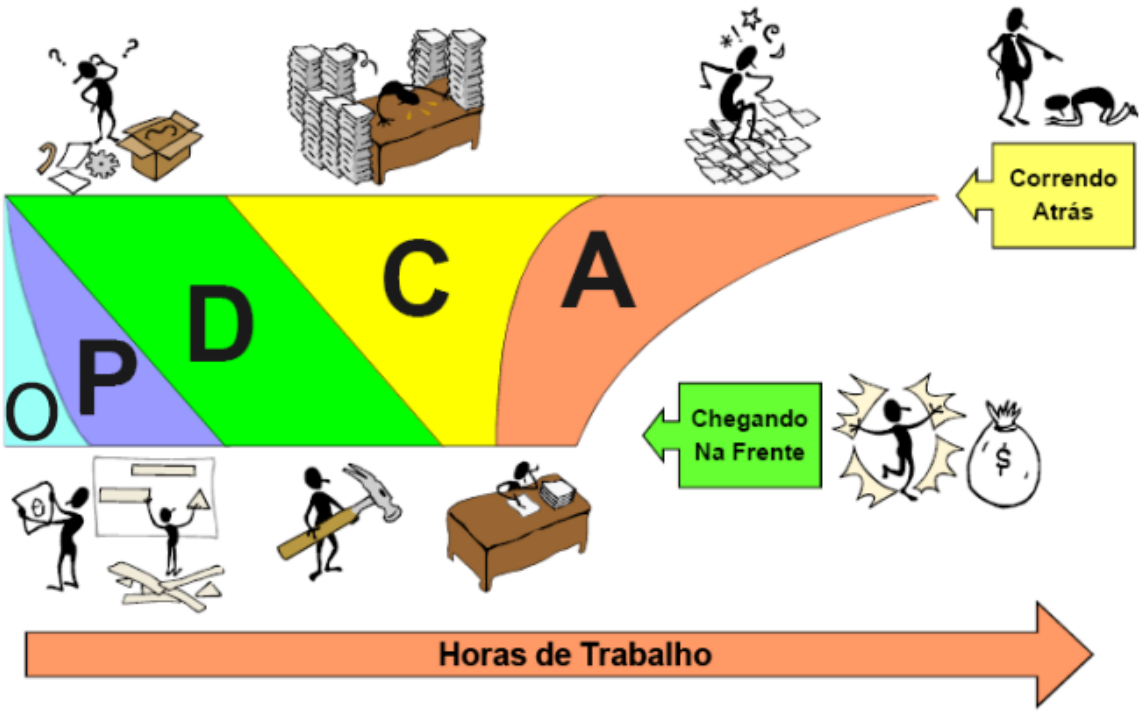


FIGURA 35: OPDCA
FONTE: OS AUTORES (2015)

Sendo assim, após um ciclo completo, o *Team Leader* adapta o conceito PDCA com as técnicas abordadas nesse trabalho; a partir de agora, a fase de planejamento, que seria dedicada à interpretação dos *Backlogs* do SCRUM, terá seu tempo dividido com a análise de comportamento da modelagem, assim como demonstrado na (FIGURA 35)

Nessa fase, denominada como *Observation*, iremos analisar sistemicamente nossa modelagem e linha de desenvolvimento, utilizando todos os artifícios discutidos nesse trabalho. Onde poderemos tirar uma única certeza desse processo de analisar informações relativas e com resultados incertos. O aprendizado será sempre um resultado certo.

4.2. Aprendizado dinâmico

O objetivo principal desse trabalho vai além de apresentar uma análise satisfatória sobre uma modelagem SCRUM. Em todas as etapas do processo

descrito nesse trabalho, não explicitamente, a análise desencadeou um processo de aprendizado em relação ao sistema observado.

Esse processo tem fundamentação nas teorias “Vigostkyanas” descritas na sessão (2.3.3). O analista a realizar um estudo como esse tem uma interação profunda com o ambiente, por naturalmente ser parte do próprio sistema. O fato de levantar as variáveis do sistema em que ele próprio participa, faz com que o analista interaja com os componentes desse sistema.

