Gestão do Conhecimento em Empresas de Desenvolvimento Ágil de Software: Conceitos e Práticas Essenciais

Eduardo Sakai, Ivan L M Ricarte, *Universidade Estadual de Campinas, UNICAMP*

***Abstract*— This paper presents the state of the art of knowledge management in software development, organized around key concepts and practices grouped in epistemological and ontological dimensions, development approach, space, and culture. This classification results from the thematic analysis of 35 research papers available on this topic, selected from an initial set of 1620 papers, through a bibliographic research that followed the principles of systematic reviews. As an original contribution, this paper presents a classification of agile software development techniques and some potential enabling factors according to the epistemology, ontology, and the SECI (Socialization, Externalization, Combination and Internalization) model.**

***Index Terms*— Agile software development, Knowledge management, Organizational aspects.**

***Resumo*— Este artigo apresenta o estado da arte da gestão do conhecimento no desenvolvimento de software, organizado segundo conceitos e práticas relacionados a dimensão epistemológica, dimensão ontológica, abordagem de desenvolvimento, espaço e cultura. Esta classificação é o resultado da análise temática de 35 artigos selecionados de um total de 1620 artigos sobre esses tópicos, levantados por meio de uma pesquisa bibliográfica que seguiu os princípios de revisões sistemáticas. Como contribuição original, este trabalho apresenta uma classificação das técnicas de desenvolvimento ágil de software e de alguns possíveis fatores capacitadores de acordo com as dimensões epistemológica, ontológica e com o modelo SECI.**

***Palavras-chave* – Desenvolvimento ágil de software, Gestão do conhecimento, Aspectos organizacionais.**

I. Introdução

Na atual era da informação, em uma economia na qual a única certeza é a incerteza, apenas o conhecimento se apresenta com uma fonte segura de vantagem competitiva [1]. Nesse contexto, organizações focam na gestão do conhecimento e do capital intelectual como seu mais importante patrimônio; ativos intangíveis passam a ter uma abordagem voltada para a aprendizagem organizacional [2].

A globalização da economia e a rapidez das alterações no contexto social e político afetam igualmente a sobrevivência imediata e a viabilidade futura das empresas. A intensidade da competição, a vulnerabilidade de mercados, a versatilidade da clientela e a variação tecnológica fazem da mudança a essência da gerência [3]. Nesse sentido, inicia-se uma tendência crescente de substituir o modelo estático tradicional por um que represente a organização como um sistema dinâmico.

No contexto de desenvolvimento de software, abordagens ágeis surgem na tentativa de suprir essa necessidade a rápidas mudanças, tanto de requisitos como de tecnologias. As equipes devem ser auto organizáveis pois os seus membros possuem autonomia. As equipes de desenvolvimento de software têm uma estrutura relativamente única, no qual a divisão do trabalho entre os membros é altamente interdependente; assim, gerenciar o conhecimento é a chave do processo [4].

Este trabalho focou nessa inter-relação entre a gestão do conhecimento e o desenvolvimento de software. Na sequência, a Seção 2 traz uma fundamentação teórica dos conceitos de gestão do conhecimento e de desenvolvimento de software. A Seção 3 sintetiza os aspectos metodológicos adotados para o desenvolvimento desta pesquisa, com foco na análise qualitativa da literatura. A Seção 4 traz uma síntese do estado da arte sobre esse tema. A Seção 5 apresenta uma classificação das técnicas utilizadas nas metodologias de desenvolvimento de software e dos fatores capacitadores para a criação de conhecimento. Finalmente, são apresentadas as conclusões e considerações finais deste trabalho.

II. Fundamentação Teórica

As bases teóricas deste estudo transcendem a disciplina de engenharia de software, passando pela epistemologia e administração, ou, mais especificamente, pela gestão do conhecimento e pelas abordagens metodológicas de desenvolvimento de software.

A. Gestão do Conhecimento

Estudos sobre a teoria do conhecimento são tão antigos quanto a própria história da filosofia ocidental. A capacidade de raciocinar e criar conhecimento diferencia o ser humano. Desses estudos derivou-se o ramo da filosofia conhecido como

epistemologia. Segundo Nonaka e Takeuchi [5], "filósofos ocidentais concordam que conhecimento é a 'crença verdadeira e justificada'. "

Michael Polanyi, ainda no domínio da filosofia, introduziu a distinção entre conhecimento tácito e conhecimento explícito. Polanyi acreditava que o conhecimento tem uma componente pessoal indispensável. O conhecimento explícito refere-se a todo tipo de conhecimento que pode ser formulado e transmitido por meio de livros, manuais ou mesmo por meio da linguagem formal. Já o conhecimento tácito é todo o conhecimento inarticulado, difícil de expressar e tem um caráter pessoal extremamente relevante [5].

A distinção entre o conhecimento tácito e o conhecimento explícito é o ponto central da teoria de Nonaka e Takeuchi e compõe a dimensão epistemológica do modelo de criação de conhecimento [5]. A outra dimensão do modelo, a dimensão ontológica, distingue em que nível ocorre a criação do conhecimento, distinguindo entre os níveis individual, grupo de desenvolvimento, e organizacional.

O modelo dinâmico de criação de conhecimento proposto por Nonaka e Takeuchi [5] "está ancorado no pressuposto crítico de que o conhecimento humano é criado e expandido através das interações sociais entre o conhecimento tácito e o conhecimento explícito. " A partir deste pressuposto, os autores definem quatro modos de conversão de conhecimento, conhecido como espiral do conhecimento ou modelo SECI (Socialização, Externalização, Combinação, Internalização). Socialização é a conversão de conhecimento tácito em conhecimento tácito; gera o conhecimento compartilhado. Externalização é a conversão de conhecimento tácito em conhecimento explícito; gera o conhecimento conceitual. Combinação é a conversão de conhecimento explícito em conhecimento explícito; dá origem ao conhecimento sistêmico. Internalização é a conversão de conhecimento explícito em conhecimento tácito; produz o conhecimento operacional.

