Scrumie: Jogo orientado a agentes para ensino de Scrum

Bruna Costa Cons[1], Leonardo Lima Marinho[1], Suelen Regina C. dos Santos[1], Marcelo Schots[2], Vera Maria B. Werneck[1,2]

[1]Programa de Mestrado em Ciências Computacionais, Universidade do Estado do Rio de Janeiro (UERJ), Brasil

[2]Departamento de Informática e Ciência da Computação, Universidade do Estado do Rio de Janeiro (UERJ), Brasil.

{suelen_cordeiro,leonardomarinho_10}@hotmail.com,
brunacons94@gmail.com, {schots, vera}@ime.uerj.br

Abstract. *The use of agile methods has become essential in software development at the present time. Among the existing methods, Scrum is one of the major ones, and is used to manage processes in companies, even outside of the scope of software systems development. Considering the relevance of this subject and the success usually obtained in learning through educational games, it was proposed Scrumie, an adaptation of 2TScrum (serious board game). Both games seek to teach Scrum project management. Scrumie added intelligence in multiagent architecture being developed with Agile Passi methodology. This article contains a proposal, modeling and implementation of the Scrumie game.*

Resumo. *O uso de métodos ágeis se tornou imprescindível no desenvolvimento de software na atualidade. Dentre os métodos existentes, o Scrum é um dos principais utilizados para gerenciar processos em empresas, mesmo fora do escopo de desenvolvimento de sistemas. Considerando a relevância deste assunto e o sucesso geralmente obtido no aprendizado por meio de jogos educacionais, foi proposto Scrumie, uma adaptação do 2TScrum (serious game de tabuleiro). Ambos os jogos buscam ensinar o gerenciamento de projetos Scrum. Scrumie adicionou inteligência numa arquitetura multiagentes, sendo desenvolvido com a metodologia Agile Passi. Este artigo apresenta a proposta, modelagem e implementação do jogo Scrumie.*

1. Introdução

O modelo em cascata (*waterfall*), caracterizado por exigir especificações do sistema bem definidas e processos de trabalho longos e sequenciais [Soares 2004] tem sido considerado um padrão para o desenvolvimento de *software*. Com a crescente complexidade dos sistemas e uma maior busca por eficiência nas empresas, este modelo começou a ser considerado inadequado em projetos de pequeno ou médio porte, ou que possuam requisitos dinâmicos. Portanto, surgiu a necessidade de incluir conceitos baseados no manifesto ágil [Beck 2001], com etapas reduzidas e mais curtas, nas quais há entrega incremental para o cliente por meio de versões do sistema.

Os métodos ágeis possuem maior foco na entrega de código do que na documentação, havendo maior integração da equipe, interação com o cliente e aceitação para mudanças nas funcionalidades do sistema. Como o *Scrum* é o *framework* ágil mais utilizado no mercado atualmente [Sille 2013], considerou-se relevante buscar a disseminação e aprendizado de seus conceitos.

Entre as formas de ensino mais eficazes, pode-se citar o uso de jogos educativos, pois, segundo Grando (2001), estes proporcionam uma participação ativa de aprendizado pelo jogador, de forma que conceitos de difícil compreensão possam ser aprendidos e sedimentados. Considerando também que os jogos possam permitir a exploração do conhecimento de forma prática e divertida, através do aspecto lúdico [Legey 2012], foi proposto um jogo de tabuleiro com o intuito de ensinar a prática de gerenciamento de projetos utilizando *Scrum*, o 2TScrum [Brito e Vieira 2017] que obteve bons resultados em relação a sua avaliação.

Neste sentido, o presente trabalho propõe jogo Scrumie, uma adaptação do 2TScrum incorporando o uso de agentes em sua implementação. O jogo 2TScrum tem como limitação separar as atividades entre os participantes da equipe *Scrum*, o que pode ser melhor abordado utilizando a arquitetura multiagentes. No contexto de ensino-aprendizagem, o emprego dos agentes, tem como propósito gerar maior qualidade e flexibilidade, tanto do ponto de vista do aluno quanto do professor. Além disso, a adaptação do jogo usando agentes pôde torná-lo mais acessível e prático, por não exigir o uso de um tabuleiro e peças físicas. O uso de agentes viabilizou a inclusão de um conceito inteligente, em que um agente provê auxílio a jogadores que precisem de dicas do jogo.

Além da introdução, este artigo é composto por mais 5 seções. A seção 2 aborda o método ágil *Scrum*, como este funciona, como a equipe é formada e quais são os seus eventos. A seção 3 apresenta o jogo Scrumie, definindo o modo de funcionamento e suas regras. A seção 4 descreve a metodologia utilizada no processo de desenvolvimento deste trabalho, o Agile PASSI, além de listar as etapas realizadas neste contexto, mostrando em seguida a modelagem realizada no desenvolvimento do jogo, enquanto a seção 5 relata a implementação do jogo. Por fim, a seção 6 apresenta as considerações finais.

