Modelando Requisitos de um Jogo Educacional Médico usando a Metodologia INGENIAS SCRUM

Fernanda T. Novo[1], Vanessa M. da Silva[2], Bruno G. Ochotorena[2] Lucas d'Amaral Pires[2], Bruna Costa Cons[1,2], Rosa Maria E. M. da Costa[1,2], Vera M.B. Werneck[1,2]

[1]Programa de Mestrado em Ciências Computacionais,
Universidade do Estado do Rio de Janeiro, Brasil
[2]Ciência da Computação, Universidade do Estado do Rio de Janeiro, Brasil
nandanovo23@gmail.com, {rcosta, vera}@ime.uer.br

Resumo. Metodologias orientadas a agentes têm sido aplicadas no desenvolvimento de jogos educativos. No entanto, o uso dos métodos clássicos pode gerar excessivos custos, tempo e dificuldades na adaptação ao escopo dos produtos. Neste trabalho, são apresentados os resultados da aplicação de um método ágil orientado a agentes (INGENIAS SCRUM) no desenvolvimento de um jogo educativo para a área da saúde. Os modelos de requisitos, o processo de implementação e desenvolvimento são discutidos e apresentados neste trabalho. Evidências positivas do uso deste método foram observadas na qualidade do produto gerado e no tempo de desenvolvimento quando comparado a outro estudo, que usou o mesmo jogo como um modelo comparativo.

Abstract. Agent-oriented methodologies have been applied in the development of educational games. However, the use of classical methods can generate excessive costs, time and difficulties to adapt it to the scope of products. This work presents the results of the application of an agile method oriented to agents (INGENIAS SCRUM) in the development process of an educational game for the health area. The requirements models, the process of implementation and development are discussed and presented. Positive evidences of the use of this method were observed in the quality of the generated product and in the development time when compared to another study that used the same game as a comparative model.

Keywords: Multi-Agents Systems, INGENIAS SCRUM Methodology, Agile Methodologies, Medical Game.

1 Introdução

Recentemente, a área de sistemas multiagentes (SMA) vem sendo reconhecida e suas técnicas utilizadas em diferentes experiências de desenvolvimento de produtos de software [1]. Em geral, esses sistemas são constituídos por agentes autônomos e inteligentes, que visam atingir um objetivo comum. Nesta perspectiva, diversos métodos orientados a agentes foram propostos [1], [2]. Logo, é importante que diante de tantos

métodos distintos, se escolha a abordagem mais adequada a cada caso específico. A prioridade dos serviços computacionais é satisfazer aos requisitos dos usuários e, por isso, o uso de métodos facilita a identificação e modelagem dos requisitos.

Por outro lado, os métodos ágeis de desenvolvimento de software têm sido utilizados de forma significante, sendo que o objetivo de se utilizar esses métodos é a busca de estratégias de construção de sistemas de alta qualidade de maneira rápida, atendendoos requisitos do sistema. Esses métodos ágeis possuem um comportamento regular e prático para o desenvolvimento de software, considerando a elaboração de planos de ação, que por sua vez, consideram implicitamente as fases futuras, de modo a adaptá-las a mudanças de acordo com a evolução do projeto.

Ferreira [3] avaliou dois métodos orientados a agentes (O-MaSE, AgilePASSI)[4] e a abordagem Scrum [5] em um estudo de caso, modelando e implementando um jogo médico educacional. Este trabalho deu continuidade a esse estudo utilizando o método INGENIAS SCRUM [6] com objetivo de confirmar o resultado obtido em Ferreira [3], gerando uma alternativa de rapidez e qualidade para o desenvolvimento de sistemas com orientação a agentes.

O método INGENIAS SCRUM [4],[6] explora a abordagem de meta-modelos do método INGENIAS e agrega os conceitos do método ágil de gerenciamento de projetos SCRUM. Baseia-se em equipes auto-organizadas e habilitadas para entregar os incrementos do produto, focadas no cliente ou proprietário do produto, que deve fornecer à equipe de desenvolvimento uma lista dos requisitos desejados, usando o valor comercial como o mecanismo de prioridade.