B. Desenvolvimento de software

Existem diversas definições para software, mas de maneira geral os autores concordam que o software não é apenas o código em execução sobre algum hardware. Ele envolve, além dos códigos, os requisitos, os algoritmos, os processos, os manuais e a manutenção ao longo do tempo. Sommerville afirma que "software não é apenas o programa, mas também todos os dados de documentação e configuração associados, necessários para que o programa opere corretamente" e define engenharia de software como "uma disciplina de engenharia relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção, depois que este entrar em operação. " [6]

É possível dividir o desenvolvimento de software em duas grandes abordagens. A primeira é a abordagem tradicional, baseada em repositórios, também conhecida como abordagem taylorista. A outra abordagem, baseada em comunidade, tem a denominação usual de abordagens ágeis.

Da abordagem tradicional, baseada em repositórios, destacam-se modelos de processo de desenvolvimento de software como modelo cascata, modelos incrementais, modelos evolucionários e modelos especializados de processos. De maneira geral, destaca-se a divisão muito bem definida das etapas dos métodos tradicionais, ou atividades de arcabouço. Essa divisão pode levar à especialização do trabalho em equipes que não forem multifuncionais. Outro ponto de destaque é a necessidade de documentação rigorosa, seja na dependência de uma análise de requisito extremamente elaborada ou na geração de documentos atualizados ao final de cada etapa ou de uma nova versão do sistema. Assim, a documentação é extremamente exaustiva e requer muito tempo e esforço para mantê-la atualizada.

No ambiente global atual, os negócios estão sujeitos a rápidas mudanças e desta forma é praticamente impossível elaborar um conjunto completo de requisitos no início de projetos de software. Muitas vezes, nem os próprios clientes que apresentam o conhecimento de domínio sabem exatamente o que precisam para seus negócios. A engenharia de software foi obrigada e rever alguns conceitos para oferecer a agilidade necessária para atender a mudanças exigidas pelo ambiente no decorrer do projeto. Baseada nos sucessos dos processos enxutos (*lean*) da indústria de bens e mercadorias tangíveis, no final da década de 1990 foram criados alguns métodos que quebravam a linha vigente das metodologias tradicionais e enfatizavam a resposta a mudanças e o fluxo de valor. São os chamados métodos ágeis de desenvolvimento de software.

Da abordagem ágil, destacam-se: o *Extreme Programming* (XP), o *Scrum*, o Desenvolvimento Adaptativo de Software (DAS), o Método de Desenvolvimento Dinâmico de Sistemas (*Dynamic System Development Method* – DSDM), o Desenvolvimento Guiado por Características (*Feature Driven Development*), o Crystal e a Modelagem Ágil (*Agile Modeling* – AM). Como afirma Sommerville, "embora esses métodos ágeis sejam todos baseados na noção de desenvolvimento e entrega incrementais, eles propõem processos diferentes para se conseguir isso. Contudo, compartilham um conjunto de princípios e, portanto, têm muito em comum" [6]. A agilidade refere-se à forma como uma metodologia responde à necessidade constante de mudança, devendo ser rápida e adaptativa, além de focar na comunicação eficiente entre todos os envolvidos no projeto, aproximando o cliente da equipe de desenvolvimento, efetuando entregas incrementais, frequentes, periódicas e funcionais de software ao longo do desenvolvimento [7].

Cada método define suas técnicas e artefatos. As principais técnicas e artefatos do *Scrum* são: *backlog* do produto, *backlog* do *sprint*, reuniões diárias do *scrum*, planejamento de *sprint*, revisão do *sprint*, retrospectiva do *sprint*, *burndown chart*, quadro de tarefas. Do XP, são: jogo de planejamento, pequenos releases, metáforas, equipe inteira, teste de cliente, ritmo sustentável (40 horas semanais), reuniões em pé, propriedade coletiva do código, programação em par, padronização de códigos, desenvolvimento orientado a testes, refatoração, integração contínua, o princípio de manter a simplicidade, cliente sempre disponível, testes de aceitação, testar antes de projetar (*test first design*), força tarefa.

III. MÉTODOS

Para identificar os trabalhos que já abordaram a relação entre gestão do conhecimento e o desenvolvimento de software, foi realizada uma revisão de literatura resultante de uma pesquisa bibliográfica, a qual tem como objetivo coletar todos os artigos publicados que reportam o tema de interesse [8]. Essa revisão seguiu os princípios da revisão sistemática de literatura e da análise temática, uma forma de pesquisa qualitativa documental.

A revisão de literatura é uma das formas de analisar e discutir os artigos publicados em uma área específica do conhecimento. Ela pode ter como objetivo a verificação de textos e artigos relacionados ao assunto estudado e que já foram publicados em congressos, simpósios, revistas e foram cadastrados em alguma base de dados ou conhecer a forma como esse assunto foi abordado e analisado em estudos anteriores. Segundo Macedo [9], a revisão de literatura "é a busca de informações bibliográficas, seleção de documentos que se relacionam com o problema de pesquisa".

Moher et al. [10] afirmam que a revisão sistemática de literatura é uma revisão de uma pergunta claramente formulada que usa métodos sistemáticos e explícitos para identificar, selecionar e avaliar criticamente pesquisas relevantes, coletar e analisar dados dos estudos incluídos na revisão. Métodos estatísticos podem ou não ser utilizados para analisar e resumir os resultados dos estudos incluídos. As revisões sistemáticas são particularmente valiosas como um meio de analisar todas as evidências sobre uma questão particular, se há alguma incerteza sobre a resposta.