2. *Scrum*

O *Scrum* é um *framework* de gestão e planejamento de projetos que segue os ideais ágeis. Ele recomenda a auto-organização da equipe, envolvimento constante do cliente no projeto, prazos de entrega curtos e flexibilidade de adaptação a mudanças.

As atividades básicas em um processo *Scrum* são: pré-planejamento, desenvolvimento e pós-planejamento. No pré-planejamento, as funcionalidades do sistema são relatadas em um documento chamado *backlog*. Cada requisito possui uma prioridade e um custo para ser implementado, e isto também é definido nesta fase, além dos riscos do projeto, ferramentas que serão utilizadas, definição da equipe e proposta de uma arquitetura de desenvolvimento. A fase de desenvolvimento é iterativa, na qual ocorrem as *sprints* (unidades de tempo de um mês ou menos que compartimentalizam o trabalho de um incremento) e onde se observa e controla possíveis mudanças nas variáveis já estabelecidas previamente. Já a fase de pós-planejamento engloba reuniões

realizadas para avaliar o progresso do projeto, testes finais e integração do sistema [Devmedia 2008].

A equipe *Scrum* é formada pelos seguintes papéis: *Product Owner*, o responsável pelo gerenciamento do *backlog* do produto; o *Scrum Master*, que garante a compreensão do *Scrum* pela equipe, além de resolver problemas que impedem o sucesso da *sprint*; e a equipe de desenvolvimento, responsável pela criação dos incrementos utilizáveis do produto.

O *Scrum* define a realização de reuniões para manter a equipe integrada junto ao cliente. A reunião de planejamento da *sprint* possui o intuito de definir os itens a serem considerados na *sprint* atual. A reunião diária é uma reunião curta, realizada para acompanhar as atividades atuais e planejar as atividades do dia. Há também duas reuniões mais informais, a revisão e a retrospectiva da *sprint*, com o objetivo de apresentar e avaliar os resultados obtidos ao fim da *sprint*. A partir das reuniões, artefatos são gerados, como o *backlog* do produto – uma lista de requisitos do sistema – e o *backlog* da *sprint*, que relata as funcionalidades selecionadas do *backlog* do produto que precisarão ser desenvolvidas na *sprint* atual [Brito e Vieira 2017].

3. O jogo Scrumie

O Scrumie possui mecanismo de jogo semelhante ao 2TScrum, no qual um jogador deve seguir um fluxo que o permite gerenciar projetos de *software*, aplicando seus conhecimentos teóricos sobre *Scrum* e estimulando novos conhecimentos. Para isto, o jogo simula situações cotidianas de desenvolvimento de *software*, motivando o jogador a ter autonomia para tomar decisões a respeito de acontecimentos no projeto que devem ser gerenciados [Brito e Vieira 2017].

Com o objetivo de instigar o aprendizado no jogo, o jogador precisa finalizar o desenvolvimento do projeto dentro do prazo e orçamento estabelecidos pelo cliente no início da partida. No decorrer do jogo, o desempenho do jogador será avaliado para que dicas possam ser dadas a ele, auxiliando-o no gerenciamento do projeto.

Igualmente ao 2TScrum, o Scrumie apresenta os mesmos elementos que contribuem para a interação do jogador com o jogo: a carta do cliente, as cartas de *backlog*, a carta de validação do *backlog*, as cartas com perfil do desenvolvedor, as cartas surpresa e as cartas de eventos [Brito e Vieira 2017].

Os componentes que integram o jogo Scrumie são representados por quase todos os papéis da equipe *Scrum*, sendo o jogador responsável pelo papel do gerente de projeto e do *Product Owner;* o agente Cliente responsável pelo papel do cliente; e o agente *Scrum Master* responsável pelo papel do *Scrum Master*. No entanto, novas adaptações foram feitas em relação aos papéis dos agentes Cliente e *Scrum Master*. O agente Cliente é o encarregado de criar cartas do cliente, cartas de *backlog* e cartas de validação do *backlog*, de acordo com o cenário elaborado para a partida. Já o *Scrum Master* é o encarregado de apresentar dicas ao jogador mediante o seu desempenho, e selecionar aleatoriamente cartas das respectivas reuniões presentes no *Scrum*.

Dois novos componentes foram adicionados com o objetivo de auxiliar a dinâmica do jogo. Um deles é o agente Gerenciador de Progresso, responsável pela avaliação de desempenho do jogador, que mantém comunicação com o agente *Scrum Master* a respeito do desempenho apresentado e gerenciando a pontuação do jogador ao

longo do jogo. Há também o agente Executor de Regras, responsável por tratar as violações de regras.

3.1. Etapas do jogo

A interação entre o jogador e o jogo Scrumie, nessa primeira versão é realizada por meio de uma interface de linha de comando sob a forma de sucessivas linhas de texto. Baseado em Brito e Vieira (2017), a seguir são detalhadas as etapas do fluxo principal que o jogador percorre durante uma partida.