Este trabalho apresenta os resultados de um estudo piloto de desenvolvimento e modelagem de requisitos utilizando o método INGENIAS SCRUM, estando organizado em 5 seções. Na primeira é fornecida uma visão geral e objetivo do artigo. Na seção 2 são discutidos os jogos educacionais na saúde e definidas as características e os requisitos do jogo MEDEDUC. Na seção 3, o método INGENIAS SCRUM é descrito em detalhes e na seção 4 o desenvolvimento do jogo MEDEDUC utilizando o método INGENIAS SCRUM é apresentado. Finalmente, na seção 5, são discutidas as considerações finais e os trabalhos futuros.

2 Jogo Médico MEDEDUC

Jogos educacionais podem servir como um meio para promover treinamento nas áreas profissionais da saúde e proporcionar uma oportunidade para a educação interdisciplinar. A crescente popularidade desses jogos ocorre pela ênfase em suas vantagens educacionais positivas sobre os métodos tradicionais de ensino [7]. Entre as vantagens em utilizar jogos educacionais, destaca-se a criação de ambientes gratificantes e atraentes, que servem como estímulo para o desenvolvimento integral dos usuários [8], [9]. Na educação profissional em saúde, os jogos são reconhecidos como *Serious Games* e adotam abordagens de simulações, jogos simulados, ambientes virtuais, jogos sociais e cooperativos e jogos de realidade alternativa [10].

O jogo educacional MEDEDUC é estruturado da seguinte maneira: O jogador (usuário) é submetido a perguntas sobre conceitos médicos que são separadas em 5 níveis principais de dificuldade. Conforme é aprovado em cada nível, ele ganha um

equipamento médico para montar seu consultório virtual. Este equipamento pode ser um computador, um estetoscópio, um aparelho de raioX, etc. Ao finalizar os cinco níveis, o jogador terá o consultório equipado com os equipamentos ganhos durante o jogo. Antes de iniciar os níveis principais, o jogador passa por uma fase de nivelamento para a verificação de seu nível de conhecimentos, sendo então, alocado em um nível específico, de acordo com seus acertos nesta fase. Caso o jogador não tenha o rendimento esperado (mais de 70% de acertos), ele é direcionado à fase de reforço. A tabela 1 lista os requisitos do jogo identificados e definidos junto com médico pneumologista. Esses requisitos foram os mesmos utilizados no estudo de casos apresentado por Ferreira [3].

Tabela 1. Lista de Requisitos do jogo MEDEDUC

	Requisito
01	Ao entrar no jogo pela primeira vez, usuário é submetido a questões para definir seu nível e as demais funcionalidades estão indisponíveis.
02	A qualquer momento o usuário pode verificar módulos disponíveis para acesso.
03	Usuário visualiza apenas módulos disponíveis conforme sua evolução.
04	Ao selecionar o nível disponível na página principal, usuário é direcionado para a primeira questão aleatória do Nível em questão.
05	As questões disponibilizadas têm níveis de conhecimento de 1 a 5, sendo as questões de nível 1 são mais fáceis do que as questões de nível 5.
06	*Feedback* ao usuário do resultado ao final do módulo (positivo ou negativo).
07	As questões do nivelamento,níveis e reforço são mescladas nos formatos:áudio,vídeo,imagens,texto.
08	Cada questão do nivelamento, de nível e reforço é composta por 3 opções de respostas (A, B e C), sendo uma opção correta e duas incorretas.
09	Ao selecionar uma opção incorreta em uma questão, o usuário é informado da opção correta.
10	Ao selecionar uma opção incorreta em uma questão, o usuário recebe um retorno com uma justificativa relatando o porquê de a opção escolhida estar incorreta. Este retorno pode ser em formato de áudio, vídeo, imagens e texto.
11	Para cada pergunta o usuário pode rever a questão, resposta e retorno. Entretanto, o usuário não consegue alterar a opção selecionada.
12	O nivelamento corresponde a 10 questões referente aos cinco níveis disponíveis. Para cada nível, existem duas questões distribuídas aleatoriamente.
13	Durante todo o nivelamento, níveis e reforço, usuário acompanha a sua evolução com uma barra de status. Para acertos, usuário vê um indicativo verde na barra de status. Enquanto para erros, usuário vê um indicativo vermelho.
14	Ao final do nivelamento, respondendo todas as questões, usuário é informado o nível alcançado: Nível 5, (acertar todas as questões do nível 1ao 4), Nível 4 (acertar todas as questões do nível 1ao 3), Nível 3 (acertar todas as questões do nível 1 e 2), Nível 2 (acertar todas as questões do nível 1) e Nível 1 (errar uma das duas perguntas do Nível 1).
15	O usuário pode interromper o nivelamento e deve iniciar o processo novamente e as respostas anteriores são desconsideradas.
16	O usuário deve realizar o Nível 5, mesmo que acerte todas questões do nivelamento.
17	Após finalizar o nivelamento, usuário é direcionado para a página principal do jogo e poderá iniciar o nível alcançado.
18	Ao entrar no jogo após realização do nivelamento, usuário permanecerá no último nível alcançado.
19	Durante os níveis 1, 2, 3, 4 e 5, usuário responde 10 questões referentes ao nível de conhecimento 1, 2, 3, 4 e 5, respectivamente.
20	O usuário deve poder conseguir visualizar uma questão, resposta e retorno. Entretanto, usuário não poderá alterar a opção selecionada.
21	Ao final de cada nível, usuário que obter 70% de acertos ou mais, está apto a avançar para o próximo nível.
22	O usuário obtém ao final de cada Nível um Equipamento para o seu consultório.