Para esta revisão de literatura, a pesquisa bibliográfica teve como tema *"gestão do conhecimento no desenvolvimento de software"*, com foco temático relacionado a metodologias ágeis. Para realizar essa pesquisa e apresentar seus resultados neste artigo, o *framework* PRISMA *(Preferred Reporting Items for Systematic Reviews and Meta-Analysis)* de revisão sistemática [10] foi adotado.

Os textos foram procurados nas bases bibliográficas IEEExplore, ACM Digital Library e ScienceDirect. Os principais conceitos, *conhecimento* e *desenvolvimento de software*, foram combinados por meio de operadores booleanos E (*AND*). Para refinar a busca, foi pré-determinado que o conceito "conhecimento" deveria estar explícito no título; o conceito "desenvolvimento de software" e suas variantes relevantes para esta pesquisa (metodologias ágeis, desenvolvimento ágil de software, melhoria de software, *scrum*, programação extrema, XP) foram combinados com o operador booleano OU (OR) e poderiam constar em qualquer parte dos textos. Por exemplo, a expressão exata utilizada para a base IEEExplore foi: (("*Document Title*":"*knowledge*") AND ("*software development*" OR "*agile methodology*" OR "*scrum*" OR "*agile software development*" OR "*extreme programming*" OR "XP" OR "*software environment*")). A Figura 1 ilustra os resultados obtidos e as principais fases do processo por meio de um diagrama PRISMA.

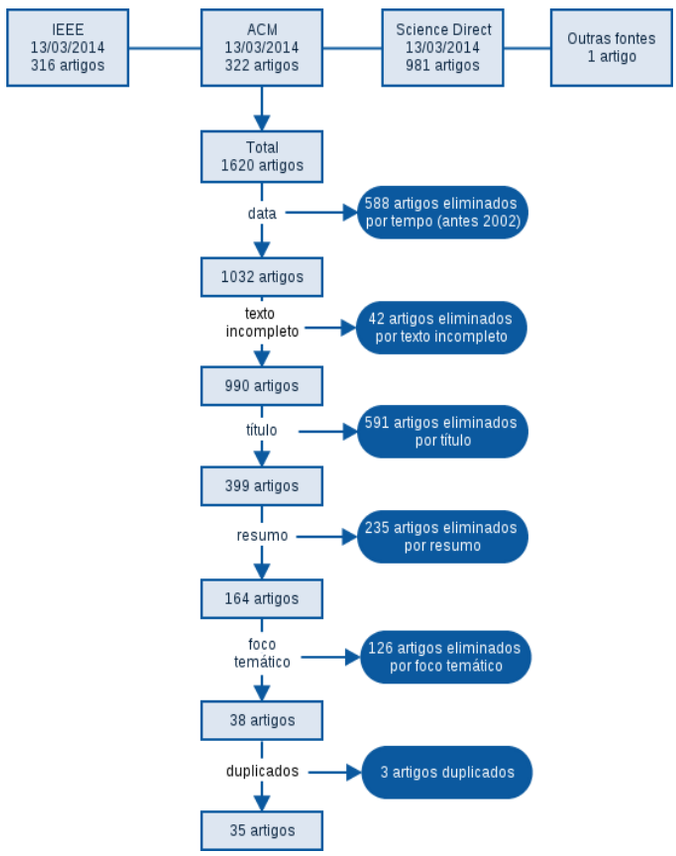

Fig. 1. Diagrama PRISMA. Das pesquisas bibliográficas realizadas em três bases de dados, 1620 artigos foram recuperados e, após aplicação de critérios de exclusão, 35 foram incluídos na revisão.

Após a seleção dos artigos foi feita uma tentativa de agrupar os conceitos semelhantes. Os artigos foram agrupados de acordo com os focos abordados e pertinentes a dimensão epistemológica e a dimensão ontológica conforme o trabalho de Nonaka e Takeuchi [5] e as abordagens de desenvolvimento de software (tradicionais e ágeis), além do espaço e cultura. Nessa classificação foi feita uma análise qualitativa dos resultados das pesquisas. Além dos resultados e conclusões, também foram considerados as premissas e os embasamentos dos artigos selecionados, baseando-se no método de análise a Teoria Fundamentada em Dados (TFD), que é um método sistemático de investigação que enfatiza a criação de novas teorias, derivada de uma análise sistemática e rigorosa dos dados [11]. A TFD, adequada para áreas pouco exploradas ou nas quais uma nova perspectiva pode ser benéfica [12], permite que pesquisadores estudem os comportamentos pessoais e as interações sociais [11].

Quadro 1. Conceitos chaves derivados da revisão.