Antes de dar início ao jogo, o agente Cliente apresenta para o jogador a seguinte narrativa [Brito e Vieira 2017]: “Um novo projeto chega à empresa e você é alocado para gerenciá-lo utilizando o *framework Scrum.* Este projeto se trata de um sistema *web* para uma biblioteca, com um orçamento e prazo fixos que você deve cumprir, mas no decorrer do tempo há diversos acontecimentos que você deve gerenciar. Além desses acontecimentos, durante a construção do produto, o seu cliente acaba entendendo melhor o que necessita e mudanças podem acontecer. Você deve manter o projeto no prazo e no orçamento estabelecidos, mas será que é possível? Você aceita esse desafio?”.

Após a leitura da narrativa, o jogador inicia a partida inserindo seu nome no sistema. Em seguida, o agente Cliente apresenta a carta do cliente, informando o produto desejado, o orçamento e o prazo para o projeto. Posteriormente, o jogador segue para a criação do *backlog* do produto, onde o agente Cliente apresenta trinta e seis itens do *backlog*, exibidos com um identificador, descrição do item e estimativa de tempo para o desenvolvimento. O jogador deve escolher apenas os itens esclarecidos na carta do cliente.

Assim que o jogador montar o *backlog* do produto, o agente Gerenciador de Progresso passa a calcular a pontuação do jogador com base no prazo e no custo divulgados nos itens selecionados. Este agente mantém o gerenciamento da pontuação do jogador ao longo do jogo, a partir dos acontecimentos e modificações presentes nas cartas das próximas etapas.

O jogador segue para a reunião com o cliente e, neste momento, o agente Cliente passa a validar os itens do *backlog* do produto segundo a carta de validação do *backlog* concedida por ele. O agente ajusta o *backlog* do produto de acordo com essa carta, incluindo os itens dela que não estão no *backlog* do produto e descartando os itens que não estão presentes nela. Se o agente verificar que o jogador acertou menos de oito itens no *backlog* do produto, ele aplica uma penalidade ao jogador. Caso contrário, oferece uma recompensa.

Depois de validado o *backlog* do produto, o jogador deve escolher três perfis de desenvolvedores para sua equipe de desenvolvimento. No caso, é exibido para o jogador cartas com os perfis dos desenvolvedores, informando a classificação, o custo monetário por *sprint* e o bônus de cada um. O agente Executor de Regras é comunicado para verificar a quantidade de desenvolvedores escolhidos pelo jogador.

Com a equipe definida, o jogador faz o planejamento da *sprint*, onde cria o *backlog* da *sprint* escolhendo no máximo sete itens do *backlog* do produto. O agente Executor de Regras é comunicado para verificar a quantidade de itens escolhidos para o *backlog* da *sprint*. Com a verificação feita e aprovada, assim que o *backlog* da *sprint* é

criado, o agente Gerenciador de Progresso atualiza o prazo do projeto, decrementando o total de tempo estimado para os itens do *backlog* da *sprint*.

Ainda no planejamento, o agente *Scrum Master* sorteia aleatoriamente duas cartas de planejamento da *sprint* e as apresenta para o jogador. Após a leitura das cartas, o jogador deve tomar decisões sobre os acontecimentos descritos nelas, considerando o prazo e o custo. Com as escolhas feitas, o agente Gerenciador de Progresso avalia o desempenho do jogador com base na média de desempenho aceitável de um jogador na partida. Caso este verifique que a atuação do jogador atual está ruim, comunica isso ao agente *Scrum Master*, que prontamente apresenta dicas ao jogador em relação às cartas de planejamento da *sprint*. Depois, o agente *Scrum Master* sorteia aleatoriamente duas cartas surpresa, aplicando suas alterações e apresentando as cartas para o jogador.

Finalizada a etapa do planejamento, segue-se para a etapa de desenvolvimento, na qual o agente *Scrum Master* sorteia aleatoriamente duas cartas de reunião diária e apresenta as cartas para o jogador. Novamente, o jogador deve considerar as informações nas cartas para tomar decisões. Feito isso, o Gerenciador de Progresso faz mais uma avaliação de desempenho, repetindo o mesmo processo descrito acima.

Ao ter gerenciado os acontecimentos presentes na reunião diária, o jogador segue para a revisão da *sprint*, onde o agente *Scrum Master* sorteia aleatoriamente uma carta de revisão da *sprint* e a apresenta para o jogador. Esta carta mostra dificuldades e soluções identificadas pelo Cliente ao expor o incremento do sistema na reunião. Baseado na informação recebida, o jogador deve realizar uma escolha. Em seguida, o Gerenciador de Progresso atua novamente da forma já descrita.