3 O Método INGENIAS SCRUM

INGENIAS é uma metodologia orientada a agentes baseada no conceito de metamodelo que define as primitivas e propriedades sintáticas de um modelo [11]. O desenvolvimento dirigido a modelos (MDD) ocorre por meio da definição de modelos, que transformam a especificação do sistema em produtos como código, testes e documentação, de forma semiautomática [4].

Os metamodelos são responsáveis por definir precisamente a linguagem de especificação e sua semântica e sintaxe, além de introduzir os modelos, necessários para o uso de ferramentas MDD, no caso, o *Development Kit* (IDK). Cada metamodelo especifica diferentes tipos de diagramas UML que podem ser construídos para descrever sua perspectiva do sistema. Assim, a especificação de um sistema em INGENIAS inclui cinco modelos: Modelo de Organização, Modelo de Agente, Modelo Tarefas e Metas, Modelo de Interação e Modelo de Ambiente [4].

INGENIAS *Development Kit* (IDK), é uma ferramenta gráfica que permite a criação de modelos em MAS,incluindo um editor gráfico e *plug-ins* que adicionam funcionalidades, auxiliando no desenvolvimento do sistema, na modelagem e na customização dos processos.

INGENIAS SCRUM é uma metodologia baseada no processo de desenvolvimento SCRUM e no metamodelo do INGENIAS. Durante a etapa de especificação de requisitos do sistema, através da construção do *product backlog*, os agentes realizam um ciclo de deliberação. Neste ciclo, os agentes identificam novas tarefas a serem escalonadas, ou retiradas do planejamento e executam cada tarefa. Isso não impede que novas tarefas ou informações sejam adicionadas aos agentes, o que é verificado por um novo ciclo.

A seguir o Modelo INGENIAS é descrito, já que forma a base do modelo INGENIAS SCRUM, que é detalhado em seguida.

3.1 Modelo INGENIAS

O INGENIAS permite a descrição do comportamento adaptativo dos agentes por diversos pontos de vista, por meio da especificação de seus modelos (Organização, Agentes, Tarefas e Metas, Interação e Ambiente).

O Modelo de Organização define a arquitetura do sistema, contendo agentes, recursos, tarefas e metas que se relacionam. Exprime a organização do MAS, fornecendo uma visão global dos agentes, os papéis em que eles participam e quais suas responsabilidades ao buscar cumprir suas metas.

