

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
CURSO DE ENGENHARIA DE SISTEMAS

MATHEUS SILVA ARAUJO

**GRAFOS DE EVOLUÇÃO DA MATURIDADE: PROPOSTA DE UM
MODELO PARA REPRESENTAÇÃO DA EVOLUÇÃO DO
CONHECIMENTO E DA CAPACIDADE**

TRABALHO DE CONCLUSÃO DE CURSO I

BELO HORIZONTE

2019

MATHEUS SILVA ARAUJO

**GRAFOS DE EVOLUÇÃO DA MATURIDADE: PROPOSTA DE UM
MODELO PARA REPRESENTAÇÃO DA EVOLUÇÃO DO
CONHECIMENTO E DA CAPACIDADE**

Trabalho de Conclusão de Curso
apresentado como requisito parcial à
obtenção do título de Engenheiro de
Sistemas, da Escola de Engenharia da
Universidade Federal de Minas Gerais.

Orientador: Profa. Dra. Ana Liddy Cenni
de Castro Magalhães

Co-orientador: Vinicius Matos Paiva

BELO HORIZONTE

2019

Dedico este trabalho aos meus avós,
pelos ensinamentos explícitos e não-
explícitos de toda uma vida.

AGRADECIMENTOS

Antes de qualquer outro agradecimento, é necessário agradecer à Ana Paula, pela paciência e compreensão durante o tempo em que estive empenhado na elaboração desse trabalho. Nosso amor é meu combustível cotidiano.

Agradeço à Professora Ana Liddy, sem a qual este trabalho não seria possível. Por toda sua dedicação enquanto coordenadora do curso de Engenharia de Sistemas e como orientadora desse trabalho e da vida. É motivo de honra sua participação aqui.

Agradeço também à DTI Digital, representada aqui pelo Vinicius, outro orientador neste trabalho e na vida; mas também a todas as outras pessoas magníficas que ali trabalham e com quem tenho a alegria de dividir algumas horas do meu dia e algumas ideias mirabolantes. O ambiente desafiador, adaptativo e complexo que ali existe é elemento fundamental neste trabalho e na minha formação.

Agradeço à minha família que permitiu que eu chegasse são e salvo até aqui.

Ao amigo Felipe, pelos diálogos que ajudaram a conduzir este trabalho.

A todos os professores e colegas de todos os cursos pelos quais passei antes de encontrar meu lugar em Engenharia de Sistemas.

Vem cá, sentar aqui do lado. Vamos pra
estrada do acaso, destino e rota tanto faz,
no caminho você escolhe o que vem
primeiro, o que vai atrás. Qual a distância
a percorrer? São quantas milhas até lá?
Que canção vamos eleger, quando
chegar perto do mar?
(ASSUNÇÃO, Leo, 2015)

RESUMO

Modelos de maturidade são ferramentas utilizadas para avaliar o nível de maturidade de um indivíduo ou organização com relação a um objetivo específico e orientar melhorias a fim de evoluir segundo aquele modelo. Neste trabalho foram levantados conceitos de gestão do conhecimento e os principais modelos de maturidade relacionados ao desenvolvimento de software. Observa-se que as formas de representação usuais desses modelos são pouco aderentes com a realidade da implantação dos modelos. Assim é proposta a representação dos modelos através da teoria de grafos, fundamentando nas diferenças de domínios baseadas nas diferenças das relações de causa e efeito do Framework Cynefin. Para validar a proposta, é proposta também a criação de uma aplicação em ambiente web que modele os grafos. Por fim, potenciais aplicações da representação e da aplicação são levantadas, como a modelagem da grade curricular de um curso de graduação.

Palavras-chave: Modelos de Maturidade e Capacidade. Grafos. Cynefin Framework. Gestão do conhecimento. Desenvolvimento Web.

ABSTRACT

Maturity models are tools used to evaluate the level of maturity of an individual or organization with respect to a specific goal and guide improvements in order to evolve according to that model. In this work, concepts of knowledge management and the main maturity models related to software development were raised. It is observed that the usual forms of representation of these models are little adherent with the reality of the implantation of the models. Thus, it is proposed the representation of the models through the theory of graphs, based on the differences of domains based on the differences of cause and effect relations of the Cynefin Framework. To validate the proposal, it is also proposed to create an application in the web environment that models the graphs. Finally, potential applications of representation and application are raised, such as modeling the curriculum of an undergraduate course.

Keywords: Maturity and Capacity Models. Graphs. Cynefin Framework. Knowledge management. Web development.

LISTA DE ILUSTRAÇÕES

Figura 1 - Valor agregado por internalização. Adaptado de (Moreira, 2005).....	17
Figura 2 - Modos de conversão do conhecimento. Adaptado de (Zambalde & Alvez, 2004)	19
Figura 3 - Modelo de Maturidade DevOps, Fonte (Mendes, 2016)	27
Figura 4 - Cynefin Framework, Fonte (Crescente, 2018)	29
Figura 5 - Grafo dirigido. Adaptado de (Cormen, 2012)	32
Figura 6 - Grafo dirigido acíclico.....	32
Figura 7 - Grafo dirigido cíclico.....	33
Figura 8 – Grafo dirigido acíclico sem ordenação topológica. Fonte (Cormen, 2012)	33
Figura 9 - Grafo dirigido acíclico após ordenação topológica. Fonte (Cormen, 2012)	34
Figura 10 - Exemplo de Rede de Fluxo. Adaptado de (Cormen, 2012).....	34
Figura 11 - Fluxo Máximo. Adaptado de (Cormen, 2012).	35
Figura 12 - Grafo RDF descrevendo Eric Miller, fonte (Manola & Miller, 2004).....	36
Figura 13 - Estrutura básica de um Grafo RDF	37
Figura 14 - Conjunto de declarações sobre o mesmo recurso, fonte (Manola & Miller, 2004).	37
Figura 15 - Grafo para o Modelo de Maturidade <i>DevOps</i>	39
Figura 16 - Funcionalidades da aplicação a ser construída	40
Figura 17 - Aplicação, Adicionar Vértice	41
Figura 18 - Aplicação, Adicionar Aresta	42
Figura 19 - Aplicação - Tela principal	43
Figura 20 - Aplicação, Editar JSON.....	43
Figura 21 - Grade Curricular do Curso de Graduação em Engenharia de Sistemas.....	46
Figura 22 - Grade Curricular do Curso de Graduação em Engenharia de Sistemas, agrupadas por período	46
Figura 23 - Critérios para o Nível de Maturidade Planejamento e Compreensão, Fonte (de Miranda, de Menezes, Ferreira, & Mendonça, 2015).....	49
Figura 24 - Projeto Music Map, Fonte (Crauwels, 2019)	50
Figura 25 - Music Map, detalhe, Fonte (Crauwels, 2019).....	50

LISTA DE TABELAS

Tabela 1 - Cálculo de fluxo máximo35

Tabela 2 - Cronograma44

LISTA DE SIGLAS

CSS	Cascading Style Sheets
ISACA	Information Systems Audit and Control Association
CMMI	Capability Maturity Model Integration
HTML	Hypertext Markup Language
RDF	Resource Description Framework
SKOS	Simple Knowledge Organization System
SPICE	Software Process Improvement and Capability Determination

SUMÁRIO

1 INTRODUÇÃO	13
2 REFERENCIAL TEÓRICO	16
2.1 CONCEITOS RELACIONADOS À GESTÃO DO CONHECIMENTO	16
2.1.1 Dado, informação e conhecimento	16
2.1.2 Gestão do Conhecimento	17
2.2 MODELOS DE MATURIDADE	19
2.2.1 Níveis de maturidade propostos por Philip Crosby	20
2.2.2 Padrões culturais de software propostos por Gerald Weinberg.....	21
2.2.3 Níveis de capacidade propostos pela norma ISO/IEC 15504 (atual série ISO/IEC 33000)	22
2.2.4 Os níveis de maturidade do CMMI.....	23
2.2.5 Modelo de Maturidade DevOps proposto por Marco Mendes.....	24
2.2.6 O modelo de representação do avanço do conhecimento SHU-HA-RI	27
2.3 CYNEFIN FRAMEWORK.....	28
2.4 TEORIA DE GRAFOS.....	31
2.5 TRABALHOS RELACIONADOS	35
3 ABORDAGEM PROPOSTA.....	39
3.1 CRONOGRAMA	43
4 POTENCIAIS APLICAÇÕES	45
4.1 GRADE CURRICULAR.....	45
4.2 SUSTENTABILIDADE E GESTÃO AMBIENTAL	46
4.3 EMPREENDEDORISMO	47
4.4 MÚSICA	49
5 CONSIDERAÇÕES FINAIS	51
REFERÊNCIAS.....	52

1 INTRODUÇÃO

Os modelos de evolução da maturidade e/ou da capacidade, muitas vezes referenciados simplesmente como modelos de maturidade, são guias destinados a avaliar processos em diversos contextos visando direcionar oportunidades de melhoria. Esses modelos orientam na determinação do nível de habilidade de uma empresa ou indivíduo na prática de um ou mais processos, de acordo com referenciais definidos pelos próprios modelos. A partir da informação disponibilizada nesses modelos, um indivíduo ou empresa pode entender melhor seu estágio atual à luz do modelo, reconhecer suas capacidades e limitações, bem como traçar planos a fim de aprimorar suas habilidades e tornar-se melhor na prática daquele conjunto de processos.

Nesses modelos, a representação da evolução da maturidade é geralmente tratada de forma linear e com pouca ou nenhuma variabilidade. As práticas relacionadas aos processos são sequenciadas em um padrão pré-formatado que é pouco preciso quando tenta representar as variâncias naturais de cada indivíduo ou organização tentando atingir um nível alto de maturidade naquele modelo.

Nesse trabalho, pretende-se utilizar uma forma menos restritiva de representação da evolução da maturidade, por meio da utilização da teoria de grafos. No modelo proposto, as relações entre as capacidades são representadas por uma rede em que a evolução pode ser percebida como uma propriedade emergente do modelo representado.

Além de uma representação mais flexível, a utilização do formalismo matemático da teoria de grafos torna possível o uso de algoritmos conhecidos para o tratamento de grafos. Acredita-se que esse uso poderá potencialmente revelar informações até então desconhecidas desses modelos de evolução da maturidade.

Para fundamentar essa proposta, foram estudados alguns dos modelos de maturidade estabelecidos na literatura (Crosby, 1979) (Weinberg, 1992) (Salviano, 2003). Nesses modelos, a maturidade pode ser vista como uma representação do conjunto de capacidades do indivíduo ou empresa. Entendendo a capacidade como a aplicação prática de conhecimento, foram também considerados conceitos relacionados à gestão do conhecimento, como a origem do conhecimento, seus tipos e modos de conversão (Moreira, 2005) e (Zambalde & Alvez, 2004).

Dentro do pensamento sistêmico, foi estudado também o Cynefin Framework (Kurtz & Snowden, 2003). Esse modelo propõe uma estrutura de classificação de sistemas em diferentes domínios: simples, complicados, complexos ou caóticos.

Neste contexto, considerando que os modelos de evolução da maturidade atuais avaliam a evolução da maturidade em um domínio simples, em que as relações de causa e consequência são repetíveis e previsíveis, este trabalho busca responder à seguinte questão: é viável um modelo de evolução da maturidade em um domínio complicado, com relações de causa e efeito não-imediatas?

O objetivo principal deste trabalho de graduação é desenvolver um método de representação dos modelos de evolução da maturidade utilizando grafos para, posteriormente, implementar uma solução computacional que faça essa representação.

Os objetivos específicos são:

1. Dominar os principais conceitos e algoritmos de teoria de grafos;
2. Analisar os principais modelos de evolução da maturidade existentes;
3. Definir os passos para a representação desses modelos por meio de grafos;
4. Criar grafos que possibilitem avaliar pelo menos um tipo de modelo de evolução da maturidade;
5. Analisar os grafos criados em relação a algoritmos pertencentes à teoria de grafos;
6. Disponibilizar a aplicação implementada para ser utilizada em diferentes contextos por potenciais pessoas interessadas.

O escopo desse trabalho engloba: a proposta de um método de representação dos modelos de evolução da maturidade por meio de grafos; a identificação de cenários que poderiam explorar a utilização desse tipo de método, visando identificar seus principais requisitos; o desenvolvimento computacional desse método; a aplicação e avaliação desse método utilizando pelo menos um dos cenários identificados.

Apesar de a identificação de cenários que poderiam explorar a utilização desse tipo de método ser parte integrante deste trabalho, não é objeto deste estudo

a criação dos vários grafos a serem aplicados a cenários distintos – englobando o levantamento das habilidades e capacidades de cada modelo e suas relações.