Depois da reunião de revisão, o jogador segue para a retrospectiva da *sprint*, onde o agente *Scrum Master* sorteia aleatoriamente uma carta de retrospectiva da *sprint* e a exibe para o jogador. Esta carta mostra acontecimentos positivos ou negativos da *sprint* que devem ser lidados pela equipe na próxima *sprint*. Da mesma forma que antes, o jogador deve tomar uma decisão e o agente Gerenciador de Progresso irá avaliá-la, tomando as medidas necessárias.

Finalmente, o agente Cliente é notificado para validar se todos os itens do *backlog* do produto foram desenvolvidos. Caso sobrem itens no *backlog* do produto, o agente Cliente informa ao agente Executor de Regras que o jogo não pode ser finalizado. Este comunica ao jogador sobre o retorno para o planejamento da *sprint*, pois o produto não foi concluído, iniciando mais uma *sprint*. Caso contrário, o Cliente informa para o jogador que o jogo foi finalizado. Em seguida, são apresentadas ao jogador sua pontuação final e sua posição no ranking. O ranking indica um *feedback* de desempenho para o jogador, levando em consideração o prazo e o orçamento resultantes no final do projeto em relação aos outros jogadores. Por último, o agente Gerenciador de Progresso atualiza a pontuação média de desempenho de um jogador com a pontuação final da partida.

4. Modelagem do sistema

O Agile PASSI é a metodologia ágil orientada a agentes utilizada no desenvolvimento do jogo Scrumie. Ela surgiu a partir da metodologia PASSI, criada inicialmente para desenvolver um sistema robótico, incluindo processos ágeis oriundos do método *Extreme Programming*, onde há a liberação de versões iterativas do sistema [Chella 2006]

É uma metodologia orientada a código, que valoriza mais o desenvolvimento do código do que a criação da documentação do mesmo. Há a identificação de agentes como um conjunto de funcionalidades expressas em casos de uso e a utilização de ontologias para descrever o domínio do sistema [Chella 2004]. A maioria das etapas de desenvolvimento são auxiliadas por automatizações e reutilização de código, por meio do *framework* disponibilizado, *Agent Factory*. Ele realiza um trabalho de engenharia reversa, construindo automaticamente um esboço das classes dos agentes a partir dos diagramas que os identificam [Chella 2006] ,[Cons 2018].

Originalmente, a metodologia recomenda a reutilização de padrões no código através da ferramenta *Agent Factory*, além da criação de ontologias e de histórias de usuário ao planejar as funcionalidades do sistema. No entanto, estas especificidades não foram utilizadas no desenvolvimento do trabalho.

A princípio, foi criado um fluxo principal do jogo, como uma lista de atividades, posteriormente utilizado para a criação dos diagramas. Algumas adaptações foram realizadas em relação às recomendações do Agile PASSI: manteve-se a sugestão de criação do diagrama de caso de uso, enquanto os diagramas de identificação de agentes e o diagrama de identificação de papéis foram incluídos. Os estereótipos usados vêm do padrão UML [Larman 2002].

Nesta etapa também foi elaborada uma lista contendo as principais regras de negócio do jogo. Isso porque estas regras detalham as funcionalidades particulares do *software* presentes nos diagramas, para satisfazer o cliente e o objetivo do negócio [Dallavalle e Cazarini 2000].

4.1. Diagrama de Caso de Uso

Este diagrama auxilia no levantamento dos requisitos funcionais do sistema Scrumie. A Figura 1 apresenta parte do diagrama que descreve um conjunto de funcionalidades do jogo.

Assim como o diagrama exemplificado na Figura 1, outros diagramas de caso de uso também foram elaborados para estabelecerem restrições que permitem o funcionamento correto do jogo.

4.2. Diagrama de Identificação de Agentes

De acordo com Henderson-Sellers (2005), a elaboração do diagrama de identificação de agentes começa a partir do diagrama de casos de uso já produzido. Isto é, ele é formado com base no agrupamento de um ou mais casos de uso em pacotes estereotipados. E, ao fazer isso, cada pacote passa a definir as funcionalidades de um agente específico. Além disso, os nomes dos pacotes são os nomes dos agentes representados.

Uma das principais características deste diagrama é o seu relacionamento, onde relacionamentos entre casos de uso de diferentes agentes são estereotipados como “communicate”, enquanto relacionamentos entre casos de uso do mesmo agente são modelados usando “include” e “extend” [Henderson-Sellers 2005]. Foram identificados 4 agentes: (i) Cliente, (ii) Executor de Regras, (iii) Scrum Master e (iv) Gerenciador de Progresso. A Figura 2 representa um exemplo deste diagrama no cenário do jogo Scrumie para os agentes Cliente e Executor de Regras.