O Modelo de Agentes identifica e descreve os agentes do sistema e suas responsabilidades, como habilidades sociais e de negociação, ou capacidade de aprendizado. Os papéis dos agentes definem as tarefas a serem realizadas e as metas a serem alcançadas e são atribuídos aos mesmos. Este modelo define, controla e gerencia o estado mental na individualidade de cada agente, além do mecanismo de tomada de decisões. Segundo HENDERSON-SELLERS [2], tem uma visão focada na funcionalidade de cada agente: propósito (quais metas o agente deve buscar alcançar), responsabilidades (quais tarefas ele deve executar) e capacidades (quais papéis ele desempenha).O com-

portamento do agente é definido por meio de três componentes: Estado Mental, Gerenciador de estado mental e Processador de estado mental.

O Modelo de Tarefas e Metas decompõe tarefas e metas identificadas em diagramas anteriores e inclui tarefas entrelaçadas entre si, introduzindo uma definição comportamental dos agentes ao sistema. As metas são desejos dos agentes e cada tarefa deve possuir uma meta associada a ela, de forma que a mesma só é uma candidata à execução caso sua meta correspondente não tenha sido alcançada. Para cada tarefa, o modelo determina quais elementos são requeridos e quais resultados são esperados. Esse modelo define como uma meta alcançada afeta outras por meio de relacionamentos de decomposição e dependência, mostrando casos em que a resolução de uma submeta resolve uma meta mais abrangente. Um modelo de tarefas e metas pode representar a forma em que uma tarefa afeta o estado mental de um agente.

O modelo de interação envolve a troca de informações ou requisições entre agentes, ou entre agentes e usuários humanos, sobre como trocam mensagens e compartilham conhecimento. Em alto nível, apenas informações sobre participantes e seus papéis na comunicação são requeridas. Informações adicionais incluem protocolos de interação, qual informação foi trocada e quais tarefas poderão processar tais informações. Segundo HENDERSON-SELLERS [2], uma interação é definida por: atores, contexto, natureza e especificação. O INGENIAS não possui um formalismo que determina qual tipo de notação para especificar interações deve ser seguida, podendo ser detalhados por diagramas UML ou os diagramas de interação GRASIA, que dividem uma interação em unidades de interação.

O modelo de Ambiente mostra os elementos que constituem o ambiente do sistema e, consequentemente, que os agentes devem perceber e interagir. Define os relacionamentos dos agentes com elementos externos. Além disso, define três entidades observáveis: os recursos consumidos pelo sistema, as aplicações utilizadas e os agentes que interagem com estes elementos.

3.2 Processo de Desenvolvimento do INGENIAS

INGENIAS é uma metodologia adaptada do *Unified Development Process*(UDP),que suporta todo o ciclo de desenvolvimento de um MAS, envolvendo o básico da elicitação de requisitos, análise, *design*, criação de código e testes. O processo de desenvolvimento do INGENIAS tem três fases (início, elaboração e construção), com dois tipos de fluxos de trabalho (análise e *design*). A tabela 2 resume essas fases, fluxos de trabalho e suas respectivas atividades.

3.3 Fases do INGENIAS SCRUM

O processo do INGENIAS SCRUM introduziu duas fases ao processo INGENIAS: fase de preparação e *Sprints*. *Sprint* é a fase principal de um projeto SCRUM, ocorrendo de forma iterativa e incremental, com o auxílio do IDK e do IAF.

O objetivo da fase de preparação é definir o planejamento da entrega, estabelecendo as funcionalidades requeridas e as tarefas necessárias para a entrega do próximo incremento. Compreende três atividades: (i) Início do *ProductBacklog* que visa pro-

duzir um *backlog*, listando as funcionalidades do sistema desejadas pelo *ProductOwner*que lista e prioriza os requisitos. (ii) Tarefas de Preparação realizada pela equipe SCRUM buscando preparar reuniões entre os *stakeholders* do sistema e o *ProductOwn*er. Nesta fase, o *backlog* é refinado e organizado. (iii) Planejamento da Entrega que compreende diversas especificações (duração, velocidade, produtos) sobre os sprints, estabelecidas pelo *SCRUM Master* juntamente com o *ProductOwner*.

Tabela2. Processo de desenvolvimento do INGENIAS [4].