Este trabalho está organizado em cinco capítulos.

O Capítulo 1 contempla a introdução ao tema, definindo o contexto em que os modelos de evolução da maturidade são válidos, a motivação e os objetivos a serem atingidos com o desenvolvimento deste trabalho.

O Capítulo 2 apresenta uma breve revisão de outros trabalhos na literatura relacionados ao tema, além de conceitos e exemplos de modelos de maturidade, conceitos sobre Teoria de Grafos e uma análise do problema proposta sob a ótica do Framework Cynefin.

O Capítulo 3 detalha a abordagem proposta, descrevendo a metodologia e cronograma para o desenvolvimento do projeto

O Capítulo 4 propõe aplicações em diferentes contextos econômicos, culturais, sociais e ambientais dos grafos de evolução do conhecimento.

O Capítulo 5 encerra o trabalho com algumas considerações finais.

2 REFERENCIAL TEÓRICO

Inicialmente, uma revisão bibliográfica foi realizada a fim de compreender os principais conhecimentos envolvidos no contexto deste trabalho, entre eles: conceitos e formas de se realizar a gestão do conhecimento (Moreira, 2005) e (Zambalde & Alvez, 2004); conceitos básicos sobre a evolução da maturidade, (Crosby, 1979), (Weinberg, 1992) e (Salviano, 2003); conceitos e distinção entre sistemas simples, complicados, complexos e caóticos por meio do Cynefin Framework (Kurtz & Snowden, 2003); estudos e algoritmos relacionados à teoria de grafos (Cormen, 2012) e (Ziviani, 2011); tecnologias disponíveis para o desenvolvimento da aplicação, englobando bibliotecas de desenvolvimento web de grafos.

2.1 CONCEITOS RELACIONADOS À GESTÃO DO CONHECIMENTO

Diversos modelos de evolução da maturidade encontrados na literatura definem a maturidade de uma empresa ou indivíduo a partir da análise do conjunto de suas capacidades (Koehlegger, Maier, & Thalmann, 2009). Tais capacidades podem ser entendidas como decorrentes do conhecimento adquirido por essa empresa ou indivíduo. Em função disso, é necessário compreender a origem e a formação do conhecimento. Os grafos de evolução do conhecimento a serem propostos tentarão representar esse avanço dentro de um processo por meio de uma rede estruturada.

2.1.1 Dado, informação e conhecimento

A definição de conhecimento pode ser obtida a partir das definições de dado e informação (Moreira, 2005).

O conceito de “dado” é consensual: os dados são entidades “dadas”, que estão disponíveis no ambiente, podem ser quantificados e possuir significado. Os dados independem da ação humana, estão presentes em todo o espaço e tempo.

O conceito de “informação” não é consensual. Existem mais de 400 definições para esse termo (Moreira, 2005). Etimologicamente, o termo vem do latim “*informatio*”, que quer dizer “em forma”. A informação pode ter as seguintes formas:

- Informação como dados contextualizados;
- Informação como mensagem comunicada, ou informação como processo;
- Informação como conteúdo comunicado, ou informação como entidade subjetiva;
- Informação como objeto, ou informação como entidade objetiva.

Assim como a definição de informação, o conceito de conhecimento não é consensual. Para este trabalho, o conhecimento pode ser compreendido como a relação de sentido entre a informação e sua aplicação prática (Moreira, 2005).

É importante observar o ganho de valor à medida que o dado é internalizado e transformado em informação e depois em conhecimento, como mostrado na Figura 1.

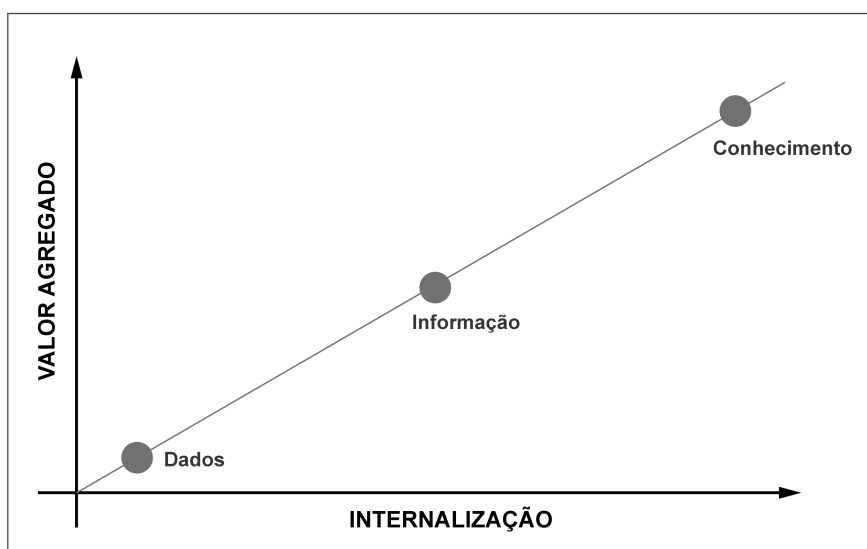


Figura 1 - Valor agregado por internalização. Adaptado de (Moreira, 2005)

2.1.2 Gestão do Conhecimento

A gestão do conhecimento pode ser compreendida como o processo de geração, codificação e transferência de conhecimento (Zambalde & Alvez, 2004).

Segundo Nonaka & Takeuchi, 1995 conhecimento pode ser classificado em duas categorias:

- Conhecimento explícito: aquele que pode ser transmitido e processado por meio de linguagem formal;
- Conhecimento tácito, ou implícito: aquele transmitido por meio do exemplo ou da convivência.

Ainda segundo Nonaka & Takeuchi, 1995 a conversão do conhecimento se dá por quatro diferentes formas, conforme apresentado na Figura 2:

- Socialização – criação do conhecimento tácito por meio do compartilhamento de experiências, como treinamentos, reuniões e interações;
- Externalização – comunicação do conhecimento tácito por meio de analogias, metáforas ou modelos;
- Combinação – troca de informações explícitas, por meio de canais digitais ou analógicos, sendo a base da educação formal;
- Internalização – aprendizado por meio da vivência prática, *learning by doing*.

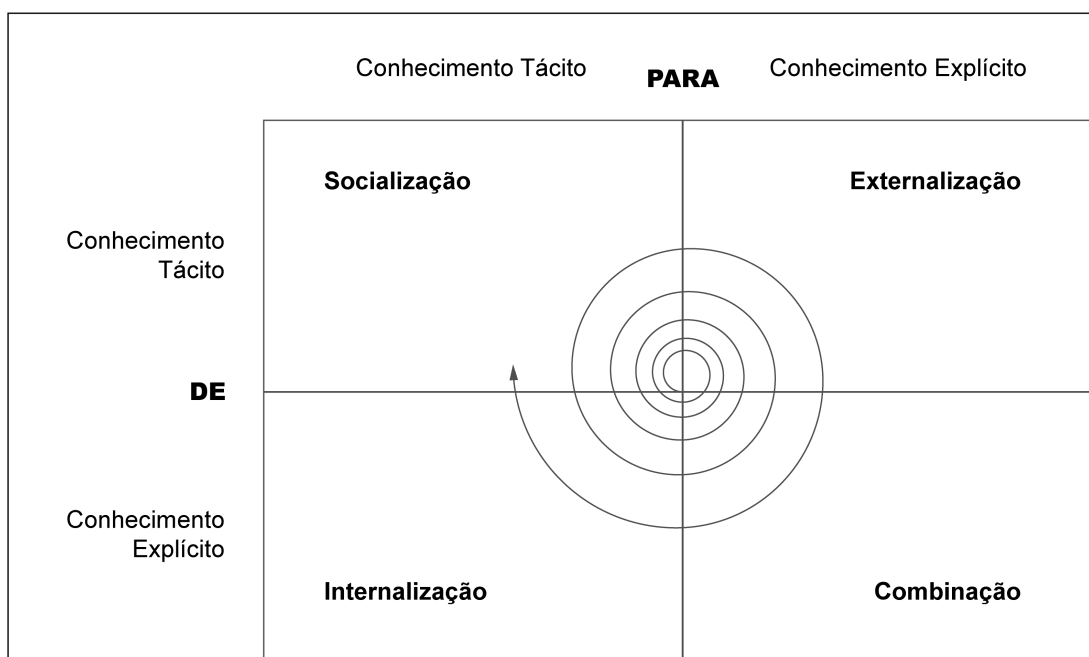


Figura 2 - Modos de conversão do conhecimento. Adaptado de (Zambalde & Alvez, 2004)

2.2 MODELOS DE MATURIDADE

Entendendo a capacidade como o conhecimento produzido por informações adquiridas e transformadas, e essas como produto de dados contextualizados e dotados de significado, a maturidade pode ser compreendida como a combinação de conhecimentos e capacidades em torno de um propósito (Koehlegger, Maier, & Thalmann, 2009).

Modelos de maturidade são, portanto, instrumentos utilizados para classificar as capacidades do indivíduo ou organização e definir ações para aumentar o nível de maturidade.

Segundo (Koehlegger, Maier, & Thalmann, 2009), à época foram identificados 74 modelos de maturidade relacionados a sistema de informação ou ciência da computação. Ainda assim, esse número representava apenas uma parcela do total de modelos, já que áreas como biologia, sociologia e psicologia também fazem uso dessa ferramenta.

Para este trabalho, serão analisados alguns modelos relacionados ao desenvolvimento de software, devido à familiaridade do autor com essa área. No entanto, as características estruturais desses são extensíveis aos demais modelos.

Os modelos de maturidade de software estão relacionados ao modelo inicialmente proposto por Crosby (1979) no contexto de qualidade em manufatura. Nesse modelo, o nível de maturidade pode ser medido à partir da adequação a um padrão de qualidade estabelecido.

2.2.1 Níveis de maturidade propostos por Philip Crosby

A ideia de maturidade relacionada ao conhecimento surge no contexto de Qualidade em Manufatura, com os trabalhos de Crosby (1979), que analisou os custos relacionados à má qualidade de um produto. Para ele, qualidade é definida como “conformidade com os requisitos”, ou ainda “defeito zero”.

A solução proposta por ele parte do princípio de que é necessário “fazer certo desde a primeira vez”. Para tornar isso possível, as organizações devem passar por cinco estágios de maturidade:

- Incerteza – estágio de desconhecimento total das causas da não-qualidade (“Não sabemos por que temos problemas com a qualidade.”);
- Despertar – questiona-se os problemas de qualidade, mas não há uma ação efetiva no sentido de corrigi-los (“Por que sempre temos problemas com a qualidade?”);
- Esclarecimento – ações com o intuito de resolver os problemas relacionados à qualidade são tomadas (“Estamos identificando e resolvendo nossos problemas”);
- Sabedoria – os problemas de qualidade são identificados e solucionados no início da produção (“Prevenção de defeitos é uma parte rotineira da nossa operação.”);
- Certeza – a qualidade se torna parte do processo produtivo (“Sabemos por que não temos problemas com a qualidade.”).

As definições de Crosby foram cunhadas no contexto de manufatura, mas são referência para a criação de diversos modelos de maturidade, em especial na área de software. Especificamente para o desenvolvimento de software, a meta de “defeito zero” de Crosby é utópica, uma vez que é impossível garantir a ausência de erros em um software que possua alguma complexidade (Beck & Andres, 2004).

2.2.2 Padrões culturais de software propostos por Gerald Weinberg

No contexto de desenvolvimento de software, Weinberg (1992) faz uma adaptação das ideias de Crosby. Ele parte de duas premissas básicas:

- Não existem duas organizações exatamente iguais;
- Não existem duas organizações totalmente diferentes.

Com essas duas premissas, ele passa a considerar as diferenças entre duas organizações. Outra mudança em sua abordagem é a noção subjetiva de qualidade: para ele, a “qualidade é valor para alguma(s) pessoa(s)” (Weinberg, 1992). Portanto, algumas afirmações verdadeiras sobre a qualidade em software podem ser:

- Defeito zero é uma forma de se perceber a qualidade;
- Ter muitas funções é uma forma de se perceber a qualidade;
- Codificação elegante é uma forma de se perceber a qualidade;
- Alto desempenho é uma forma de se perceber a qualidade;
- Baixo custo de desenvolvimento é uma forma de se perceber a qualidade;
- Desenvolvimento rápido é uma forma de se perceber a qualidade;
- Facilidade para o usuário é uma forma de se perceber a qualidade.

Weinberg questiona também o uso da palavra “maturidade”. Para ele, “a palavra *maturidade* não é um fato, mas um julgamento” (Weinberg, 1992).