Autor	Ano	Método de Pesquisa		Gestão de conhecimento no desenvolvimento de software										
				Dimensão Epistemológica		Dimensão Ontológica			Abordagem		Espaço		Cultura	
		Quantitativo	Qualitativo	Explícito	Tácito	Individual	Grupo de desenv.	Organizacional	Taylorista	Ágil	Local	Distribuído	Facilitador	Barreiras
Armour	2000			X	X	X								
Rus e Lindvall	2002		X	X	X	X	X	X					X	X
Ramesh	2002		X	X	X		X							
Klint e Verhoef	2002			X	X		X	X			X	X		
Chau, Maurer e Melnik	2003		X	X	X				X	X				
Melnik E Maurer	2004		X		X		X			X	X			
Ye, Yamamoto e Kishida	2004			X	X		X		X	X		X		
Ye	2005		X	X	X		X					X		
Bahli	2005		X	X	X		X		X	X				
Feng	2006			X				X		X		X	X	
Ye	2006		X	X	X			X				X		
Kokkoniemi	2008			X	X		X			X				
Yanyan e Renzuo	2008		X		X	X								
Boden e Avram	2009		X		X			X		X		X	X	
Chen, Shie e Liang	2009	X			X		X						X	
Dakhli e Chouikha	2009			X	X			X	X	X				
Dingsøyr, Bjørnson e Shull	2009		X	X	X				X	X				
Fægri	2009		X		X		X			X	X			
Levy e Hazzan	2009		X		X		X			X	X			X
Biao-wen	2010			X	X			X						X
Faegri, Dyba e Dingsøyr	2010		X		X		X			X			X	
Huang, Shih e Hsu	2010	X	X		X		X						X	
Kavitha e Ahmed	2010			X	X		X			X	X	X		
Neves, Rosa, Correia e Castro	2011		X	X	X		X			X	X			
Abdullah e Talib	2011		X	X			X		X					
Corrigan	2012				X	X								
Dorairaj, Noble e Malik	2012		X	X	X		X			X		X		
Eloranta e Koskimies	2012			X	X		X			X	X			
Nidhra, Yanamadala, Afzal e Torka	2013		X	X	X		X					X	X	
Ryan e O'Connor	2013	X			X		X			X	X			
Kamunya e Waweru	2013			X	X		X	X				X	X	
Razzak , Ahmed e Smite	2013		X	X	X		X	X		X	X	X		
Samoilenko e Nahar	2013		X	X			X	X				X	X	X
Dingsøy e Šmite	2014			X	X		X				X	X		
Santos et al	2014	X		X	X		X	X		X				

Embora a revisão sistemática de literatura seja feita na prática por mais de uma pessoa, para minimizar a subjetividade, o fato de ter sido feito neste trabalho de maneira solo não invalida o método pois, além de oferecer um bom senso de orientação de como procurar e encontrar os artigos, ela garante a não tendenciosidade da busca em relação a pergunta formulada.

O Quadro 1 mostra a classificação dos conceitos dos artigos selecionados para a revisão de literatura.

IV . RESULTADOS

Os resultados obtidos com a revisão de literatura foram classificados de acordo com as dimensões epistemológica e ontológica, com as abordagens tradicional e ágil, além do espaço e cultura.

A. Dimensão Epistemológica

A dimensão epistemológica do modelo de Nonaka e Takeuchi [5] compreende dois tipos de conhecimento: explícito e tácito. Empresas de software têm uma proporção maior de conhecimento tácito do que empresas de outras áreas, pois dependem mais fortemente das habilidades dos indivíduos que desenvolvem os seus produtos. Assim, a gestão desse tipo conhecimento é obviamente mais importante e urgente [13]. Porém, as empresas de software devem buscar soluções que enfatizam tanto o conhecimento tácito quanto o conhecimento explícito ao planejarem iniciativas de gestão do conhecimento para apoiar a engenharia de software [14].

Técnicas que promovem a interação face a face como a programação em par, a rotação de pares ou as reuniões diárias

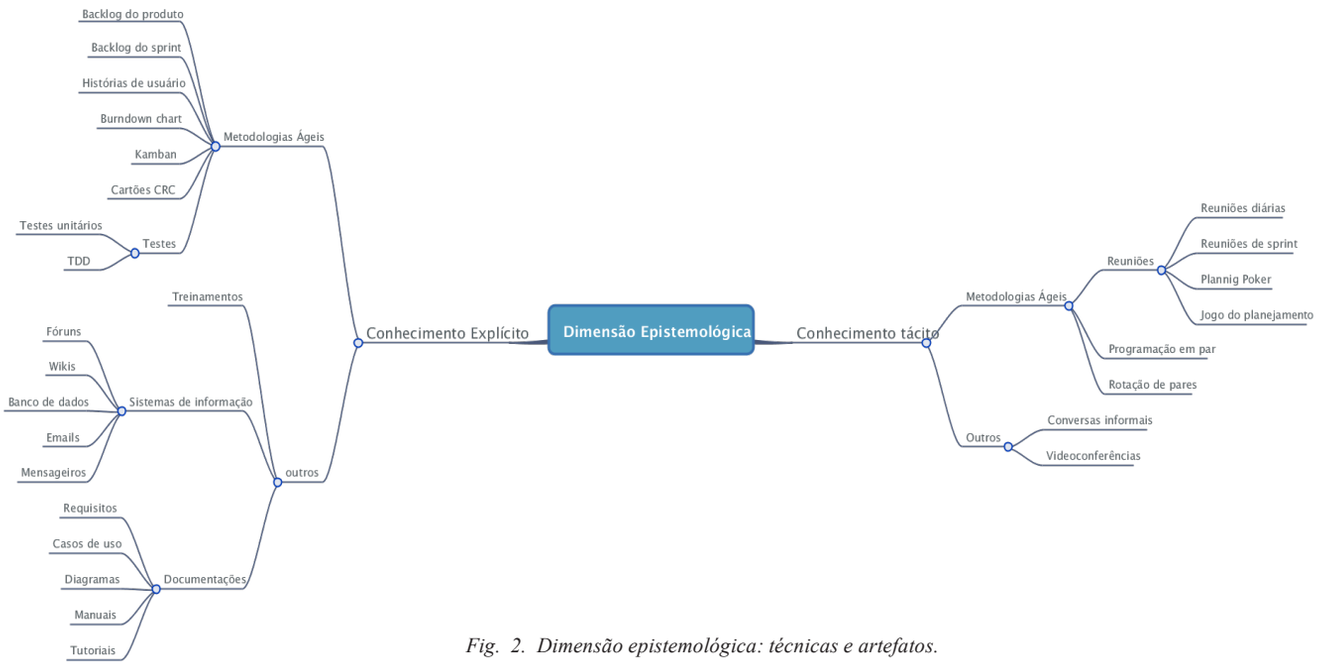

Fig. 2. Dimensão epistemológica: técnicas e artefatos.