	Fases		
	Concepção	**Elaboração**	**Construção**
ANÁLISE	Gerar casos de uso e identificar as ações com o modelo de interação correspondente.	Refinar os casos de uso, com novos componentes.	Estudar os casos de uso restantes.
		Gerar o modelo de agentes.	Validar código.
	Estruturar a arquitetura com modelo de organização.	Identificar os fluxos de trabalho e tarefas nomodelo de organização.	
	Gerar o modelo de ambiente, que reflete elicitação de requisitos.	Obter o modelo de tarefas e meta.	
		Refinar o modelo de ambiente, introduzindo novos elementos.	
DESIGN	Gerar um protótipo.	Concentrar o modelo de organização em fluxos de trabalho.	Gerar novos diagramas/refinar o modelo de agente.
		Refinar o modelo de tarefas e metas.	Estudar relacionamentos sociais da organização.
		Mostrar como as tarefas são executadas no modelo de interação.	Validar código.
		Gerar um modelo de agente que mostra os padrões de estado mental.	Gerar modelo de componentes.
			Gerar modelo de desenvolvimento.
			Gerar modelo de desenvolvimento.

O único produto produzido na fase de Preparação é o *Product Backlog*, formado por um conjunto de itens compostos, o *Product Backlog Items* (PBI) descrito por modelos de agentes e, opcionalmente, por um modelo de organização.

No INGENIAS SCRUM, um *sprint* dura no máximo 21 dias, pois o uso das ferramentas dirigidas a modelos ajuda a promover ciclos curtos. Sua meta é implementar e testar código enquanto alguma meta dos MAS é alcançada. Um *Sprint* envolve várias atividades, que são:(i) Planejamento do *Sprin*t, (ii) Atualização do *Product Backlog* (iii) Trabalhos Diários como refinar e criar modelos, programar e validar código (iv) Gerenciar Problemas (v) Conduzir Reunião Diária, (vi) Revisão do *Sprint*, (vii) Conduzir Retrospectiva, (viii) Liberação do Trabalho.

A equipe envolvida no processo *Sprint* é: (i) *Product Owner* que representa o cliente e define os itens que compõem o *Product Backlog* analisando as mudanças sugeridas e ordena as prioridades; (ii) *SCRUM Master* que tem papel de facilitador, deve agir como um técnico para equipe SCRUM e para o *Product Owner*; (iii) *SCRUM Team* que pode ser composto por três a cinco pessoas, responsáveis pelo incremento

do produto. Possui participação imprescindível nas seguintes atividades: planejamento de *sprin*t, atualização do *product backlog*, trabalhos diários e a condução de reuniões SCRUMdiárias; (iv) *Stakeholder* que tem participação opcional e pode ser representado por qualquer pessoa que não participa diretamente do projeto, mas pode influenciar o desenvolvimento do produto.

O modelo *Sprint* gera diversos produtos de trabalho compostos, baseados em especificações do INGENIAS. Eles são: Incremento do produto, *Product Backlog*, *Sprint Backlog* e Modelos INGENIAS. No Incremento do produto, o código fonte é escrito explicitando a estrutura do produto, em um estágio em que é passível de entrega ao fim de cada *Sprint*. No *Product Backlog* um conjunto de modelos INGENIAS especificam as funcionalidades do produto a serem obtidas. No *Sprint Backlog* são definidas uma lista de tarefas a serem realizadas para obter as funcionalidades desejadasque a equipe deve completar durante o Sprint. Os Modelos INGENIAS são produtos de tipo estrutural, composto e comportamental. Os modelos são exigidos para o uso das ferramentas IDK e IAF para geração de código automático e produção de testes.

4 Desenvolvimento do MEDEDUC com INGENIAS SCRUM

O jogo MEDEDUC utilizando o método do INGENIAS SCRUM foi desenvolvido num período de dois meses. O *SCRUM Team* foi composto por três alunos do último ano do curso de Ciência da Computação, o *SCRUM Master Team* por dois professores com especialidade em Engenharia de Software e Inteligência Artificial, e o *Product Owner* por um aluno de mestrado em Ciências Computacionais.