A palavra “maduro” vem do latim *maturus* e significa atingir a última fase do crescimento e desenvolvimento natural. Para Weinberg, a progressão pelos estágios de Crosby não é exatamente “natural”, mas sim dispendiosa. Assim como a qualidade pode ser subjetiva, a maturidade também deve ser. Não existem padrões culturais mais ou menos maduros, mas sim mais ou menos *adequados*.

Colocadas as ponderações sobre a visão de qualidade e maturidade de Crosby, Weinberg propõe seis padrões culturais de software (Weinberg, 1992):

- Esquecido – “Nós nem sabemos que estamos realizando um processo”;
- Variável – “Nós fazemos qualquer coisa que sentimos no momento”;
- Rotina – “Nós seguimos nossas rotinas (exceto quando em pânico)”;
- Direção – “Nós escolhemos entre nossas rotinas segundo os resultados que elas produzem”;
- Antecipação – “Nós estabelecemos rotinas baseados em nossa experiência prévia com elas”;

- Congruência – “Todos estão envolvidos na melhoria de tudo o tempo todo”.

2.2.3 Níveis de capacidade propostos pela norma ISO/IEC 15504 (atual série ISO/IEC 33000)

A norma ISO/IEC 15504 é desenvolvida desde 1993 pela ISO em conjunto com o projeto SPICE (*Software Process Improvement and Capability Determination*), tendo sido inicialmente publicada em outubro de 2003. Atualmente a norma ISO/IEC 15504 foi atualizada e transformada na série ISO/IEC 33000). Ambas estão estruturadas em duas dimensões: dimensão de processos e a dimensão de capacidade de processos.

A dimensão de processos define os processos Primários, Organizacionais e de Apoio que compõem a avaliação de uma organização.

Na dimensão de capacidade de processos, a capacidade dentro de um processo é avaliada em seis níveis crescentes, desde o nível inferior, 0, até o nível superior, 5 (Salviano, 2003):

- Nível 0 – Incompleto, no qual o processo não está implementado ou não é funcional, ou seja, não atinge seus objetivos;
- Nível 1 – Executado, no qual os objetivos do processo são alcançados de alguma maneira;
- Nível 2 – Gerenciado, no qual o processo é executado de maneira planejada e controlada;
- Nível 3 – Estabelecido, no qual o processo, além de executado e gerenciado, é instanciado a partir de um processo padrão definido;
- Nível 4 – Previsível, no qual o processo passa a ser executado dentro de limites quantitativos bem definidos, com acompanhamento estatístico;
- Nível 5 – Otimizado, no qual o processo previsível pode ser aprimorado continuamente.

As normas ISO/IEC 15504 e 33000 definem, ainda, as dimensões que devem ser avaliadas em uma organização e os critérios de avaliação para cada dimensão (Salviano, 2003).

2.2.4 Os níveis de maturidade do CMMI

O *Capability Maturity Model Integration*, mais conhecido como CMMI, é gerenciado pelo CMMI Institute, uma organização da ISACA (*Information Systems Audit and Control Association*).

Assim como a ISO/IEC 15504, ele define práticas de referência para a maturidade em áreas específicas, como Engenharia de Sistemas, Engenharia de Software, Desenvolvimento Integrado de Processo e Produto e Seleção de Fornecedores (Salviano, 2003).

O CMMI tem duas representações distintas: a representação contínua e a representação por estágios.

Na representação contínua, inspirada na ISO/IEC 15504, há os mesmos seis níveis de capacidade e cada processo pode estar em um nível diferente, dependendo do interesse ou necessidade da organização.

Já na representação por estágios, há cinco diferentes níveis pré-determinados de maturidade. A sequência dos níveis não deve ser desconsiderada, já que um nível serve como habilitador para o próximo. Em função disso, mesmo que a maioria dos processos de uma organização estejam no nível três, por exemplo, caso existam processos no nível dois, a organização ainda será considerada como estando no nível dois.

Os níveis de maturidade definidos pela CMMI são:

- Nível 1: Inicial, *Ad-hoc*, no qual os processos são imprevisíveis, pouco controlados e reativos;
- Nível 2: Gerenciado, no qual os processos são caracterizados a cada projeto, e as ações relacionadas são frequentemente reativas;
- Nível 3: Definido, no qual os processos são caracterizados para a organização, gerando instâncias por projeto, e as ações relacionadas são frequentemente proativas;
- Nível 4: Quantitativamente gerenciado, no qual os processos são medidos e controlados estatisticamente sendo, portanto, previsíveis;
- Nível 5: Em otimização, que enfatiza a melhoria contínua dos processos quantitativamente gerenciados.

2.2.5 Modelo de Maturidade DevOps proposto por Marco Mendes

No universo do desenvolvimento de software contemporâneo, o *DevOps* é a combinação de culturas, práticas e produtos a fim de garantir entregas de software mais rápidas, com menos falhas, mais assertivas e seguras em ambiente produtivo (Kim, Humble, & Debois, 2018).

A palavra *DevOps* vem da junção entre *Development* e *Operations* e representa a união entre os times que desenvolvem e os times que operam e suportam um software em produção.

Estudos estatísticos realizados por Forsgreen, Humble, & Kim, 2018 mostram que organizações com sólida aplicação de práticas *DevOps*:

- Reduzem o tempo de desenvolvimento e entrega em produção de uma funcionalidade solicitada pelo time de negócios;
- Aumentam a frequência de novas versões de um software;
- Reduzem o tempo para recuperação em caso de falhas do software;
- Reduzem o índice de falhas no desenvolvimento do software.

A implantação de uma cultura *DevOps* em uma companhia passa pela implantação de diferentes práticas nos processos de desenvolvimento e operação de um software (Mendes, 2016).

Para orientar essa implantação, Marco Mendes propõe um Modelo de Maturidade com oito práticas e cinco níveis de maturidade em cada prática.

Os cinco níveis de maturidade são:

- Maturidade 1 – Inicial
 - “Ausência da prática ou iniciativas *ad hoc* realizadas isoladamente por algumas pessoas”;
 - Entregas de software em produção em bases semestrais ou anuais são comuns;
 - Retrabalhos acima de 30%;
 - Conflitos entre desenvolvimento, qualidade e operações são comuns.
- Maturidade 2 – Consciente
 - “Prática embrionária no time ou organização. Resultados iniciais e ainda inconsistentes”;

- Busca pela redução do tempo de ciclo e retrabalhos;
- Ações iniciais em projetos pilotos, com ferramentas de automação;
- Aproximação entre desenvolvimento, qualidade e operações.
- Maturidade 3 – Gerenciado
 - “Prática sistematizada pelo time ou organização. Resultados de negócio começam a se tornar visíveis pelo corpo gestor”;
 - Práticas estão em curso em projetos importantes;
 - Resultados de negócio visíveis;
 - Retrabalhos descem para 15 a 30%;
 - Entregas em homologação em base semanal e produção em base mensal.
- Maturidade 4 – Avançado
 - “Prática otimizada pelo time e que se torna parte da cultura da organização. Resultados de negócio sempre visíveis pelo corpo gestor”;
 - Práticas estão em curso na maioria dos projetos;
 - Existe forte automação das práticas através de ferramentas de automação;
 - Novos projetos e produtos já nascem com o uso das práticas de *DevOps*.
- Maturidade 5 – Melhoria Contínua
 - “Prática em melhoria contínua. Time possui excelência na realização da prática, que foi completamente assimilada na cultura da organização”;
 - Excelência no uso das práticas;
 - Retrabalhos abaixo de 15%;
 - Entregas em homologação em base diária e produção em base semanal;
 - Times de desenvolvimento, qualidade e operações trabalham em sinergia.

As oito práticas propostas por Mendes são:

- Qualidade de código
 - Diz respeito às práticas do time relacionadas à qualidade do código fonte do software.

- Infraestrutura como Código
 - Diz respeito às práticas do time relacionadas à configuração dos ambientes do software através de código fonte.
- Gestão de Builds
 - Build, no contexto de desenvolvimento de software, é a versão compilada do código fonte. Essa prática diz respeito ao processo de compilação do código fonte utilizado pelo time.
- Gestão de Releases
 - Em desenvolvimento de software, release é o processo de liberação de um software para o usuário final. Essa prática diz respeito ao processo de entrega do software compilado em ambientes de produção.
- Gestão de Testes
 - Em software, testes podem ser automatizados através de código fonte. Essa prática diz respeito ao nível de maturidade do time quanto a testes automatizados.
- Testes de Carga, Estresse e Injeção de Falhas
 - Testes de Carga, Estresse e Injeção de Falhas são testes cujo propósito é avaliar o comportamento e performance do software em situações adversas. Essa prática diz respeito ao nível de maturidade quanto aos testes de performance.
- Gestão de Configuração
 - Diz respeito às práticas do time relacionadas à administração das alterações no software.
- Monitoração de Aplicações
 - Diz respeito às práticas utilizadas pelo time para monitorar e analisar softwares em ambiente de produção.

Para representar o avanço em todas as práticas, Mendes propõem a utilização de um gráfico do tipo radar, como mostrado na Figura 3. Nessa figura, ele mostra uma empresa XYZ que está iniciando a prática de *DevOps*, e uma outra ABC que tem alto nível de maturidade em *DevOps*.

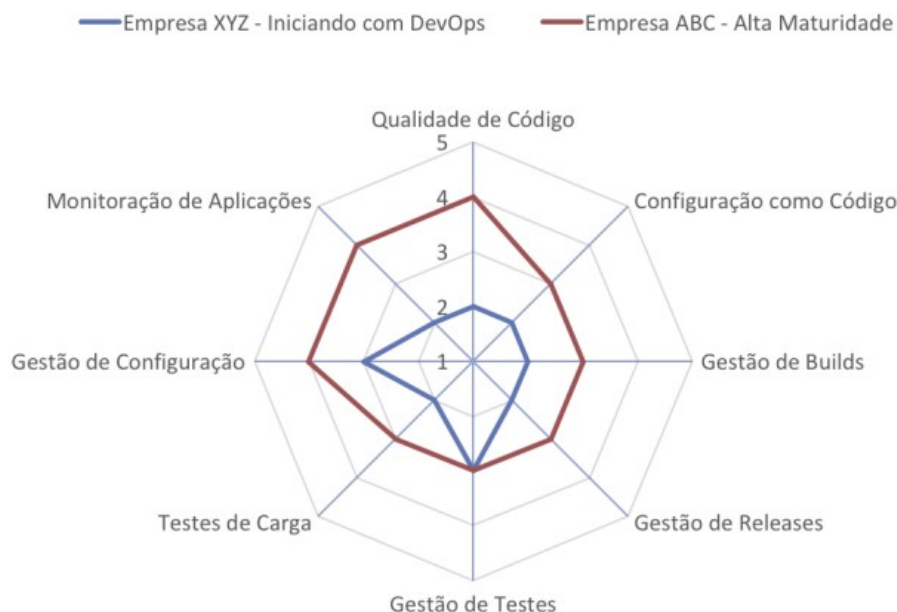


Figura 3 - Modelo de Maturidade DevOps, Fonte (Mendes, 2016)

O modelo de maturidade proposto por Mendes será utilizado como objeto de estudo para validar a proposta de representação através de grafos de evolução da maturidade deste trabalho.

2.2.6 O modelo de representação do avanço do conhecimento SHU-HA-RI

A maioria dos modelos de evolução da maturidade da literatura, incluindo os aqui apresentados, possuem uma estrutura em que os processos analisados seguem um caminho com as etapas:

1. O processo não é realizado ou acontece aleatoriamente (“Eu não faço ideia do que estou fazendo.”);
2. O processo é realizado de forma instintiva (“Estou apenas fazendo.”);
3. O processo, além de realizado, passa a ser gerenciado (“Entendo o que estou fazendo.”);
4. O processo passa a ser medido para encontrar melhores práticas (“Posso descrever o que estou fazendo.”);
5. O processo ideal é definido e passa a ser executado (“Conheço a melhor forma de fazer o que estou fazendo.”);

6. O processo ideal é executado repetidamente e passa a ser aprimorado ('Estou sempre melhorando o que estou fazendo.').

Esses seis passos podem ser comparados ao Shu-Ha-Ri nas artes marciais japonesas, especialmente o Aikido (Fowler, 2014).

Nesse modelo de representação do avanço do conhecimento, as três etapas são definidas da seguinte forma:

- Shu – estágio inicial, em que o aluno segue os ensinamentos do mestre com precisão;
- Ha – nesse estágio, o aluno passa a aprender os princípios que norteiam a técnica;
- Ri – no último estágio, o aluno já não está mais aprendendo com um mestre, ele já compreende os princípios e é capaz de criar sua própria abordagem.