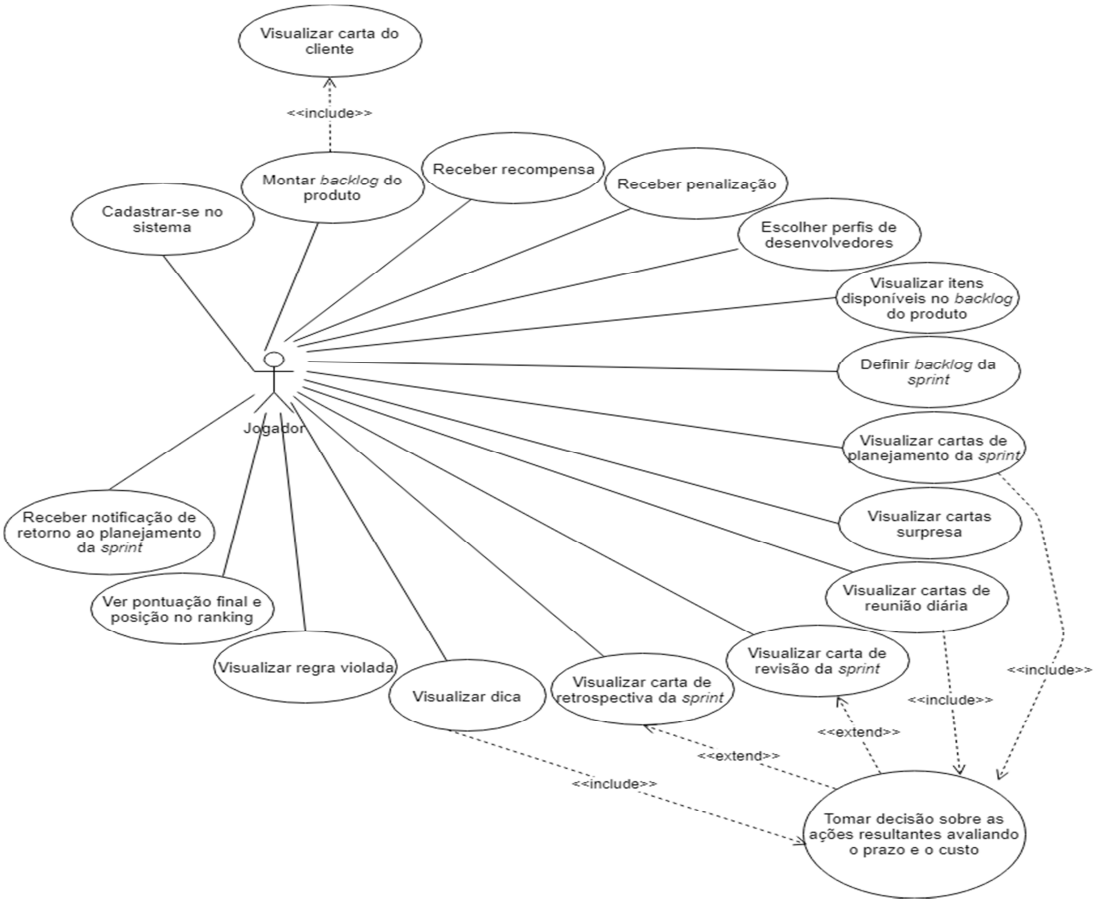

Figura 1. Parte do diagrama de caso de uso voltada às atividades do jogador

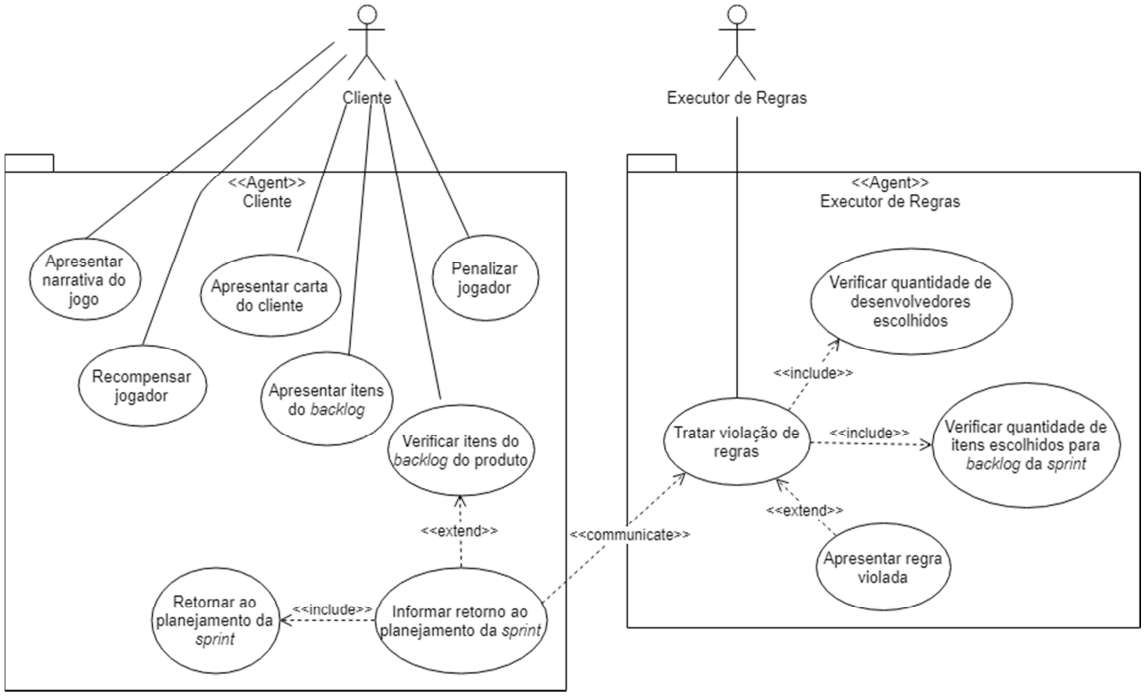

Figura 2. Parte do diagrama de identificação de agentes

4.3. Diagrama de Identificação de Papéis

Esse é um diagrama de sequência UML, onde cada objeto é utilizado para simbolizar o papel do agente, cuja sintaxe é <nome-do-papel>: <nome-do-agente>. Um agente pode desempenhar papéis distintos dentro do mesmo diagrama [Henderson-Sellers 2005].

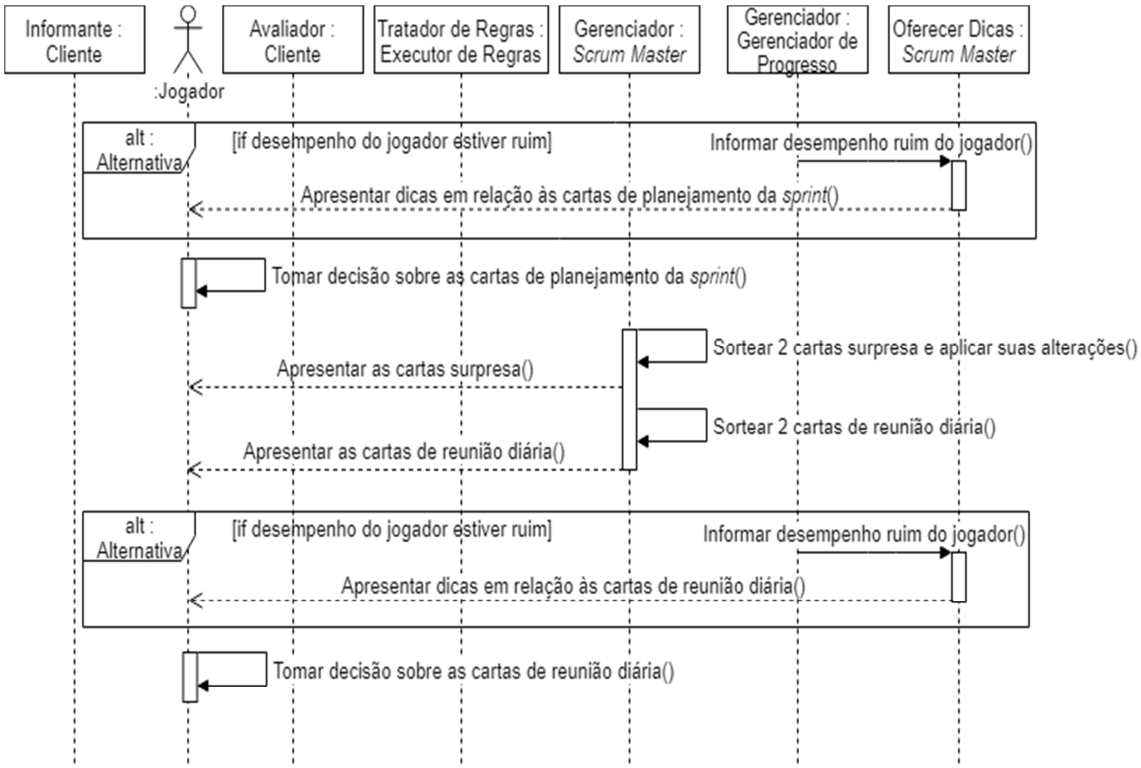

Figura 3. Diagrama de identificação de papéis

A Figura 3 mostra um exemplo do diagrama, que explora a troca de mensagens ao longo de parte do cenário do jogo Scrumie, envolvendo comunicação entre agentes e seus papéis. A elaboração deste diagrama também parte do princípio de que o diagrama de identificação de agentes já foi desenvolvido.

5. Implementação do jogo

Após a modelagem, deu-se início à implementação do jogo. Nesta etapa, primeiramente foram escolhidas as tecnologias utilizadas. Em seguida, foi definida a organização dos componentes do sistema, como arquivos de configuração e agentes, a partir das informações sobre o jogo 2TScrum e da modelagem. Com estas informações estabelecidas, o sistema foi desenvolvido.