Com base na lista de requisitos definida pelo *PRODUCT OWNER* e *SCRUM MASTER*, foram definidos 4 Sprints a priori no *Product Backlog* (tabela 3) e entregue ao *SCRUM Master Team.*

Tabela 3. Product Backlog Inicial do MEDEDUC

	Artefato	Previsão	Entrega	Atraso
1	Relatório de Entregas Previstas Sprint 1	16/10/2017	16/10/2017	0
2	Reunião de revisão Sprint 1	23/10/2017	23/10/2017	0
3	Relatório de Entregas Previstas Sprint 2	30/10/2017	30/10/2017	0
4	Reunião de revisão Sprint 2	31/10/2017	31/10/2017	0
5	Relatório de Entregas Previstas Sprint 3	06/11/2017	06/11/2017	0
6	Reunião de revisão Sprint 3	13/11/2017	13/11/2017	0
7	Relatório de Entregas Previstas Sprint 4	27/11/2017	27/11/2017	0
8	Entrega do Jogo	04/12/2017	04/12/2017	0
9	Apresentação do Jogo	12/12/2017	12/12/2017	0

Na 1ª versão do *Sprint backlog* foram definidas as seguintes tarefas: (i) Definir a tela inicial com os componentes: Novo, Sair, *Layout* inicial e Imagem da tela, (ii) Unir os agentes na modelagem, (iii) Criar tela intermediária, (iv) Conectar com BD, (v) Criar o modelo de Organização (vi) Criar modelo de casos de uso.

Na 2ª versão do *backlog* foram definidas as tarefas: (i) Organizar o Backlog; (ii) Buscar perguntas para o jogo; (iii) Criar tela telas de perguntas tipo som; (iv) Criar

tela telas de perguntas tipo imagem; (v) Ler as perguntas do BD e mostrar na tela; (vi) Criar metamodelo de Agentes; (vii) Criar modelo de metas e tarefas.

Na 3ª versão do *backlog* foram definidas as tarefas: (i) Criar tela simulando o ganho do equipamento; (ii) Criar tela simulando o reforço; (iii) Criar regra na modelagem caso a pessoa acerte tudo, ela necessariamente começa do nível 5; (iv) Inserir perguntas no BD (v) Destacar resposta selecionada; (vi) Demonstração visual (no texto) de acerto ou erro; (vii) Criar modelo de interação.

Na 4ª versão do backlog foram definidas as tarefas: (i) Criar justificativa para respostas incorretas; (ii) Criar forma de salvar histórico de respostas; (iii) Criar tela final do jogo; (iv) Inserir equipamentos que serão os prêmios; (v) Cancelar progresso quando o usuário sair no meio do nível; (vi) Sincronizar a modelagem com o código desenvolvido pelo protótipo; (vii) Criar modelo de ambiente

Embora na prática, tenham sido definidos somente 4 *Sprints*, na fase de preparação foram necessários 6 *Sprints.* O tempo de desenvolvimento se manteve o mesmo que foi acordado no inicio entre o *SCRUM Team*, o *SCRUM Master Team* e o *Product Owner*. A tabela 4 apresenta uma visão Geral dos *Sprints Backlogs* de todo o desenvolvimento.

Tabela 4. Visão Geral dos *Sprints backlog*.