privilegiam a socialização do conhecimento tácito. A técnica de programação em par permite que tanto o conhecimento tácito como o conhecimento explícito possam se propagar de maneira eficaz, pois este tipo de interação permite utilizar uma variedade muito maior de metáforas do que conversa por meio da tecnologia da informação [4] e oferecem a perspectiva de uma comunicação mais rica por causa da capacidade de transmitir múltiplos sinais, por exemplo, presença física, inflexão de voz e linguagem corporal [15].

Todas as ferramentas baseadas na tecnologia da informação promovem a externalização do conhecimento explícito. As wikis têm recebido uma atenção especial dos estudiosos pois, além de codificar o conhecimento de maneira explícita, permitem a colaboração em tempo real independente do tempo e do espaço. A wiki se apresenta como uma ferramenta de alto impacto na gestão do conhecimento, pois oferece oportunidades para a criação colaborativa de conhecimento na equipe e facilita a aprendizagem individual [16].

B. Dimensão Ontológica

A dimensão ontológica consiste nos três níveis de agrupamento de pessoas dentro de uma organização: individual, grupo de desenvolvimento e organizacional.

Se um projeto de desenvolvimento de software é bem-sucedido depende inteiramente do fator humano [17]. Algumas técnicas contemplativas podem melhorar o fator humano no desenvolvimento de software sob as perspectivas da flexibilidade, atenção, criatividade e a confiança em si mesmo e entre os membros da equipe [18].

Acelerar a eficiência de aprendizagem em grupo é muito mais complicado do que a prática individual [19]. A aprendizagem ou a internalização do conhecimento explícito é um processo social. Não se aprende sozinho, mas principalmente por meio do conhecimento tácito adquirido a partir de interações com os outros indivíduos. Além disso, como o desenvolvimento de software é um processo totalmente social, é importante para desenvolver a confiança organizacional e individual nas equipes e também entre as equipes e os clientes [16].

A rotação de trabalho é uma prática conhecida para construir conhecimento geral, além do nível de equipe [20] e pode contribuir para a melhoria da redundância do conhecimento [21]. Benefícios da redundância conhecimento incluem a inovação decorrente da integração de diferentes áreas do conhecimento e melhor apreciação de questões como o alinhamento da visão organizacional [21].

C. Abordagens

Nos métodos tradicionais é basicamente utilizada a documentação para a captura de conhecimento adquirido nas atividades de um ciclo de vida de projetos de software. Essas abordagens levam uma vantagem na externalização do conhecimento e assim reduzem a probabilidade de perda de conhecimento no caso de saída de algum funcionário talentoso [16]. Porém, a divisão do trabalho é, muitas vezes, rigorosamente aplicada e leva a especializações, de maneira que pessoas são considerados facilmente substituíveis [15]. Além disso, nesta abordagem não se pode capturar o conhecimento tácito e contextual [22].

As abordagens ágeis se baseiam em conhecimento tácito para compartilhar conhecimento. É atraente se basear neste tipo de conhecimento porque relaxa a exigência de documentar exaustivamente o conhecimento [23]. Além disto, as equipes ágeis são equipes multifuncionais que promovem o compartilhamento de conhecimento específico do projeto por meio da frequente interação face a face, a comunicação eficaz e colaboração do cliente [24]. No entanto, isso não resolve o problema da organização de ser dependente de seus empregados e de seus conhecimentos tácitos.

Fig. 3. Dimensão ontológica: fatores capacitadores para a criação de conhecimento organizacional.

D. Espaço e Cultura

O espaço de desenvolvimento de software normalmente é tratado como local ou distribuído. O compartilhamento físico do espaço de desenvolvimento permite uma maior socialização do conhecimento, enquanto que sistemas de informação aparecem como ferramentas de extrema importância para equipes distribuídas.

Quanto às equipes de desenvolvimento locais, as paredes do espaço de trabalho de desenvolvimento servem como meio de comunicação, constituindo um espaço informativo e colaborativo. A informação afixada nas paredes inclui, entre informações adicionais relevantes, o status das tarefas pessoais que pertencem à iteração atual e as medidas tomadas [25].

Em ambientes distribuídos, pode ser muito difícil de lidar com as barreiras espaciais e temporais, as relações legais organizacionais, nacionais e as diferentes culturas [26]. A transferência de conhecimento nesse caso se dá por meio do uso eficaz da comunicação mediada pela tecnologia, tais como ferramentas de gestão do conhecimento, a wiki e a videoconferência [24].

As equipes auto organizadas e a autonomia dos membros estão no cerne do sucesso da implementação das metodologias ágeis e da gestão do conhecimento. Este aspecto cultural indica que nem todo grupo de desenvolvimento pode trabalhar de forma orientada à criação de conhecimento e à agilidade de desenvolvimento.

Essa nova perspectiva exige que os funcionários aprendam novos conhecimentos e novas habilidades, o que pode mudar as posições estruturais internas da organização, aumentar a intensidade de trabalho e fazer com que os funcionários sintam seus interesses ameaçados [27].

Tanto a gestão do conhecimento como o desenvolvimento ágil de software são dois processos organizacionais que enfrentam barreiras comuns quando introduzidos e aplicados. Ambas as disciplinas lidam com a cultura organizacional e com a gestão da mudança e são melhores analisadas e implementadas quando feitas de maneira simultânea [25].