É possível inferir um paralelo entre o Shu-Ha-Ri e os modelos de maturidade estudados. Nessa comparação, o Shu se relaciona com as etapas 1 e 2; o Ha se relaciona com as etapas 3 e 4, e o Ri com as etapas 5 e 6.

2.3 CYNEFIN FRAMEWORK

Enquanto trabalhavam para a IBM Global Services, Kurtz & Snowden (2003) propõem o Framework Cynefin, uma estrutura de compreensão e classificação de sistemas para auxiliar gestores e líderes na tomada de decisão dependendo de sua previsibilidade (Snowden & Boone, 2007). A palavra Cynefin é de origem galesa e pode ser interpretada como "habitat" (University of Wales, 2019) ou ainda "lugar a que se pertence".

O Framework Cynefin sugere que domínios diferentes exigem respostas diferentes, e para distinguir os domínios ele analisa as relações da causa e efeito dos eventos que neles ocorrem. Compreender como essas relações acontecem é importante para selecionar as ferramentas adequadas para se trabalhar no sistema em questão. Dessa forma, o Cynefin não "fornece" uma solução, ele "sugere" uma melhor abordagem para encontrar a solução.

Inicialmente, o Cynefin distingue os sistemas em três grupos: os ordenados, os complexos e os caóticos. Em sistemas ordenados, as relações entre causa e

efeito existem, são repetíveis e podem ser previstas. Em sistemas complexos, essas relações continuam existindo, mas não podem ser previstas. Em sistemas caóticos, não existem relações entre causa e efeito, ou elas não podem ser determinadas.

Após essa definição inicial, Kurtz & Snowden (2003) separam os sistemas ordenados em “sistemas simples” e “sistemas complicados” e criam o domínio do desordenado para sistemas que não podem se encaixar em nenhum dos domínios. Ficando portanto com cinco diferentes domínios: Simples/Óbvio, Complicado, Complexo, Caótico e Desordenado, Figura 4.

Os domínios são diferentes de categorias, uma vez que os sistemas podem estar próximos de uma ou de outra fronteira, e eventualmente podem migrar entre os diferentes domínios à medida que são mais bem compreendidos ou que seus modelos são aprimorados (Keogh, 2018).

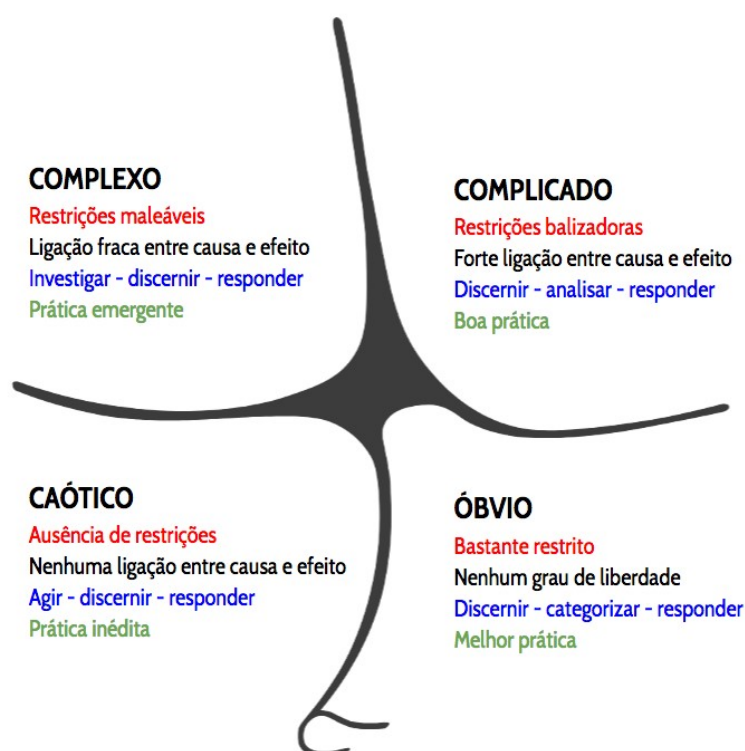


Figura 4 - Cynefin Framework, Fonte (Crescente, 2018)

No domínio do Simples/Óbvio (*Known*), estão sistemas e contextos em que as relações de causa e efeito são repetíveis e previsíveis. O processo de decisão acontece no formato: “sente, categoriza e responde”. Nesses contextos, podem ser aplicados processos padrão com ciclos de revisão e medidas claras.

Um exemplo de um sistema simples é a atualização do antivírus dos computadores em uma empresa. A equipe responsável por essa atualização já

realizou essa atividade antes e pode estabelecer um plano e um cronograma para essa execução (Ribas, 2018).

No domínio do Complicado (*Knowable*), as relações de causa e efeito estão dispersas no tempo, mas ainda assim são repetíveis e podem ser analisadas. O processo de decisão acontece no formato: “sente, analisa e responde”. Para esses contextos, técnicas analíticas e reducionistas são bons guias nas tomadas de decisões.

A construção de um dispositivo de GPS, *Global Position System*, é um exemplo de um sistema complicado. Apesar de envolver conceitos e tecnologias avançados, os requisitos e o público alvo do produto são bem conhecidos. Um grupo de engenheiros com conhecimentos específicos é capaz de construir o dispositivo com algum esforço (Ribas, 2018).

No domínio do Complexo, há muitas possibilidades, as relações de causa e efeito só mostram coerência em uma análise retrospectiva. O processo de decisão acontece no formato: “sonda, sente e responde”. Para decisão nesses contextos, são necessárias múltiplas, pequenas e distintas intervenções baseadas em gerenciamento de padrões, filtros de perspectiva e análise sistemas complexos adaptativos.

Exemplos de sistemas no domínio do Complexo são empresas *start-ups* criando produtos inexistentes ou inovadores no mercado. Nesses sistemas não é possível prescrever os resultados, as empresas precisam trabalhar com pequenos ciclos de descoberta e validação de seus serviços ou produtos no mercado, é um cenário de incerteza em que a solução vai sendo construída por meio desses pequenos ciclos.

No domínio do Caótico, os sistemas e contextos não apresentam nenhuma relação entre causa e efeito em um nível sistêmico. O processo de decisão acontece no formato: “age, sente e responde”. Nesses contextos, são possíveis apenas pequenas intervenções para estabilizar situações.

Para o domínio do Caótico, um exemplo é uma casa pegando fogo. Uma pessoa sozinha e sem condições de apagar o incêndio se preocupa apenas em sair da casa, não há tempo de planejar uma rota de fuga.

As ferramentas atualmente utilizadas para representar os sistemas de evolução da maturidade consideram as relações entre causa e efeito lineares e repetíveis, à medida que sequenciam as capacidades/conhecimentos em um

caminho linear pré-definido, não admitindo variabilidade e desconsiderando as relações emergentes. Isso define uma “melhor prática”, característica do domínio do Simples/Óbvio dentro do Framework Cynefin.

O que esse trabalho propõe é representar os sistemas de evolução da maturidade por meio dos grafos, possibilitando uma visão sistêmica mais abrangente das relações entre os elementos (capacidades/conhecimentos) desses sistemas. Dessa forma, as relações de causa e efeito menos restritivas permitindo diferentes caminhos na evolução da maturidade, definindo “boas práticas”, característica do domínio do Complicado no Framework Cynefin.

2.4 TEORIA DE GRAFOS

Para este trabalho, será utilizada a Teoria de Grafos como fundamentação matemática. A seguir, os principais conceitos utilizados neste trabalho são apresentados.

Um grafo é composto por um conjunto de vértices e por um conjunto de arestas. Um vértice pode representar um objeto simples e unitário qualquer, podendo ter nome e outros atributos. Uma aresta representa a relação entre dois vértices (Ziviani, 2011).

Conforme ilustrado na Figura 5, um grafo dirigido **G** é um par **(V,E)**, em que:

- **V** é um conjunto finito de elementos, denominado “conjunto de vértices”.
- **E** é uma relação binária entre elementos de **V**, denominado “conjunto de arestas”. As arestas em um grafo dirigido estão “direcionadas” em um único sentido de um vértice de origem para um vértice de destino¹.

¹ Grafos em que as arestas não são “direcionadas” são chamados de grafos não-dirigidos. Neste trabalho serão utilizados grafos dirigidos.

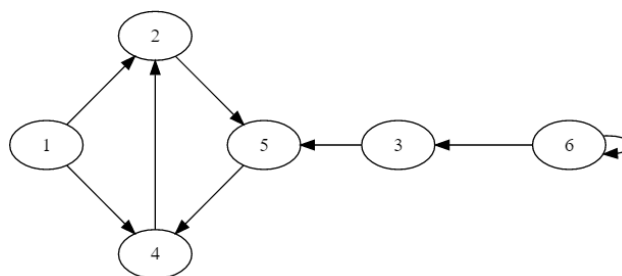


Figura 5 - Grafo dirigido. Adaptado de (Cormen, 2012)

Nos grafos de evolução do conhecimento, as capacidades serão representadas como vértices e as relações de precedência entre elas serão representadas como arestas dirigidas.

Para um grafo dirigido, existem dois tipos de graus em um vértice. O número de arestas que chegam ao vértice é definido como *in-degree*, grau de entrada, e o número de arestas que saem do vértice é definido como *out-degree*, grau de saída.

Na Figura 5, o vértice de número 5, por exemplo, tem grau de entrada 2 (chegam nele arestas dos vértices 3 e 2) e grau de saída 1 (dele sai uma aresta para o vértice 4).

Um caminho em um grafo dirigido pode ser compreendido como uma sequência de vértices que ligam um vértice inicial a um vértice final. Na Figura 5, o caminho entre o vértice 1 e o vértice 5 é o conjunto {1, 2, 5}, não há um caminho entre o nó 5 e o nó 1.

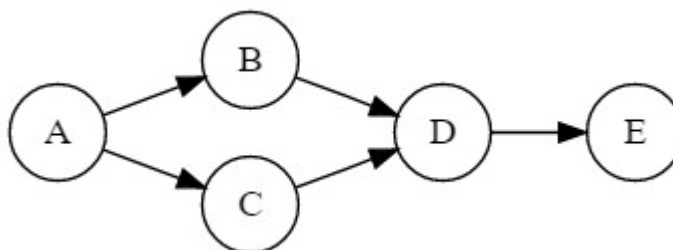


Figura 6 - Grafo dirigido acíclico

Os grafos utilizados neste trabalho serão do tipo dirigidos acíclicos, ou seja, seus caminhos não possuem vértices repetidos, não há formação de ciclos. O grafo na Figura 6 é um grafo acíclico. Na Figura 7 foi inserida uma aresta entre os vértices D e A, tornando o grafo cíclico com, no mínimo, os ciclos {A, C, D, A} e {A, B, D, A}.

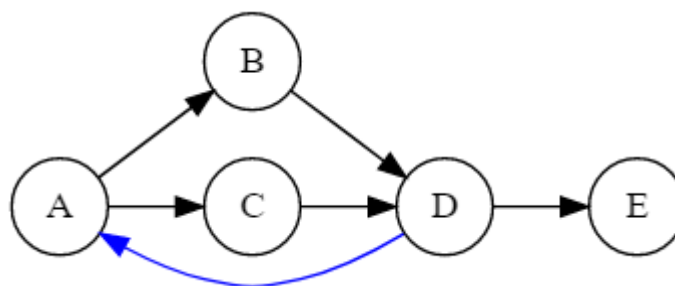


Figura 7 - Grafo dirigido cíclico

Para grafos dirigidos acíclicos o conceito de ordenação topológica representa a sequência em que, para todo vértice V , todos os vértices para os quais V tem aresta de saída aparecem depois de V .

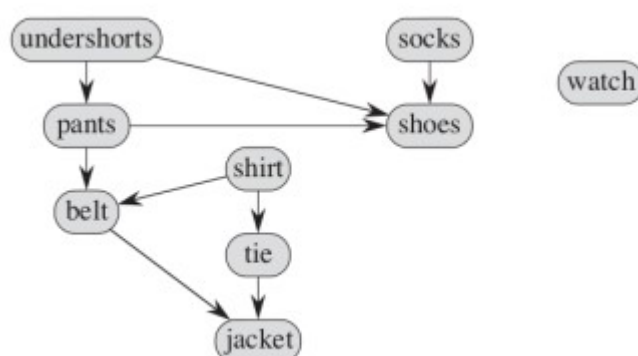


Figura 8 – Grafo dirigido acíclico sem ordenação topológica. Fonte (Cormen, 2012)

Para ilustrar este conceito, uma pessoa ao se vestir pela manhã deve vestir algumas peças da roupa antes de outras, para algumas peças a ordem não é importante. A Figura 8 mostra as relações de precedência. Na Figura 9 o mesmo grafo é apresentado após a ordenação topológica, que sugere uma ordem para vestir as peças, todas as setas estão apontando para a direita.