DATA	TAREFAS DEFINIDAS
09/10	**Tarefas Sprint 1** Definição da tela inicial com os componentes: Novo, Sair, Layout inicial e Imagem da tela. Primeira modelagem para o sprint 1.
16/10	**Revisão do Sprint 1** Pontos a serem corrigidos: união dos agentes na modelagem e botão da tela inicial do jogo.
	Tarefas do sprint 2 , Consertar o botão sair; Unir os agentes na modelagem; Pensar nivelamento; Fazer as demais modelagens dos agentes; Criar tela intermediária; Conectar com BD.
20/10	Todas as tarefas do Sprint 2 foram concluídas.
23/10 - 29/10	**Revisão do Sprint 2** Pontos a serem corrigidos: Falta na modelagem Agente usuário, devendo este estar presente no próximo Sprint.
	Tarefas do Sprint 3 Adicionar Agente Usuário, Modelo de metas, Organizar o Backlog, Buscar perguntas para jogo, Telas Som, Telas de Imagem, Implementar acerto e erro do usuário, Ler perguntas do BD, aparecer na tela
30/10	**Revisão do Sprint 3** Tudo ok.
06/11	**Tarefas do Sprint 4 ***Acesso ao histórico do usuário (próximo sprint) Tela simulando o ganho do equipamento, Tela simulando o reforço (identificação das telas), Completar a modelagem de metas, Criar regra na modelagem caso a pessoa acerte tudo, ela necessariamente começa do nível 5, Modelo de iteração, Inserir perguntas no BD, Destacar resposta seleciona, Sincronização com as bolinhas, Criar delay entre perguntas
06/11-	**Revisão do Sprint 4** Corrigir modelo de interação
12/11	**Tarefas do Sprint 5** Fazer casos de uso, Corrigir modelo de interação, Imagem de justificativa, Músicas, Nova coluna banco de dados, Botão de opções funcionar, Salvar histórico do nível 1
13/11	**Revisão do Sprint 5** Tudo ok.
- 10/12	**Tarefas do Sprint 6 (último sprint necessário)** Adicionar nova coluna no BD: justificativa, Tela final do jogo, Grasia Modelo de Interação, Terminar a tela do histórico, Terminar de implementar a tela inicial, Aplicar níveis concluídos, Pesquisar 4 equipamentos, Implementar tela de perguntas com imagem, Telas de pergunta indica o nível do jogador, Cancelar progresso quando sair no meio do nível, Implementar final de nível, Adicionar novos informações a nova coluna do BD, Terminar de implementar os níveis, Sincronizar a modelagem com o código desenvolvido pelo protótipo, Casos de uso: reforço, nivelamento e ganho de equipamento. Fazer modelo de ambiente, Gravar vídeo mostrando o jogo, Corrigir modelo ambiente.

O *backlog* do produto foi realizado de forma dinâmica e incrementado conforme as reuniões de planejamento. A tabela 4 contém os passos definidos com suas respectivas datas. Os *sprints* realizados foram documentados e controlados a partir de um documento de especificação no GoogleDoc e da ferramenta *online* Trello[1] .

Os diagramas requeridos pela metodologia INGENIAS SCRUM foram desenvolvidos utilizando a ferramenta INGENIAS Development Kit (IDK). Dentre os modelos desenvolvidos estão: Modelo de Organização, Modelo de Casos de Uso, Modelo de agentes, Modelo de metas e de tarefas, Modelos de interação e Modelo de ambiente.

O modelo de organização (Figura 1) do MEDEDUC começou a ser modelado no *Sprint* 1 e descreve os agentes, recursos, tarefas e os metas coexistem, sendo identificados 5 agentes: Agente Gerenciador de Conhecimento (Gcon), Agente Gerenciador de Telas (Gtel), Agente Gerenciador de Banco de Dados (Gbd), Agente Master (Gmas) e Agente Usuário (Gus). Esse diagrama facilita o entendimento e a visualização, a organização é apresentada com uma visão ampla apresentando o relacionamento entre agentes. A revisão do Sprint 2, *Product Owner* destacou que um agente não estava presente na modelagem (Agente Usuário). Desta forma, o agente foi inserido no modelo de organização no Sprint 3.