V. TÉCNICAS E FATORES QUE PROMOVEM A CRIAÇÃO DE CONHECIMENTO ORGANIZACIONAL

Por meio da revisão de literatura e do referencial teórico foi possível identificar algumas técnicas provenientes das metodologias ágeis e os fatores capacitadores para a criação de conhecimento organizacional. De acordo com os materiais analisados foram feitas diferentes classificações.

Os resultados sugerem que tanto as dicotomias de conhecimento (explícito e tácito) quanto de desenvolvimento de software (tradicional e ágil) são muito importantes para a gestão do conhecimento.

A Figura 2 mostra as técnicas e os artefatos classificados de acordo com a dimensão epistemológica. Já a Figura 3 apresenta os fatores capacitadores para a criação de conhecimento organizacional classificados de acordo com a dimensão ontológica.

A Figura 4 sintetiza as principais técnicas, artefatos e fatores capacitadores classificados de acordo com o modelo SECI. Algumas técnicas favorecem mais de uma forma de conversão de conhecimento.

As reuniões diárias, as reuniões de *sprint*, o *Planning Poker* ou o Jogo do Planejamento têm sua principal relevância na socialização pois essas interações sociais face a face possibilita a transmissão de múltiplos sinais melhorando a capacidade de reestruturação dos modelos mentais. Também se destacam na conversão do conhecimento tácito em conhecimento explícito quando são incentivados os usos de analogias e metáforas, o que também possibilita externalização desses modelos mentais.

A técnica de programação em par recebeu a atenção de diversos estudiosos. Quando duas pessoas atuam junto em um mesmo problema elas trazem diferentes visões de mundo, o que aumenta a variedade de requisitos da equipe em suprir a exigência do ambiente por meio da redundância de conhecimento. A rotação de pares, além dos mesmos benefícios da programação em par, ainda propicia a elevação do conhecimento do nível individual até o nível organizacional da dimensão ontológica.

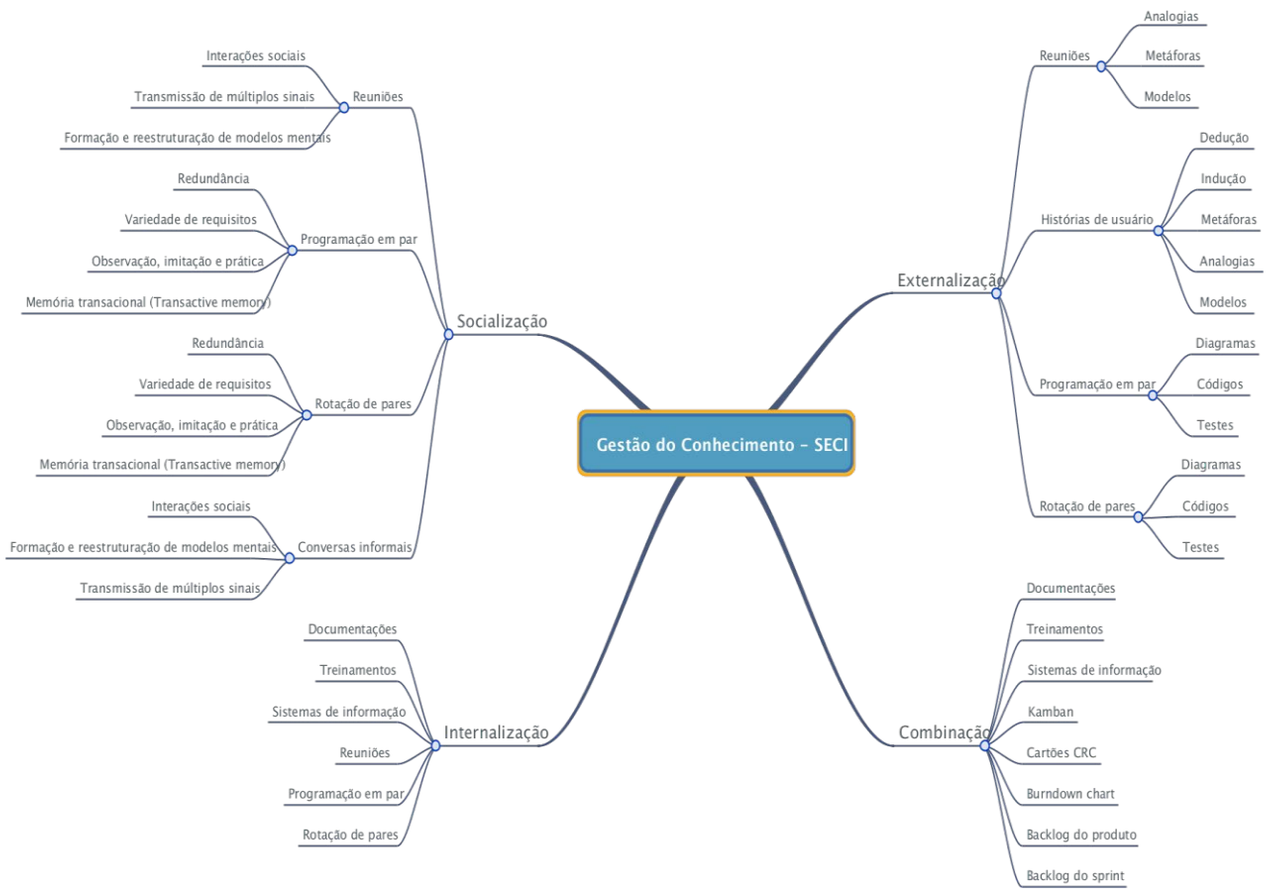

Fig. 4. Gestão do conhecimento no desenvolvimento de software, de acordo como o modelo SECI.