Ao decidir a partir de quais tecnologias o jogo seria construído, levou-se em conta a necessidade da velocidade de desenvolvimento e da capacidade da tecnologia em representar bem a arquitetura multiagentes. Neste ponto, optou-se por não usar um *framework*, uma vez que isso demandaria um período de estudos e compreensão do mesmo, podendo atrasar a entrega da aplicação, e indo além da simplicidade proposta na modelagem.

Com essas ideais em consideração, optou-se pela utilização da plataforma Node.js[1], que é capaz de executar a linguagem de programação JavaScript em aplicações *desktop*. Dessa forma, o código foi escrito em TypeScript[2], uma linguagem que é compilada para JavaScript e que fornece a versão mais recente do mesmo, agregando funcionalidades, como um melhor suporte à tipagem de dados e à orientação a objetos. Como o JavaScript é a linguagem de programação utilizada em todos os navegadores modernos e uma das mais usadas no mercado [Salles 2018], o código já escrito pode ser facilmente portado para uma plataforma web, podendo ser executado em praticamente qualquer dispositivo com acesso à mesma.

Depois das tecnologias terem sido definidas, a organização da aplicação foi pensada com os objetivos de facilitar a utilização de boas práticas de desenvolvimento e de agilizar a implementação. Para tal, separou-se o sistema nos seguintes agentes: (i) Cliente, responsável pela inicialização e por apresentar o cenário da jogabilidade; (ii) Gerenciador de Progresso, responsável pelo controle do fluxo de interações entre o jogador e o sistema; (iii) Scrum Master, que gerencia todas as cerimônias do Scrum; (iv) Executor de Regras, que valida as atividades do jogador

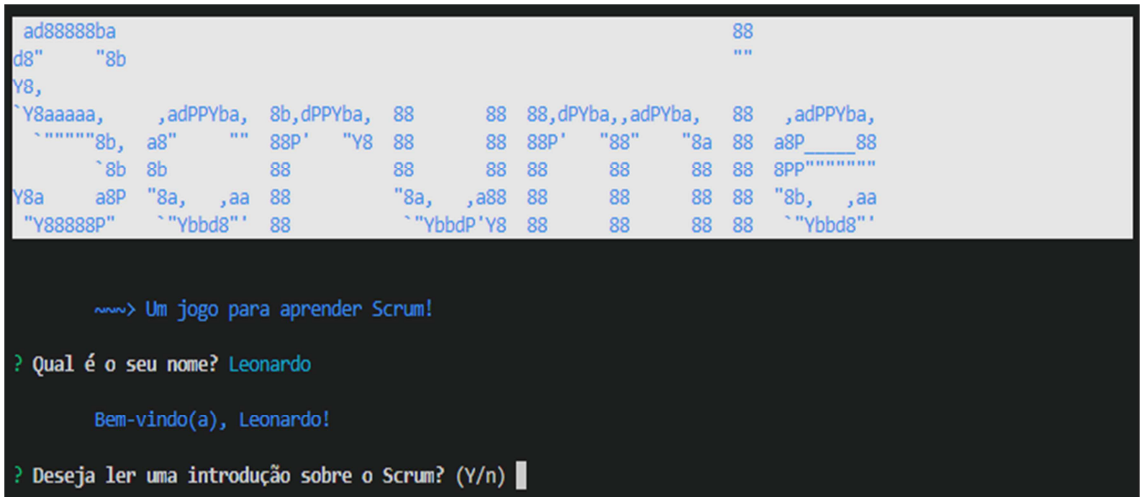

Figura 4. Tela inicial do jogo Scrumie

Além disso, optou-se por não utilizar banco de dados, já que nesta versão apenas as informações sobre a classificação dos jogadores são armazenadas. As configurações, do jogo são carregadas e armazenadas durante a execução como objetos. Como modo de

[1] Disponível em: http://nodejs.org/en/about/

[2] Disponível em: https://www.typescriptlang.org/

interação com o jogador, a interface de linha de comando foi a escolhida, devido à sua simplicidade e eficiência.

Durante a implementação, algumas das técnicas utilizadas foram: reuso de código através de herança e de classes auxiliares, nomes bem definidos, formatação padronizada e funções pequenas e objetivas. Deste modo, eventuais versões futuras do jogo terão sua manutenção e seu desenvolvimento facilitados.

A implementação dos agentes proporcionou a adição de inteligência ao jogo, além de ter facilitado a separação das responsabilidades e dos papéis dos módulos do sistema, permitindo uma melhor organização.

6. Conclusões

O jogo Scrumie representa a maior contribuição deste artigo, como proposta de investir no aprendizado prático por meio do uso de jogos educativos em *software*. Considera-se um estudo relevante devido à busca crescente por eficiência em empresas no desenvolvimento de sistemas e à consequente importância dada ao conhecimento de como aplicar métodos ágeis.