A ordenação topológica neste trabalho poderá ser utilizada para definir um sequenciamento possível entre as capacidades do modelo representado.

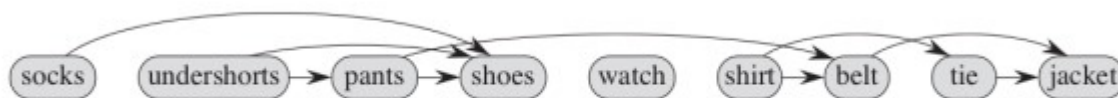


Figura 9 - Grafo dirigido acíclico após ordenação topológica. Fonte (Cormen, 2012)

Em teoria dos grafos, uma “rede de fluxo” é definida como um grafo dirigido acíclico no qual cada aresta tem uma “capacidade² não negativa”. Há dois vértices distintos: “fonte” e “sumidouro”. Todo vértice pertencente ao grafo se encontra em um caminho entre a “fonte” e o “sumidouro”. A Figura 10 apresenta uma Rede de Fluxo (Cormen, 2012).

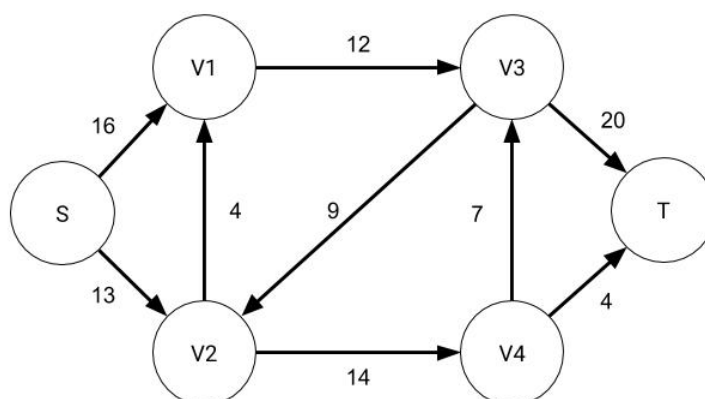


Figura 10 - Exemplo de Rede de Fluxo. Adaptado de (Cormen, 2012).

As “redes de fluxo” nesse trabalho podem ser utilizadas em “subgrafos”³ definidos a partir de dois vértices do grafo para calcular o esforço existente entre uma capacidade representada no grafo e uma outra subsequente.

Em “redes de fluxo”, o “fluxo máximo” é definido como o fluxo de intensidade máxima que respeita os limites de capacidade dos vértices.

² O termo “capacidade” utilizado aqui não deve ser confundido o mesmo termo utilizado no contexto de “níveis de capacidade”. Ele foi mantido nesse contexto porque assim está definido na literatura referenciada.

³ Um subgrafo SG do grafo G é um grafo cujo conjunto de vértices é um subconjunto do conjunto de vértices do grafo G.

A Figura 11 ilustra esse conceito. O vértice S é a fonte e o vértice T, o sumidouro. Os valores inteiros próximos às arestas na primeira imagem representam as capacidades máximas de cada aresta.

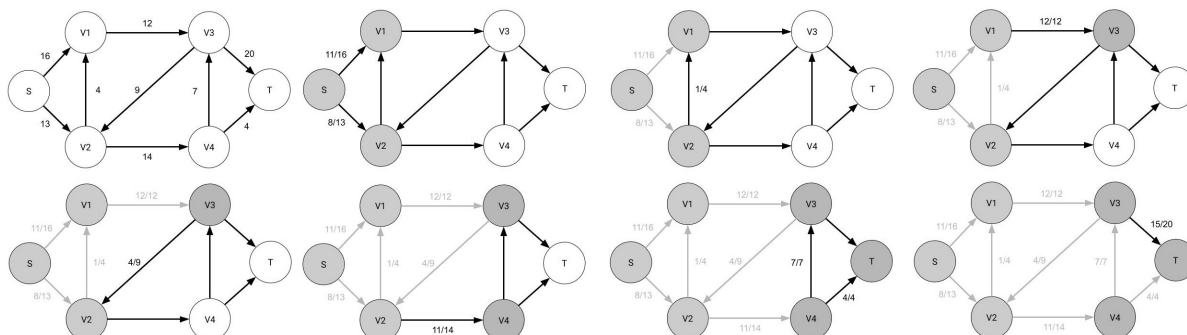


Figura 11 - Fluxo Máximo. Adaptado de (Cormen, 2012).

O cálculo do fluxo a cada iteração no grafo é detalhado na Tabela 1. Como pode-se notar, o fluxo máximo acontece na iteração 1, e tem valor de 19 unidades.

Tabela 1 - Cálculo de fluxo máximo

Iteração	Fluxo	Origem -> Destino (Unidades)	Estoque a cada iteração					
			S	V1	V2	V3	V4	T
1	19	S->V1 (11)	-	11	8	0	0	-
		S->V2 (8)						
2	1	V2->V1 (1)	-	12	7	0	0	-
3	12	V1->V3 (12)	-	0	7	12	0	-
4	4	V3->V2 (4)	-	0	11	8	0	-
5	11	V2 -> V4 (11)	-	0	0	8	11	-
6	7	V4 -> V3 (7)	-	0	0	15	0	4
7	4	V4 -> T (4)						
8	15	V3 -> T (15)	-	0	0	0	0	19

2.5 TRABALHOS RELACIONADOS

A representação do conhecimento através de grafos já é utilizada com objetivos diferentes do proposto por esse trabalho em outras aplicações.

O SKOS, *Simple Knowledge Organization System*, define uma forma padrão para representar sistemas de organização do conhecimento baseando-se no RDF,

Resource Description Framework. Seu propósito é compartilhar e associar sistemas de organização do conhecimento pela Web (Miles & Bechhofer, 2009).

O SKOS tenta utilizar das similaridades de outros sistemas de organização como tesouros, taxonomias, esquemas de classificação e sistemas de cabeçalhos de assunto para permitir o compartilhamento de dados em diferentes aplicativos.

O RDF é uma linguagem de representação de informações sobre recursos voltada para a Web. Seu objetivo principal é representar metadados sobre recursos da Web, como data e autor de uma página, informações de direitos autorais e licenciamentos (Manola & Miller, 2004). No entanto, sua estrutura generalista pode ser adaptada e utilizada para representar “coisas” que podem ser identificadas na Web. A Figura 12 mostra um grafo descrevendo Eric Miller, um dos autores da referência bibliográfica sobre RDF.

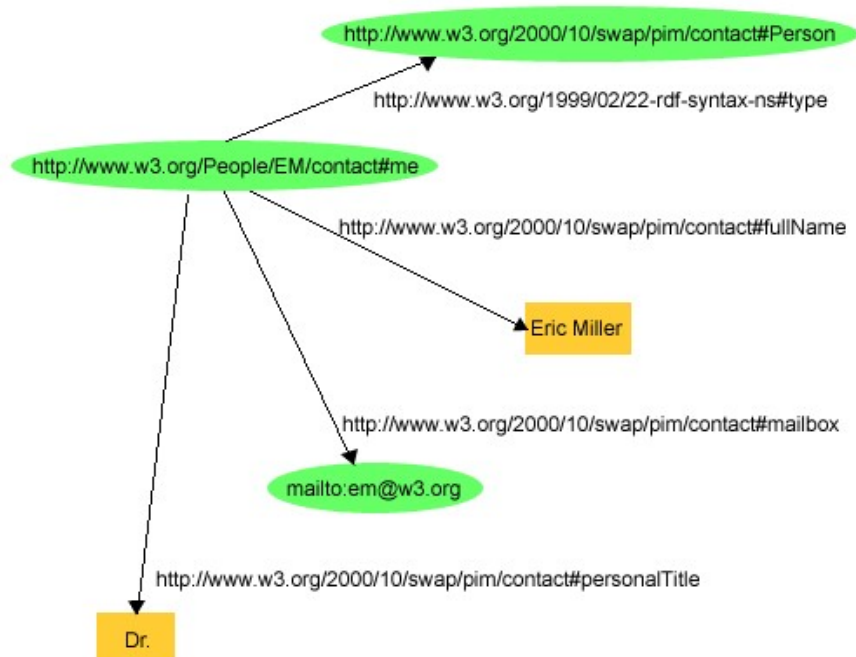


Figura 12 - Grafo RDF descrevendo Eric Miller, fonte (Manola & Miller, 2004)

O RDF se baseia em três conceitos, que formam uma Tripla RDF:

- Sujeito, a “coisa” descrita pela declaração RDF;
- Predicado, a propriedade específica do sujeito;
- Objeto, o valor da propriedade.

O sujeito e o objeto são representados como vértices e o predicado como aresta de um grafo, como mostrado na Figura 13.

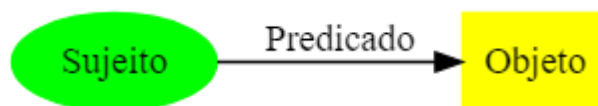


Figura 13 - Estrutura básica de um Grafo RDF

Um exemplo da utilização do RDF para a frase “A página <http://www.example.org/index.html> foi escrita por **John Smith** em **inglês** no dia **16 de agosto de 1999**” é o conjunto de sentenças abaixo representados pelo grafo na Figura 14.

- A página <http://www.example.org/index.html> tem um criador cujo valor é John Smith.
- A página <http://www.example.org/index.html> tem uma data de criação cujo valor é 16 de agosto de 1999
- A página <http://www.example.org/index.html> tem uma linguagem cujo valor é inglês.

Nas sentenças observamos o padrão: o **objeto** tem a **propriedade** cujo valor é **valor**.

A representação através de um grafo RDF combina diferentes sentenças (Triplas RDF) podendo o objeto de uma sentença ser sujeito em outra.

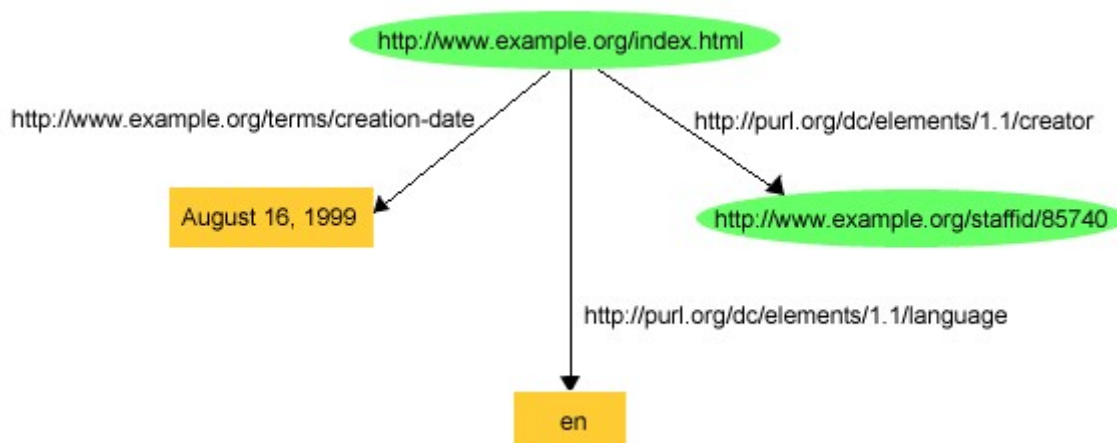


Figura 14 - Conjunto de declarações sobre o mesmo recurso, fonte (Manola & Miller, 2004).

Como pode ser observado, o SKOS e o RDF aplicam a teoria de grafos na representação do conhecimento com o propósito de descrever as relações entre sujeitos e objetos. A proposta deste trabalho é diferente, à medida que pretende

representar as relações de precedência entre capacidades a fim de descrever como se dá sua evolução.

3 ABORDAGEM PROPOSTA

A partir do referencial teórico apresentado no capítulo anterior, pretende-se desenvolver uma aplicação que represente os grafos de evolução da maturidade.

Para validar a aplicação construída, será utilizado o modelo de maturidade *DevOps* proposto por Marco Mendes⁴, a escolha se deve à familiaridade do autor com o modelo e sua aplicação prática imediata.

No modelo de maturidade *DevOps*, dentro de cada prática, Mendes propõe para cada nível de maturidade um conjunto de atividades que devem ser desenvolvidas. Cada uma dessas atividades será representada como um vértice do grafo. As relações de precedência entre as atividades serão definidas baseando-se na experiência do autor com o modelo e com desenvolvimento de software. As relações de precedência serão as arestas do grafo dirigido. A Figura 15 apresenta uma versão inicial dessa modelagem.