VI. CONCLUSÕES

Este trabalho apresentou alguns dos principais conceitos de criação de conhecimento na empresa e as abordagens metodológicas de desenvolvimento de software. Em seguida foi feito uma revisão sistemática de literatura, com auxílio do *framework* PRISMA, para levantar o estado da arte da gestão do conhecimento no desenvolvimento de software. A análise dos resultados baseou-se na Teoria Fundamentada em Dados.

Como resultados, foram apresentados os principais conceitos e práticas que os estudiosos têm encontrado, reunidos sobre a classificação da dimensão epistemológica (conhecimento tácito e conhecimento explícito) e da dimensão ontológica (individual, grupo de desenvolvimento e organizacional), além das abordagens de desenvolvimento de software (tradicionais e ágeis) e do espaço e da cultura. Além disso, na tentativa de responder às questões de pesquisa que nortearam esta revisão de literatura, foram elaborados três mapas (Figuras 2, 3 e 4) que codificaram os conceitos envolvidos tendo como base as duas dimensões (epistemológica e ontológica) e o modelo SECI de criação do conhecimento proposto por Nonaka e Takeuchi [5].

A grande quantidade de artigos que se utilizaram de métodos qualitativos (57%) ou que não foram possíveis identificar o método utilizado (34%) mostra que o tema ainda foi pouco explorado e evidencia a necessidade de uma classificação das técnicas das metodologias ágeis e fatores capacitadores para a criação do conhecimento.

Com este trabalho, foi possível verificar a existência de algumas lacunas na gestão do conhecimento no desenvolvimento de software. Assim, aproveitando-se de algumas dessas lacunas, questões que poderão ser abordadas em trabalhos futuros incluem:

- Como as equipes de desenvolvimento de software gerem o conhecimento na prática?
- Como as equipes ágeis criam conhecimento no desenvolvimento de software?
- Qual o melhor processo gerencial para a criação de conhecimento no desenvolvimento de software?

REFERÊNCIAS

[1] I. Nonaka. "A empresa criadora de conhecimento," in: *Harvard Business Review. Gestão do conhecimento – on Knowledge management*. Rio de Janeiro: Campus, 2000.

[2] I. Chiavenato. *Introdução à Teoria Geral da Administração*. Rio de Janeiro: Campus, 2003.

[3] P. R. Motta. *Transformação organizacional: a teoria e a prática de inovar*. 5 ed., Rio de Janeiro: Ed. Qualitymark, 2001.

[4] S. Ryan, R. V. O'Connor. "Acquiring and sharing tacit knowledge in software development teams: An empirical study," *Information and Software Technology*, vol. 55, no. 9, pp. 1614-1624, 2013.

[5] I. Nonaka, H. Takeuchi. *Criação de conhecimento na empresa: Como as empresas japonesas geram a dinâmica da inovação*. Rio de Janeiro: Campus, 1997.

[6] I. Sommerville. *Engenharia de Sofware*. São Paulo: Pearson Addison-Wesley, 2007.

[7] E. Ries. *A Startup Enxuta: como os empreendedores atuais utilizam a inovação contínua para criar empresas extremamente bem-sucedidas*. São Paulo: Lua de Papel, 2012.

[8] J. Wainer. "Métodos de pesquisa quantitativa e qualitativa para a ciência da computação," in: T. Kowaltowski, K. Breitman. (Org.). *Atualização em informática 2007*. Rio de Janeiro: Sociedade Brasileira de Computação e Editora PUC-Rio, 2007, p. 221-262.

[9] N. D. Macedo. *Iniciação à pesquisa bibliográfica: guia do estudante para a fundamentação do trabalho de pesquisa*. São Paulo: Edições Loyola, 1994.

[10] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman. "Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement," *PLoS Medicine*, vol. 339, no. 7, pp. 6, 21, 2009.

[11] B. G. Glaser, A. L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine, Chicago: Sociology Press, 1967.

[12] R. S. Schreiber, P. N. Stern. *Using Grounded Theory in Nursing*. Broadway, New York: Springer Publishing, 2001.

[13] R. K. Kavitha, M. S. I. Ahmed. "A Knowledge Management Framework for Agile Software Development Teams," in *Proceedings of the 2011 International Conference on Process Automation, Control and Computing,* Coimbatore, India, 2011, pp. 1–5.

[14] T. Dingsøyr, F. O. Bjornson, F. Shull. "What Do We Know about Knowledge Management?" *IEEE Software*, vol.26, no.3, pp.2–5, 2009.

[15] G. Melnik, F. Maurer. "Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing," in *Proceedings of the Agile Development Conference*, Salt Lake City, Utah, 2004, pp. 21-31.

[16] T. Chau, F. Maurer, G. Melnik. "Knowledge Sharing: Agile Methods vs. Tayloristic Methods," in *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Linz, Áustria, 2003, pp.302–307.

[17] Z. Yanyan, X. Renzuo. "The Basic Research of Human Factor Analysis Based on Knowledge in Software Engineering," in *Proceedings of the International Conference on Computer Science and Software Engineering*, Wuhan, China, 2008, vol. 5, pp. 1302–1305.

[18] J. M. Corrigan. "Augmented Intelligence-The new AI-Unleashing Human Capabilities in Knowledge Work," in *Proceedings of the International Conference on Software Engineering*, Zurich, Switzerland, 2012, pp.1285-1288.

[19] H. C. Huang, H. Y. Shih, S. C. Hsu. "Team Structure to Accelerate Knowledge Diffusion: A Case Study in Computer Software Developer," in *Proceedings of the IEEE International Conference on Management of Innovation and Technology*, Singapore, Singapore, 2010, pp.928–933.