O uso de agentes facilitou a separação de papéis envolvidos na equipe *Scrum*, apesar de trazer maior complexidade para a implementação. Por ser uma arquitetura diferente da convencional, utilizar agentes foi um desafio, no sentido de saber inserir a autonomia e a inteligência geralmente associadas aos agentes da maneira correta. Esse primeiro protótipo foi desenvolvido de forma bastante simples e no futuro pretende-se a incorporação de uma arquitetura BDI por meio de um framework orientado a agentes. Assim, a abordagem de ter agentes representando pessoas de uma equipe serviu como uma solução para este problema abstrato, gerando bons resultados. No futuro, esses agentes podem evoluir, estendendo o jogo para uso de múltiplos papéis.

Alguns pontos negativos originados no 2TScrum não foram abordados para melhoria no Scrumie, como o fato de possuir apenas um cenário de desenvolvimento simulado – o sistema *web* para uma biblioteca; e possuir um limite de participantes no jogo, sendo que o Scrumie admite apenas um jogador. Sendo assim, esses pontos ficam sugeridos como melhorias futuras. Além deles, em relação à implementação, próximas versões do jogo poderiam criar uma interface gráfica, para torná-lo mais atraente aos jogadores; e migrá-lo para uma implementação *web*, para que tenha mais acessibilidade.

Referências

Beck, Kent. Embracing change with extreme programming. Computer, v. 32, n. 10, p. 70-77, 1999.

Beck, Kent; et al. Manifesto for agile software development. Disponível em: <http://agilemanifesto.org/>. Acesso em 12 de dez. de 2018.

Brito, A.; Vieira, J. '2TScrum': A Board Game to Teach Scrum. In: Proceedings of the 31st Brazilian Symposium on Software Engineering. ACM, 2017. p. 279-288.

Chella, Antonio; et al. Agile PASSI: An agile process for designing agents. International Journal of Computer Systems Science & Engineering, vol. 21, no. 2, p. 133-144, 2006.

Chella, Antonio; et al. From passi to agile passi: Tailoring a design process to meet new needs. In: Intelligent Agent Technology, (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on. IEEE, p. 471-474, 2004.

Cons, Bruna. Comparação entre Metodologias Ágeis Orientadas a Agentes. 2018. 71 f. Monografia (Graduação em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

Dallavalle, Silvia Inês; Cazarini, Edson Walmir. Regras do Negócio, um fator chave de sucesso no processo de desenvolvimento de Sistemas de Informação. Encontro Nacional de Engenharia de Produção. São Paulo, 2000.

De Paula, Felipe. MAS Ontology: uma ontologia de métodos orientados a agentes. 2014. 160 f. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. 2014.

Devmedia, Equipe. Processos Ágeis para desenvolvimento de Software – Parte 02. DevMedia. Disponível em: <https://www.devmedia.com.br/processos-ageis-para-desenvolvimento-de-software-parte-02/9209>.

Ferreira, Vitor. Comparação de Desenvolvimento Orientado a Agentes para Jogos Educacionais: Um estudo de caso. 2015. 163 f. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. 2015.

Grando, Regina. O jogo na educação: aspectos didático-metodológicos do jogo na educação matemática. Unicamp, 2001 <www.cempem.fae.unicamp.br/lapemmec/cursos/el654/2001/jessica_e_paula/JOGO.doc> Acesso em 18/maio/2003.

Henderson-Sellers, Brian; Giorgini, Paolo. Agent-oriented Methodologies. Hershey: Idea Group, 2005. 429 p.

Knublauch, Holger. Extreme programming of multi-agent systems. Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 2 - AAMAS '02, 2002.

Larman, Craig. Utilizando UML e padrões. Bookman Editora, 2002.

Legey, Ana Paula et al. Desenvolvimento de Jogos Educativos Como Ferramenta Didática: um olhar voltado à formação de futuros docentes de ciências. Alexandria: Revista de Educação em Ciência e Tecnologia, v. 5, n. 3, p. 49-82, 2012.

Moratori, Patrick. Por que utilizar jogos educativos no processo de ensino aprendizagem. UFRJ. Rio de Janeiro, 2003.

Pressman, Roger. Engenharia de Software. Edição 6. Porto Alegre: MCGrawHill, 2010. 711 p.

Salles, Filipe. Top 10 linguagens de programação mais usadas no mercado. DevMedia. Disponível em: <https://www.devmedia.com.br/top-10-linguagens-de-programacao-mais-usadas-no-mercado/39635>.

Shehory, Onn; Sturm, Arnon. Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks. Berlim: Springer-Verlag Berlin Heidelberg, 2014. 331 p.

Sille, Felipe; Braga, Juliana Cristina. Software educacional para prática do scrum. In: Anais dos Workshops do Congresso Brasileiro de Informática na Educação. 2013.

Soares, Michel. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. INFOCOMP, v. 3, n. 2, p. 8-13, 2004.

Sommerville, Ian. Engenharia de Software. Edição 9. Tradução Ivan Bosnic; Kalinka Oliveira. São Paulo: Pearson, 2013. 529 p.