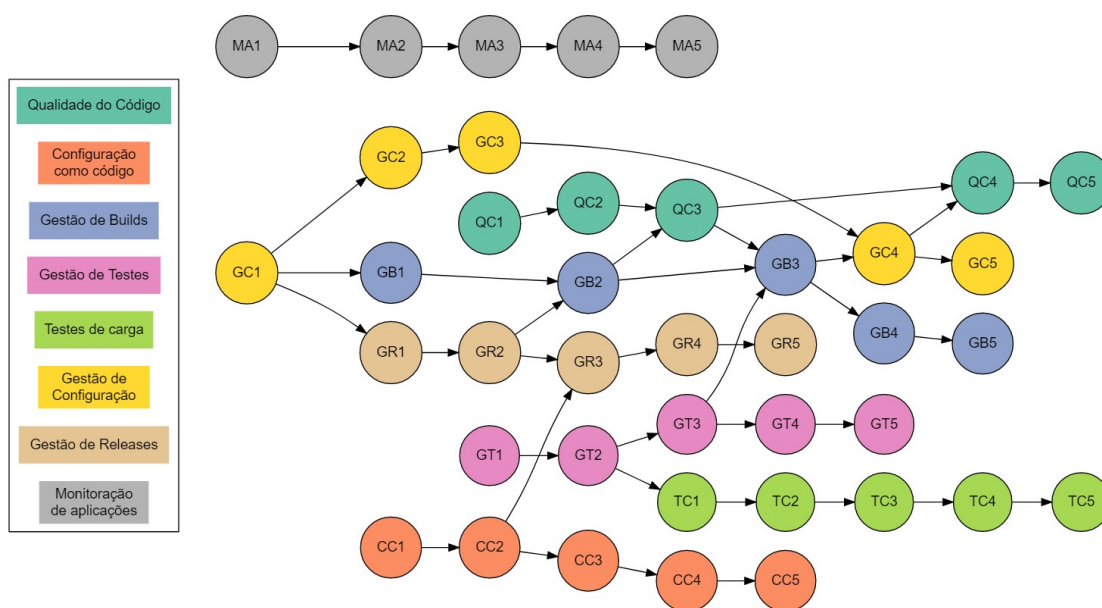


Figura 15 - Grafo para o Modelo de Maturidade *DevOps*

O desenvolvimento da aplicação deste trabalho será realizado em ambiente web, devido à experiência que o autor já tem com o ambiente e à disponibilidade de

⁴ Adicionalmente será utilizada também a grade curricular da graduação em Engenharia de Sistemas, especificada no próximo capítulo.

bibliotecas prontas para lidar com a representação gráfica de grafos nesse ambiente. Foram avaliadas duas bibliotecas JavaScript:

- Viz.js – Essa biblioteca é uma adaptação do software de visualização de grafos Graphviz para o ambiente Web. Para utilizar essa biblioteca, é necessário descrever o grafo na linguagem própria do software, DOT. (Daines, 2019)
- Vis.js – Essa é uma biblioteca de visualização de gráficos de propósito geral. Um de seus componentes é o Network, que pode ser utilizado para representar grafos. Além de ser uma biblioteca nativa para o uso em ambiente Web, a utilização dessa biblioteca é feita por meio de código estruturado, em que os vértices e arestas são representados como objetos (Almenda, 2019).

Por ser uma biblioteca nativa para ambiente web e por ter uma linguagem estruturada de representação, a biblioteca Vis.js mostrou ser mais adequada para os propósitos deste trabalho.

A aplicação será, portanto, desenvolvida utilizando as linguagens e ferramentas padrão para o desenvolvimento Web: JavaScript, HTML, *Hypertext Markup Language* e CSS, *Cascading Style Sheets*. Seu código fonte será disponibilizado em um repositório público da plataforma de hospedagem *GitHub*.

Para utilizar a aplicação, o usuário deverá acessar um endereço web a ser disponibilizado. A Figura 19 apresenta um protótipo de baixa fidelidade da aplicação.

A aplicação terá dois modos básicos: edição e análise do grafo, como mostrado na Figura 16.

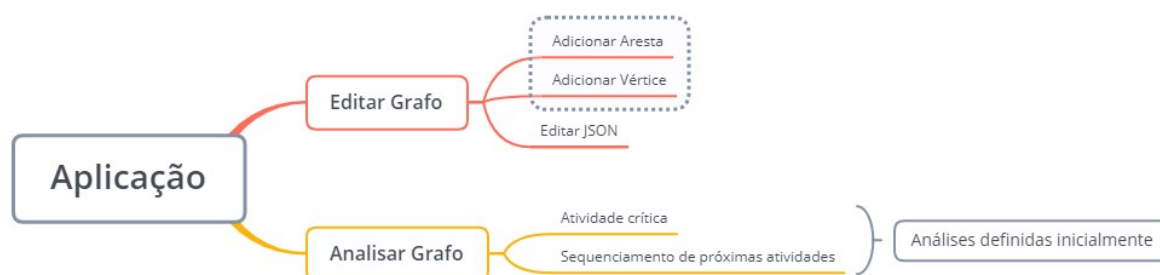


Figura 16 - Funcionalidades da aplicação a ser construída

No modo de edição, o usuário poderá definir a estrutura do grafo, acrescentando arestas e vértices.

Para adicionar vértices, Figura 17, o usuário deverá fornecer:

- Nome do vértice;
- Um peso associado, esse peso será utilizado para calcular de sequenciamento;
- Uma categoria a partir de uma lista pré-definida. A categoria será utilizada apenas para representar em cores diferentes o vértice no grafo;
- Se o vértice está realizado ou não. Para o caso do modelo de maturidade DevOps, esse campo define se a atividade já foi incorporada ou não. Essa informação será utilizada para calcular o sequenciamento de próximas atividades.



Adicionar Vértice

Nome:

Peso:

Categoria:

☒ Vértice realizado

Close Save Changes

Figura 17 - Aplicação, Adicionar Vértice

Para adicionar arestas, Figura 18, o usuário deverá fornecer os vértices de origem e destino.

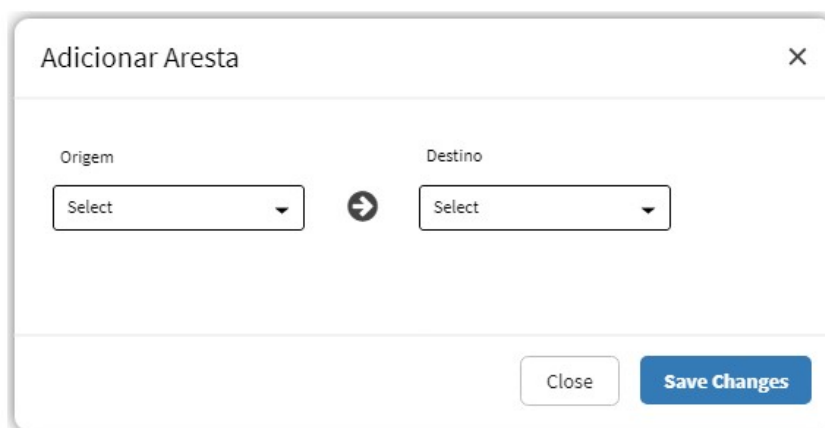
A modal window titled "Adicionar Aresta" with a close button (X) in the top right corner. Inside the modal, there are two labels: "Origem" and "Destino". Below each label is a dropdown menu with the text "Select" and a downward arrow. Between the two dropdowns is a circular arrow icon pointing to the right. At the bottom right of the modal, there are two buttons: a "Close" button and a "Save Changes" button.

Figura 18 - Aplicação, Adicionar Aresta

Os dados serão trabalhados pela aplicação em formato JSON, *JavaScript Object Notation*, em função disso será disponibilizada também a edição dos dados diretamente nesse formato por usuários que tenham domínio sobre essa sintaxe, a Figura 20 mostra um protótipo dessa funcionalidade.

Inicialmente são propostas duas análises nos grafos. Durante as próximas etapas do trabalho, é possível que surjam outras análises.

A análise de “atividade crítica” irá utilizar o conceito de peso de um vértice para definir quais são as atividades que têm maior impacto para a evolução da maturidade no grafo. Conhecer as atividades mais impactantes no avanço da maturidade é uma informação importante para as organizações, uma vez que elas podem concentrar mais esforços nessas atividades.

O “sequenciamento de próximas atividades” irá calcular, a partir das relações de precedência e da informação de realização ou não de uma atividade e utilizando conceitos de redes de fluxo, um sequenciamento de próximas atividades a serem realizadas. Essa análise é relevante porque auxilia a organização na definição dos próximos passos a serem tomados para evoluir naquele modelo de maturidade.

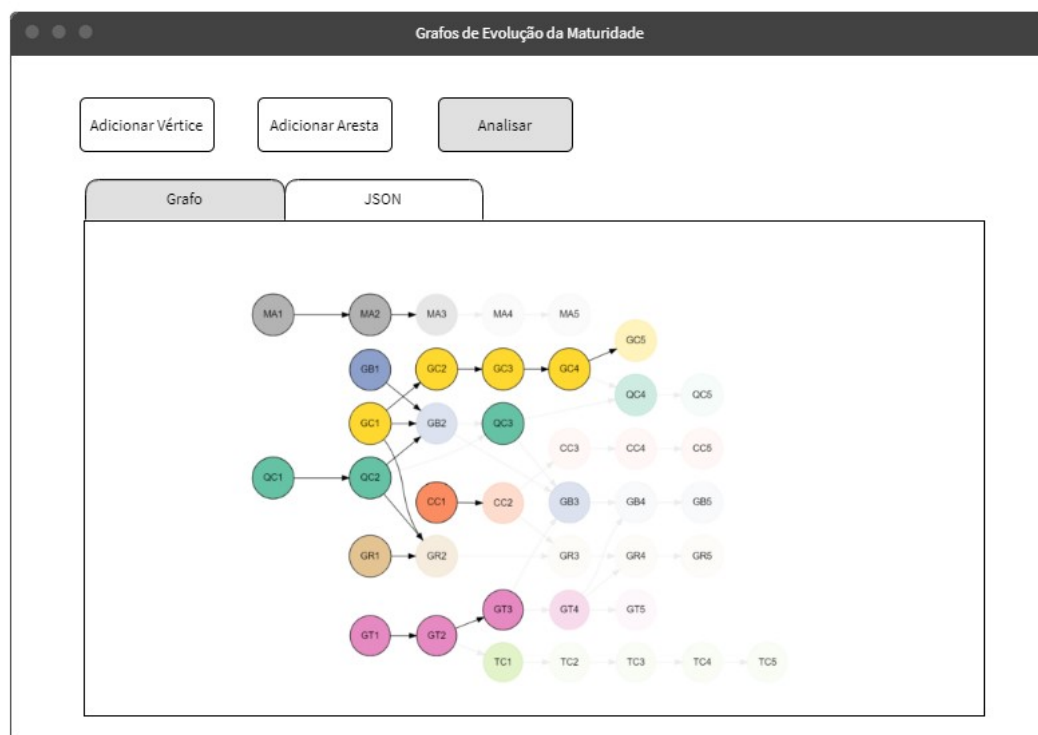


Figura 19 - Aplicação - Tela principal

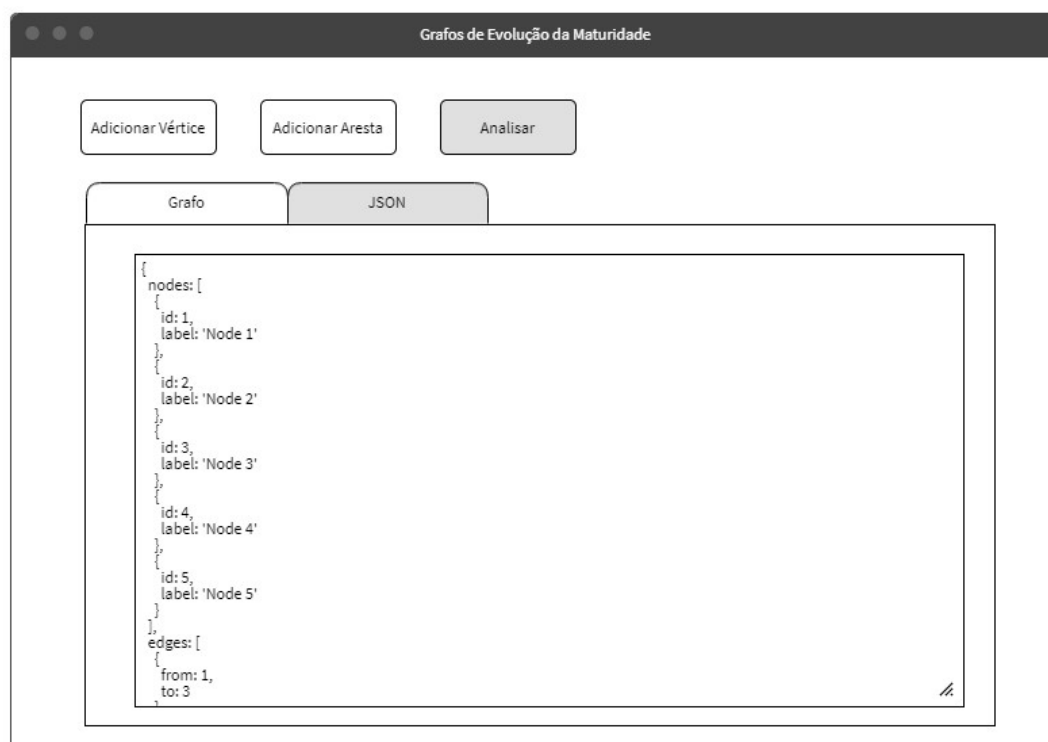


Figura 20 - Aplicação, Editar JSON

3.1 CRONOGRAMA

Para a continuação do trabalho será abordada uma estratégia baseada na metodologia Scrum, em que o projeto é dividido em iterações curtas com duração aproximada de 2 semanas, chamadas de sprint. A cada sprint um entregável evolutivo do trabalho é disponibilizado. Considerando a continuação do trabalho entre a primeira semana de agosto e a última de novembro, serão oito sprints detalhados na Tabela 2.