[20] T. Dingsoyr, D. Smite. "Managing Knowledge in Global Software Development Projects," *IT Professional*, vol.16, no. 1, pp.22-29, 2014.

[21] T. E. Fægri, T. Dybå, T. Dingsøyr. "Introducing knowledge redundancy practice in software development: Experiences with job rotation in support work," *Information and Software Technology*, vol. 52, no. 10, pp. 1118–1132, 2010.

[22] Y. Ye. "Supporting Software Development as Knowledge-Intensive and Collaborative Activity," in *Proceedings of the International Workshop on Interdisciplinary Software Engineering Research*, Shanghai, China, 2006, pp. 15-22.

[23] I. Rus, M. Lindvall. "Knowledge Management in Software Engineering," *IEEE software*, vol. 19, no. 3, pp. 26–38, 2002.

[24] S. Dorairaj, J. Noble, P. Malik. "Knowledge Management in Distributed Agile Software Development," in *Proceedings of the Agile Conference,* Dallas, Texas, 2012, pp. 64-73.

[25] M. Levy, O. Hazzan. "Knowledge Management in Practice: The Case of Agile Software Development," in *Proceedings of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, Vancouver, Canada, 2009, pp. 60–65.

[26] A. Boden, G. Avram. "Bridging Knowledge Distribution-The Role of Knowledge Brokers in Distributed Software Development Teams," in *Proceedings of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, Vancouver, Canada, 2009, pp.8–11.

[27] L. Biao-Wen. "The Analysis of Obstacles and Solutions for Software Enterprises to Implement Knowledge Management," in *Proceedings of the 2nd IEEE International Conference on Information Management and Engineering*, Chengdu, China, 2010, pp. 211–214.

[28] R. Abdullah, A. M. Talib. "Knowledge Management System Model in Enhancing Knowledge Facilitation of Software Process Improvement for Software House Organization," in *Proceedings of the International Conference on Information Retrieval and Knowledge Management*, 2012.

[29] P. G. Armour. "The Five Orders of Ignorance," *Communications of the ACM*, vol. 43, no. 10, pp. 17–20, 2000.

[30] B. Bahli, E. S. A. Zeid. "The Role of Knowledge Creation in Adopting Extreme Programming Model: An Empirical Study," in *Proceedings of the ITI 3rd International Conference on Information and Communications Technology*, 2005.

[31] D. N. Chen, Y. J. Shie, T. P. Liang. "The Impact of Knowledge Diversity on Software Project Team's Performance," in *Proceedings of the 11th International Conference on Electronic Commerce*, 2009, p. 222.

[32] S. Dakhli, M. B. Chouikha. "The Knowledge-Gap Reduction in Software Engineering," in *Proceedings of the Third International Conference on Research Challenges in Information Science*, 2009.

[33] V. P. Eloranta, K. Koskimies. "Aligning Architecture Knowledge Management with Scrum," in *Proceedings of the WICSA/ECSA*, 2012, pp. 112–115.

[34] J. Feng. "A Knowledge Management Maturity Model and Application," in *Proceedings of the Technology Management for the Global Future Conference*, 2006.

[35] T. E. Fægri. "Improving General Knowledge in Agile Software Organizations: Experiences with Job Rotation in Customer Support," in *Proceedings of the Agile Conference*, 2009, pp. 49–56.

[36] S. Kamunya, M. Waweru. "Utilization of Knowledge Management Tools in Software Development," in *Proceedings of the IST-Africa Conference and Exhibition*, 2013.

[37] P. Klint, C. Verhoef. "Enabling the creation of knowledge about software assets," *Data & Knowledge Engineering*, vol. 41, no. 2-3, pp. 141–158, 2002.

[38] J. K. Kokkoniemi. "Gathering Experience Knowledge from Iterative Software Development Processes," in *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, 2008.

[39] F. T. Neves, A. M. R. Correia. "Knowledge Creation and Sharing in Software Development Teams Using Agile Methodologies: Key Insights Affecting their Adoption," in *Proceedings of the 6th Iberian Conference on Information Systems and Technologies*, 2011.

[40] S. Nidhra, M. Yanamadala, W.Afzal, R. Torkar. "Knowledge transfer challenges and mitigation strategies in global software development – A systematic literature review and industrial validation," *International Journal of Information Management*, vol. 33, no. 2, pp. 333–355, 2013.

[41] B. Ramesh. "Process Knowledge Management with Traceability," *IEEE Software*, vol. 19, no. 3, pp. 50–52, 2002.

[42] M. A. Razzak, R. Ahmed, D. Mite. "Spatial Knowledge Creation and Sharing Activities in a Distributed Agile Project," in *Proceedings of the IEEE 8th International Conference on Global Software Engineering Workshops*, 2013, pp. 24–30.

[43] N. Samoilenko, N. Nahar. "Knowledge Sharing and Application in Complex Software and Systems Development in Globally Distributed High-Tech Organizations Using Suitable IT Tools," in *Proceedings of the Technology Management in the IT-Driven Services (PICMET)*, pp. 1280–1294.

[44] V. Santos, A. Goldman, H. Filho, D. Martins, M. Cortés. "The Influence of Organizational Factors on Inter-team Knowledge Sharing Effectiveness in Agile Environments," in *Proceedings of the 47th Hawaii International Conference on System Sciences*, 2014, pp. 4729–4738.

[45] K. Schwaber, M. Beedle. *Agile Software Development with Scrum*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[46] Y. Ye, Y. Yamamoto, K. Kishida. "Dynamic Community: A New Conceptual Framework for Supporting Knowledge Collaboration in Software Development," in *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, 2004.