Essa interação gera uma visão mais profunda sobre o ambiente em que ele atua. A partir do momento em que as informações levantadas são projetadas em modelos mentais e descritas graficamente em modelos conceituais, o analista passa a internalizar as informações que ele mesmo descobriu.

A cada novo ciclo do projeto, o analista realiza uma nova análise e assim aprende com o histórico do projeto. Ou seja, as análises anteriores, comparadas com as novas situações, passam a ser o fator mediador desse aprendizado. Logo o analista passa a esclarecer as suas dúvidas com um elemento mais experiente, que ele mesmo gerou.

Quando esse estudo passa a ser aplicado como estratégia de melhoria da modelagem, o sistema altera o seu comportamento e o ato de se reajustar a essas novas condições força as variáveis do sistema a compartilharem as novas informações entre elas mesmas; assim, o sistema, que é constituído pelo analista que realizou as alterações e diversas outras pessoas, aprende por proximidade e acaba evoluindo o conjunto como um todo.

Por fim, não existe um grande produto gerado por esse trabalho. Apenas conceitos que, se levados em consideração, podem gerar um bem muito precioso em qualquer ambiente, o aprendizado.

4.3. Sugestões

Nesse trabalho, o estudo realizado foi efetuado sobre uma modelagem, mas apenas a título didático para facilitar a compreensão das técnicas apresentadas. Os conceitos apresentados por esse trabalho podem gerar uma gama de novos estudos para a engenharia de software, por exemplo, análises mais profundas sobre uma

modelagem em especifico, levantando métricas de qualidade para a modelagem, ou até mesmo um *checklist* para validação do emprego do modelo.

As possibilidades para estudos com base nesse material são inúmeras. A continuidade desse estudo pode ajudar para que cada vez mais a engenharia de software se fortaleça, logo disponibilizamos as técnicas aqui apresentadas para que seja base de estudo por nossa comunidade acadêmica.

REFERÊNCIAS

ABRAHAMSSON, P.; Salo, O.; Ronkainen, J.; Warsta, J. **Agile software development methods review and analysis**. Finlândia: VIT Publications, 2002.

AGILE ALLIANCE. **Princípios Por Trás Do Manifesto Ágil.** Disponível em <http://www.agilemanifesto.org/iso/ptbr/principles.html>. Acesso em: 15/05/2015.

ALVARENGA, B.; MAXIMO, A. **Curso de Física**. vol.2. Harba, São Paulo, 1986.

ALVAREZ, M. **Organização, sistemas e métodos**. V.1, São Paulo: McGraw-Hill, 1990.

BECK, K. et al. **Manifesto for Agile Software Development**, 2001. Disponível em: <http://www.agilemanifesto.org>. Acesso em: 04 jun. 2015.

BELLINGER, G. **Systems Thinking** - An Operational Perspective of the Universe, 1996. Disponível em <http://www.systems-thinking.org/systhink/systhink.htm.> Acesso em 01/07/2015.

BULSUK. **PDCA.** 2008. Disponível em: <https://en.wikipedia.org/wiki/PDCA#/media/File:PDCA_Cycle.svg>. Acesso em: 01/07/2015.

CAMPOS, V. FALCONI, TQC – **Controle da Qualidade Total** (no estilo japonês), Fundação Cristiano Otoni/Escola de Engenharia da Universidade Federal de Minas Gerais. Belo Horizonte, 1992.

CAMPOS, V. FALCONI, **Gerenciamento da Rotina do Trabalho do Dia a Dia**, 6ª ed., Belo Horizonte, Editora de Desenvolvimento Gerencial, 1994.

CAMPOS, V. F. **O verdadeiro poder**. Nova Lima: INDG Tecnologia e Serviços Ltda., 2009.

CHIAVENATO, I. **Gestão de Pessoas**: O novo papel dos recursos humanos nas organizações. 6ª tiragem. Rio de Janeiro: Campus, 1999.

COCKBURN, A.; HIGHSMITH, J. **Agile Software Development:** The People Factor, IEEE Computer, v.34, 2001.

COHEN, D.; LINDVALL, M.; COSTA, P. **Agile software development a dacs state-of-the-art report:** Technical report, Fraunhofer Center for Experimental Software Engineering. Maryland and The University of Maryland, 2003.