Tabela 2 - Cronograma

Sprint	Data Limite	Entregável
Sprint 01	16/Ago	Versão 0.1 da aplicação com a funcionalidade de desenhar um grafo a partir de um JSON pré-definido
Sprint 02	30/Ago	Versão 0.2 da aplicação com a funcionalidade de editar o grafo
Sprint 03	13/Set	Versão 0.3 da aplicação com uma funcionalidade de análise do grafo
Sprint 04	27/Set	Versão 0.4 da aplicação com mais uma funcionalidade de análise
Sprint 05	18/Out	Evolução da aplicação e Revisão bibliográfica
Sprint 06	1/Nov	Evolução da aplicação e experimentação com diferentes modelos
Sprint 07	15/Nov	Evolução da aplicação a partir de feedbacks recebidos
Sprint 08	29/Nov	Versão final da aplicação com análise e discussão de resultados

4 POTENCIAIS APLICAÇÕES

Este trabalho propõe uma forma de representação da evolução da maturidade e a criação de uma ferramenta que faça essa representação. Para validar a ideia será utilizado o modelo de maturidade *DevOps* proposto por Marco Mendes, no entanto essa forma de representação e a ferramenta criadas têm outras potenciais aplicações, como mostrado neste capítulo.

4.1 GRADE CURRICULAR

Apesar de não ser tratado como um modelo de evolução da maturidade nos moldes dos modelos descritos anteriormente, estruturas do tipo grades curriculares podem representar uma rede para a evolução do conhecimento.

Assim como os modelos de maturidade definem os critérios e parâmetros de medição da evolução das capacidades, a grade curricular de um curso de graduação – ao definir e sequenciar as disciplinas obrigatórias e optativas que devem ser cumpridas por um estudante – funcionam como modelos de evolução do conhecimento.

As disciplinas podem ser compreendidas como capacidades, ou vértices nos grafos. E as relações de pré-requisitos formais podem ser entendidas como vínculos entre essas capacidades, ou arestas nos grafos.

Pretende-se neste trabalho utilizar a grade curricular do curso de graduação em Engenharia de Sistemas também para validar a ferramenta proposta.

As disciplinas e suas relações de pré-requisitos foram modeladas como o Grafo apresentado na Figura 21.

Na Figura 22 as disciplinas foram agrupadas por seus respectivos períodos curriculares.

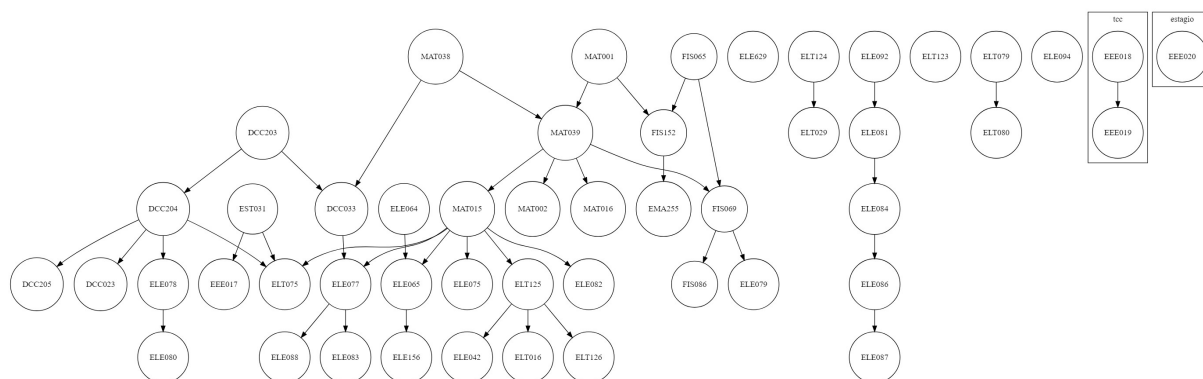


Figura 21 - Grade Curricular do Curso de Graduação em Engenharia de Sistemas

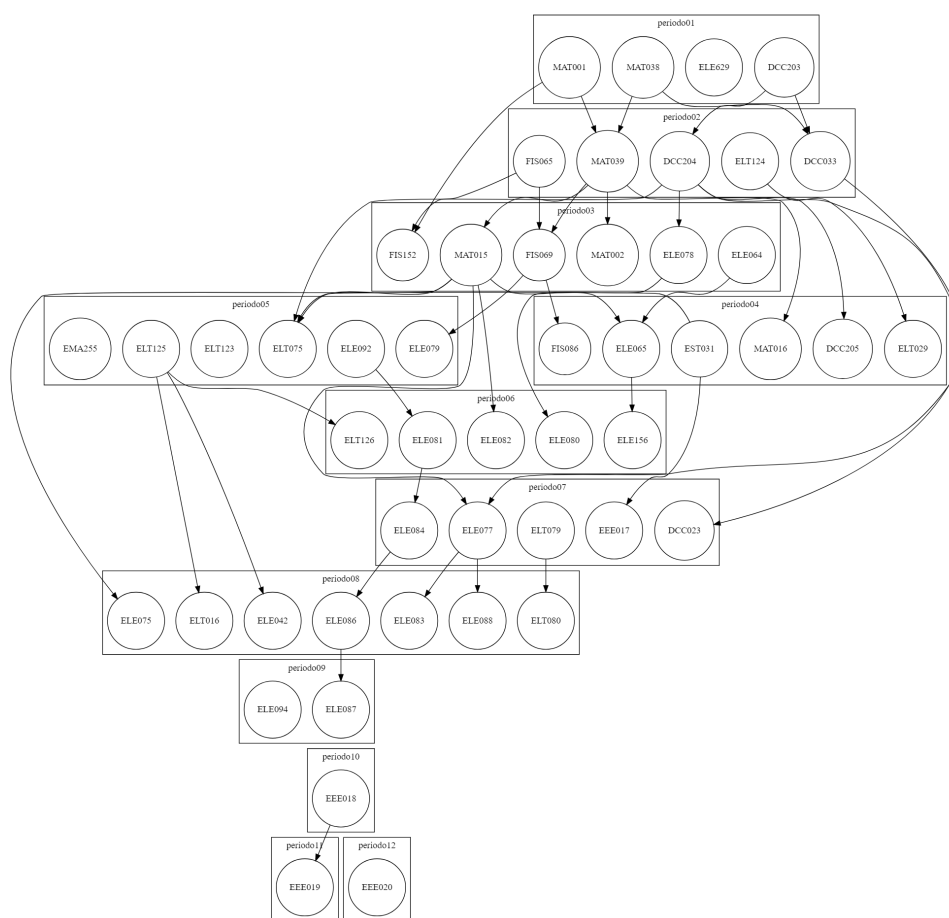


Figura 22 - Grade Curricular do Curso de Graduação em Engenharia de Sistemas, agrupadas por período

4.2 SUSTENTABILIDADE E GESTÃO AMBIENTAL

O Modelo Hierárquico de Lowell é uma ferramenta desenvolvida para avaliar o nível de maturidade de uma organização com relação à sustentabilidade (da Rocha, Mendes, & Moris, Número 30, Dezembro de 2013).

Nesse modelo, são propostos cinco níveis de maturidade:

- Nível 1: Indicadores de conformidade e cumprimento
- Nível 2: Indicadores de performance e utilização de materiais
- Nível 3: Indicadores dos efeitos gerados pela empresa
- Nível 4: Indicadores de ciclo de vida e cadeia de suprimentos
- Nível 5: Indicadores de sistemas sustentáveis

Em cada nível são definidos diversos indicadores que devem ser implementados para que o nível seja atingido. Assim como foi feito com as disciplinas e as relações de pré-requisito, é possível modelar os indicadores de Lowell como vértices e suas relações como arestas para representar esse modelo de maturidade como um grafo.

Em Costa Filho & Rosa, 2017, a maturidade em gestão ambiental de uma organização é classificada em três níveis: passiva, reativa e proativa.

No nível de abordagem passiva, a organização considera as questões ambientais como secundárias, não há investimentos e a empresa fica sujeita a multas e penalidades legais. No nível de abordagem reativa, a organização cumpre a legislação obrigatória a fim de evitar penalidades. E já no nível de abordagem proativa, as ações relacionadas ao meio-ambiente são incorporadas ao planejamento de atividades operacionais da organização.

Após definir os três níveis, Costa Filho & Rosa fazem a análise do nível de maturidade em gestão ambiental de três empresas no estado de Goiás.

A utilização de grafos com as relações de precedência entre as habilidades e capacidades adquiridas e por adquirir pelas empresas avaliadas potencialmente poderia trazer uma melhor visualização da real situação dessas e de outras organizações quanto à gestão ambiental.

4.3 EMPREENDEDORISMO

Modelos de maturidade também são utilizados para avaliar empresas vinculadas a incubadoras. de Miranda, de Menezes, Ferreira, & Mendonça, 2015 propõe um modelo para avaliação das empresas vinculadas a IEFT/UFV,

Incubadora de Empresas de Base Tecnológica da Universidade Federal de Viçosa/MG.

Esse modelo é uma adaptação do modelo de referência do CERNE, Centro de Referência para Apoio a Novos Empreendimentos. Essa adaptação foi feita para melhor se adequar às empresas incubadas naquela instituição.

No modelo proposta para a IEBT/UFV são utilizados cinco eixos e cinco níveis de maturidade.

Os eixos são:

- Tecnologia – desenvolvimento da solução;
- Gestão – utilização de técnicas modernas de gestão;
- Mercado – avaliação quanto à viabilidade mercadológica da solução;
- Financeiro – evolução financeira do empreendimento;
- Empreendedor – desenvolvimento pessoal dos empreendedores.

Os níveis de maturidade são:

- Nível 0 – Seleção
- Nível 1 – Planejamento e Compreensão
- Nível 2 – Desenvolvimento e Organização I
- Nível 3 – Desenvolvimento e Organização II
- Nível 4 – Sustentabilidade e Maturidade

Dentro de cada nível, há uma série de critérios que devem ser atendidos pelas empresas para que elas sejam consideradas pertencentes àquele nível, como ilustra a Figura 23.

Para esse modelo, os grafos de evolução da maturidade seriam facilmente adaptados, uma vez que esse modelo segue a mesma estrutura dos demais apresentados neste trabalho.

Eixo	Critério
Gestão	Planejamento Estratégico
	Promoção do comportamento ético
	EVTECIAS (Estudo de Viabilidade Técnica, Econômica, Comercial e de Impacto Ambiental e Social)
Financeiro	Planejamento Financeiro
	Contabilidade em dia
	Fluxo de Caixa
	Definição da participação dos sócios
Tecnologia	Registro da Propriedade Intelectual
	Desenvolvimento contínuo do produto de base tecnológica
	Desenvolvimento de produtos não tecnológicos
Mercado	Pesquisa de mercado
	Plano de Marketing
Empreendedor	Plano de Desenvolvimento Pessoal
	2 (duas) atividades de qualificação:

Figura 23 - Critérios para o Nível de Maturidade Planejamento e Compreensão, Fonte (de Miranda, de Menezes, Ferreira, & Mendonça, 2015)

4.4 MÚSICA

Music Map é um projeto que busca representar a genealogia da música. (Crauwels, 2019).

Nessa representação, o eixo horizontal agrupa os diferentes gêneros de música e o eixo vertical mostra o avanço do tempo, como mostrado na Figura 24.

Quando se dá um zoom na figura, os diferentes estilos musicais são mostrados e as relações entre eles são apresentadas, como mostra o detalhe na Figura 25.

As unidades e as relações entre elas já estão definidas para esse modelo, de forma que já existem as informações necessárias para modelar um grafo. A partir dessa modelagem, seria possível aplicar algoritmos de fluxo em grafos para encontrar relações emergentes entre os gêneros musicais.

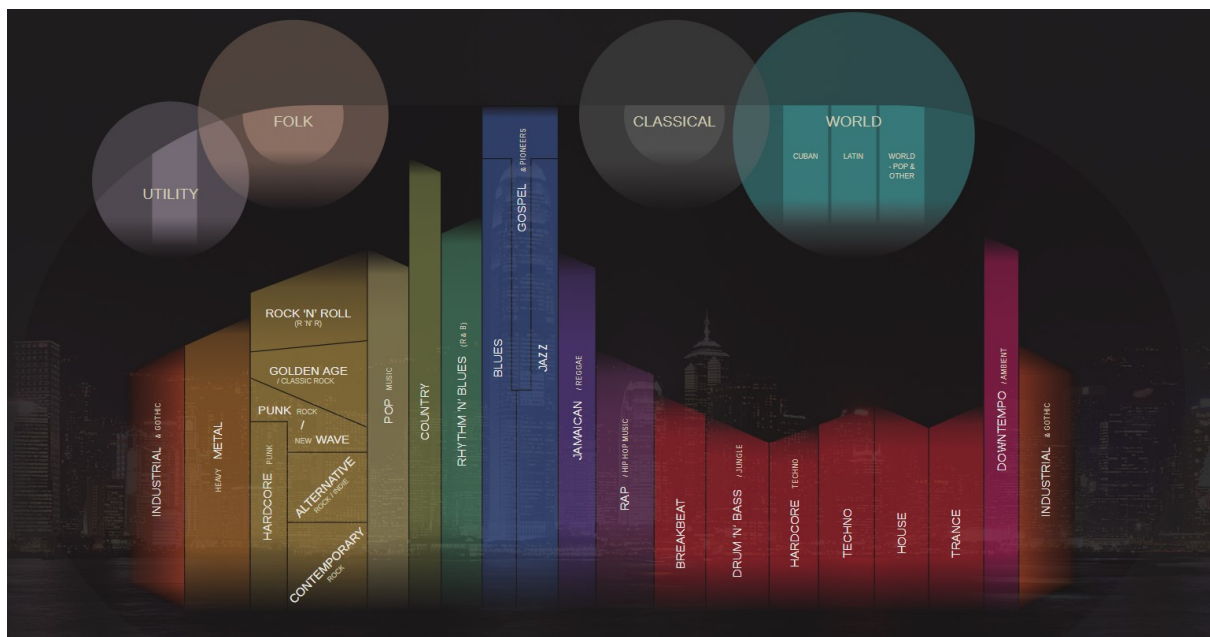


Figura 24 - Projeto Music Map, Fonte (Crauwels, 2019)

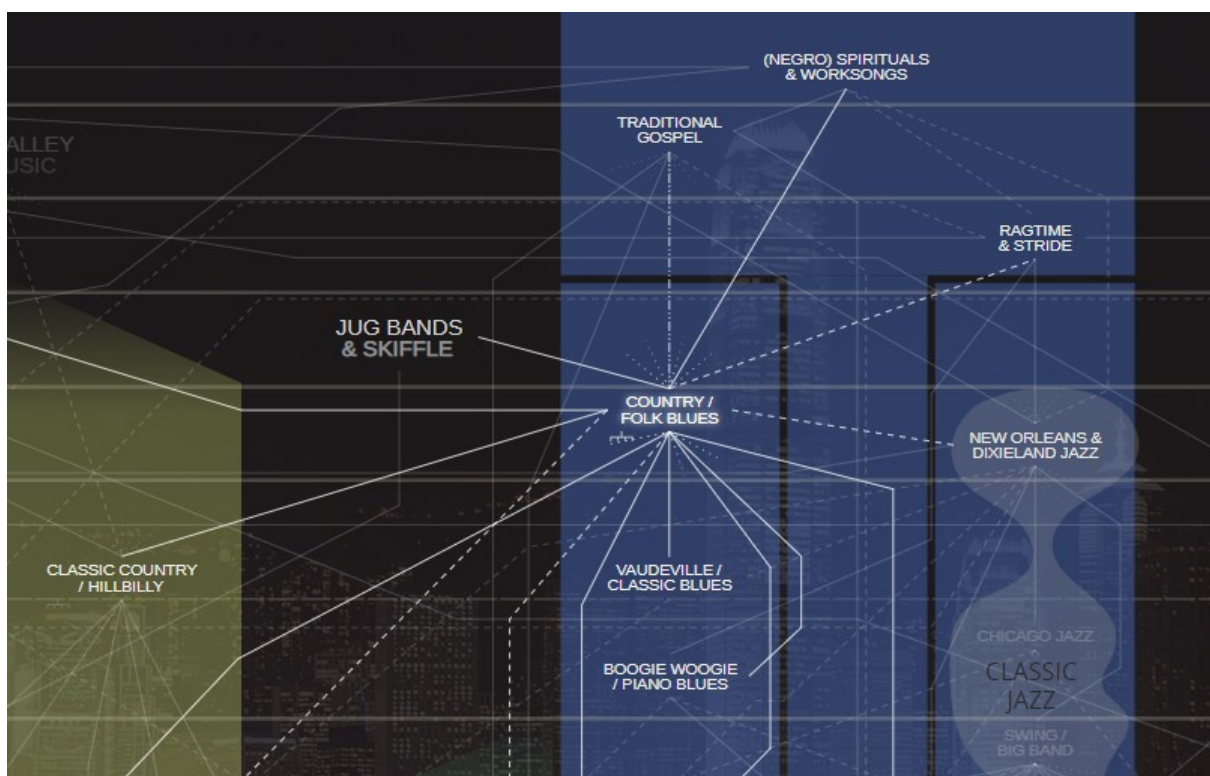


Figura 25 - Music Map, detalhe, Fonte (Crauwels, 2019)

5 CONSIDERAÇÕES FINAIS

Modelos de maturidade são ferramentas utilizadas para avaliar as capacidades de um indivíduo ou organização com relação a um processo ou objetivo específico. As representações usuais dos modelos de maturidade são restritivas e comumente deixam de lado especificidades de indivíduos ou organizações que tentam evoluir seguindo aquele modelo.

A teoria de grafos é um ramo da matemática responsável por estudar e definir estruturas chamadas grafos que representam entidades e suas relações. São diversas as aplicações de grafos, inclusive na gestão do conhecimento através dos grafos RDF. Este trabalho apresentou a proposta de se utilizar grafos para representar modelos de maturidade.

Para justificar essa proposição foi utilizado o Framework Cynefin, uma ferramenta de auxílio à classificação de sistemas. Nesse framework, sistemas com relações de causa e efeito evidentes são trabalhados no domínio do Simples/Óbvio; sistemas com relações de causa e efeito não evidentes e dispersas estão no domínio do Complicado. Os modelos de maturidade no formato atual podem ser entendidos como ferramentas no domínio do Simples/Óbvio. Ao analisar os modelos de maturidade através da teoria de grafos, as relações de causa e efeito passam a não ser evidentes, surgindo relações emergentes, e dessa forma, a proposta desse trabalho tenta levar a análise da evolução da maturidade para o domínio do Complicado.

Foi proposta a criação de uma aplicação que represente os modelos de maturidade através de grafos, os requisitos iniciais dessa aplicação e um cronograma básico para execução foram apresentados.

Foram avaliadas também potenciais aplicações do modelo proposto aqui e da aplicação que será criada.

Com as definições aqui apresentadas, este trabalho segue para a implementação da aplicação proposta.

REFERÊNCIAS

- Almenda, O. N. (2019). *Documentação vis.js*. Fonte: vis.js: <http://visjs.org/>
- Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. Bookman.
- Cormen, T. H. (2012). *Algoritmos, Teoria e Prática*. Rio de Janeiro: Elsevier.
- Costa Filho, B. A., & Rosa, F. d. (2017). Maturidade em Gestão Ambiental: Revisitando as melhores práticas. *REAd. Revista Eletrônica de Administração*, vol. 23 no. 2.
- Crauwels, K. (3 de Junho de 2019). *Music Map*. Fonte: Music Map: <https://www.musicmap.info/>
- Crescente, C. (7 de Agosto de 2018). *Cynefin para todos!* Fonte: <https://medium.com/@clau.crescente/cynefin-para-todos-123668785ac4>
- Crosby, P. B. (1979). *Quality Is Free*. New York: McGraw-Hill.
- da Rocha, K. E., Mendes, J. V., & Moris, V. A. (Número 30, Dezembro de 2013). Avaliação do nível de maturidade em sustentabilidade através. *Revista Brasileira de Ciências Ambientais*, 68-78.
- Daines, M. (2019). *Documentação viz.js*. Fonte: viz.js: <https://github.com/mdaines/viz.js/wiki>
- de Miranda, A. F., de Menezes, B. M., Ferreira, N. M., & Mendonça, P. J. (2015). Modelo de Maturidade: uma ferramenta para mensurar o desenvolvimento. *25ª Conferência ANPROTEC*.
- Forsgreen, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press.
- Fowler, M. (22 de Agosto de 2014). *ShuHaRi*. Fonte: Martin Fowler: <https://martinfowler.com/bliki/ShuHaRi.html>
- Keogh, L. (19 de Julho de 2018). *Cynefin for Everyone!* Fonte: <https://medium.com/@lunivore/cynefin-for-everyone-d5f47d9bd102>

- Kim, G., Humble, J., & Debois, P. (2018). *Manual de Devops. Como Obter Agilidade, Confiabilidade e Segurança em Organizações Tecnológicas*. Alta Books.
- Koehlegger, M., Maier, R., & Thalmann, S. (2009). Understanding Maturity Models Results of a Structured. *Proceedings of I-KNOW '09: 9th international conference on knowledge management and knowledge technologies*.
- Kurtz, C. F., & Snowden, D. J. (2003). The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Systemas Journal - Volume: 42, Issue: 3*, 462-483.
- Manola, F., & Miller, E. (10 de Fevereiro de 2004). *RDF Primer*. Fonte: The World Wide Web Consortium (W3C): <https://www.w3.org/TR/2004/REC-rdf-primer-20040210/>
- Mendes, M. (13 de Dezembro de 2016). *Maturidade em Práticas DevOps*. Fonte: Marco Mendes - Agile, lean, arquitetura, programação e devops: <https://marco-mendes.com/2016/12/13/maturidade-em-praticas-devops/>
- Miles, A., & Bechhofer, S. (18 de Agosto de 2009). *SKOS Simple Knowledge Organization System Reference*. Fonte: The World Wide Web Consortium (W3C): <https://www.w3.org/TR/skos-reference/>
- Moreira, D. A. (2005). *Teoria e prática em gestão do conhecimento*. Belo Horizonte: Dissertação (Mestrado em Ciência da Informação, UFMG).
- Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation*. Nova York: Oxford University Press.
- Ribas, T. (1 de Outubro de 2018). *Como o gestor pode tomar melhores decisões*. Fonte: <https://thomazribas.com/como-o-gestor-pode-tomar-melhores-decisoes/>
- Salviano, C. F. (2003). *Melhoria e Avaliação de Processo com ISO/IEC 15504 (SPICE) e CMMI*. Lavras: Universidade Federal de Lavras.
- Snowden, D. J., & Boone, M. E. (Novembro de 2007). A leader's framework for decision making. A leader's framework for decision making. *Harvard Business Review*, pp. 69-76.
- University of Wales. (12 de Junho de 2019). *Welsh-English / English-Welsh On-line Dictionary*. Fonte: Welsh-English / English-Welsh On-line Dictionary: <http://www.geiriadur.net/index.php?page=ateb&term=cynefin&direction=we&type=all&whichpart=beginning>

Weinberg, G. M. (1992). *Software com qualidade*. Rio de Janeiro: Makron Books.

Zambalde, A. L., & Alvez, R. M. (2004). *Gestão do Conhecimento e Inovação*.
Lavras: Universidade Federal de Lavras.

Ziviani, N. (2011). *Projeto de Algoritmos*. São Paulo: Cengage Learning.