CONSTRUIR. **Conheça o método de gestão PDCA**, 2008. Disponível em: <http://blog.construir.arq.br/conheca-o-metodo-de-gestao-pdca/>. Acesso em: 01/07/2015.

DEMING, W. **Qualidade**: a revolução na produtividade. Rio de Janeiro, Marques Saraiva, 1990.

DEVMEDIA. Biblioteca online de tecnologia. Disponível em: <http://www.devmedia.com.br/introducao-aos-processos-de-software-e-o-modelo-incremental-e-evolucionario/29839>. Acesso em: 01/07/2015.

GOODMAN. **Judging the quality of life after surgical operations**. J. chron. Dis, 1989.

HIGHSMITH, J. **Agile Project Management**, Creating innovative products, AddisonWesley, 2004

HIGHSMITH, J. **Agile Software Development Ecosystems**. Boston: Addison Wesley, 2002.

ISHIKAWA, K. **Introduction to Quality Control**, 3A Corporation, Tokyo, 1989.

ISHIKAWA, K. **Controle de Qualidade Total**: à maneira japonesa, Editora Campos, Rio de Janeiro, 1993.

KLEINER, A.; ROTH, G. **How to make experience your company's best teacher**. *Harvard Business Review*, v. 75, n. 5, 1997.

MAXIMIANO, A.C.A. **Administração de projetos**: Transformando ideias em resultados. São Paulo: Atlas, 1997.

MOUNTAIN. **Mountain Goat Software**. Disponível em: <http://www.mountaingoatsoftware.com/scrum>. Acesso em: 22/02/2015.

NOVAK, J.; GOWIN, D. **Aprender a aprender**. 2.ed. Lisboa: Plátano Edições Técnicas, 1999.

OKADA, A.; SHUM, B.; SHERBORNE, T. **Knowledge Cartography**: software tools and mapping techniques. London: Springer-Verlag. 2008.

OLIVEIRA, D. **Sistemas, organizações e métodos**: uma abordagem gerencial. 13. ed. São Paulo, 2002.

POPPENDIECK, M. **Lean programming**. Software Development Magazine, 2001. Disponível em <HTTP://www.agilealliance.org/articles/LeanProgramming.htm>. Acesso em 23/01/2015.

PRESSMAN, ROGER S. **Engenharia de Software**. São Paulo. 6. ed. McGrawHill, 2006.

RODRIGUES e SOUTO; ANDRADE, A.; SELEME, A.. **O Desafio da Mudança Sustentada nas Organizações e na Sociedade**. Porto Alegre: Bookman, 2006.

SCHWABER, K. **Scrum Development Process**, In: OOPSLA Business Object Design and Implementation Workshop, J. Sutherland, et al., Editors. 1997.

SCHWABER, K.; BEEDLE, M. **Agile Software Development with SCRUM**. Prentice Hall, 2002.

SCHWABER, K. **Agile Project Management with Scrum**. Redmond: Microsoft Press, 2004.

SELEME. **Noções Fundamentais,** 2005. Disponível em: <http://slideplayer.com.br/slide/347872/>. Acesso em 01/07/2015.

SENGE, P. **A Quinta Disciplina**. São Paulo: Nova Cultural, 1990.

SENGE, P. **A Quinta Disciplina** – Arte, Teoria e Prática da Organização de Aprendizagem. Rio de Janeiro, 1996.

SENGE, P. **A Quinta Disciplina**: caderno de campo: estratégias e ferramentas para construir uma organização que aprende. Rio de Janeiro: Qualitymark Ed., 2000.

SOMMERVILLE, I. **Engenharia de software**. São Paulo. 6. ed. Pearson Education Companion, 2003.

VYGOTSKY, L. **A construção do pensamento e da linguagem**. Trad. P. Bezerra. São Paulo: Martins Fontes, 2001. (Original publicado em 1934).

WERKEMA. M. C. C. Ferramentas estatísticas básicas para o gerenciamento de processos. Belo Horizonte: Fundação Cristiano Ottoni, 1995.

WHITE; GUNSTONE. **How to build Concept Maps**: NASA Classroom of the Future Project, 1997.

WIND Y.; CROOK C.; GUNTHER R. **The Power of Impossible Thinking**: Transform the Business of Your Life and the Life of Your Business, 2004.