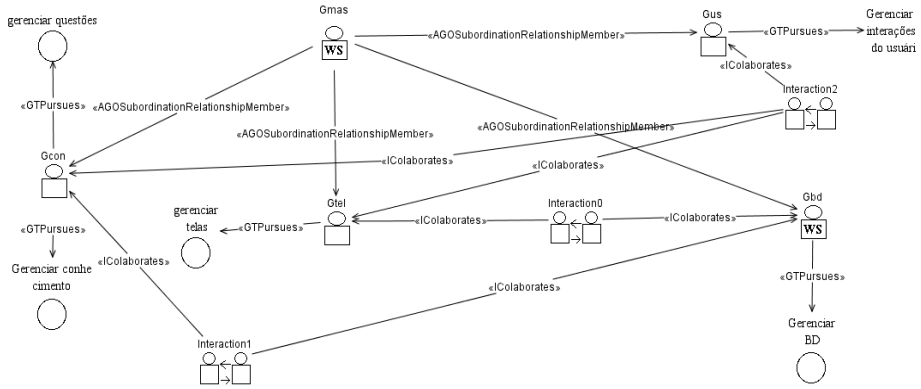

Fig. 1. Diagrama de Organização do Jogo MEDEDUC

[1] https://trello.com/b/YIuaGwo0/medeeduc

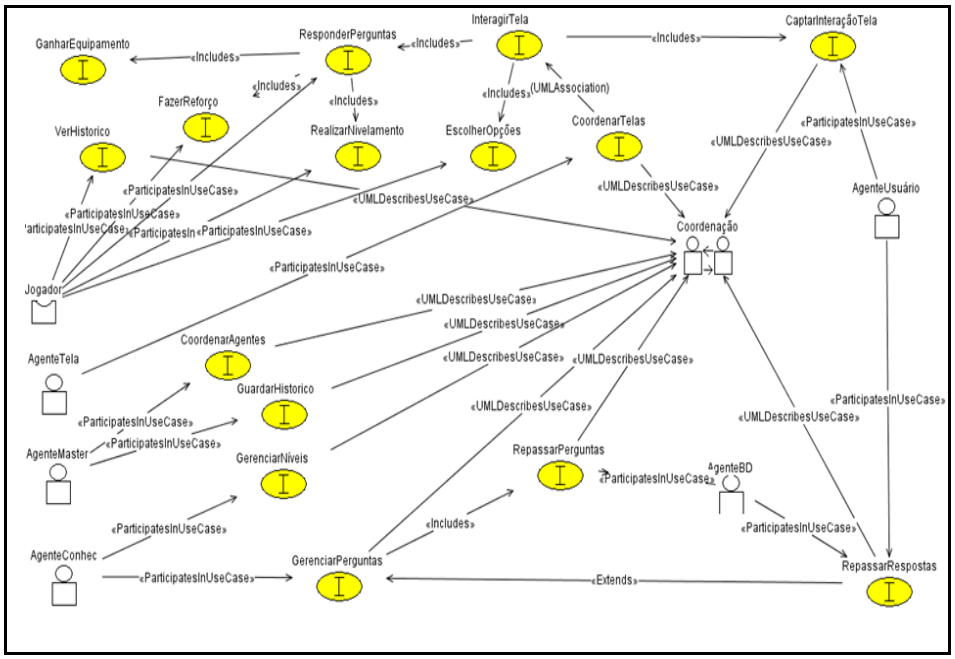

Fig. 2. Modelo de Casos de Uso do Jogo MEDEDUC

O ponto de vista do modelo de agentes (Figura 3) representa em detalhes a funcionalidade de cada agente: propósito, as responsabilidades e recursos. Foi modelado no *Sprint* 2 e alterado nos *Sprint* 3,4,5 e 6. Para melhor se adequar ao trabalho, houve pequenas alterações realizadas no último *Sprint* (6).

O Agente Gerenciador de Conhecimento (Gcon) tem como objetivo gerenciar todas as questões referentes às perguntas inseridas no jogo. Está relacionado aos estudantes de medicina e suas ações são: mostrar perguntas na tela e corrigir respostas.

O Agente Gerenciador de Telas (Gtel) gerencia os processos relativos às telas presentes no jogo (por exemplo, transições, sair do jogo). Está relacionado aos estudantes de medicina e sua ação é realizar mudanças entre telas do jogo.

O Agente Gerenciador de Banco de Dados (Gbd) gerenciao banco de dados contendo as questões do jogo. No Ambiente estão os agentes gerenciador de telas e gerenciador de conhecimento, e suas ações são repassar perguntas ao gerenciador de telas e salvar perguntas.

O Agente Master (Gmas) gerencia a integração entre todos os demais agentes para um funcionamento correto do jogo e se relaciona com todos os demais agentes do jogo. Sua sua ação é gerenciar/supervisionar interações entre os demais agentes.

O Agente Usuário (Gus) foi adicionado no *Sprint* 3 e gerencia as interações entre o usuário e o jogo e se relaciona com agente de tela, agente de conhecimento e estudantes de medicina. Suas ações são capturar respostas dos estudantes e repassar para o agente de tela e agente de conhecimento.

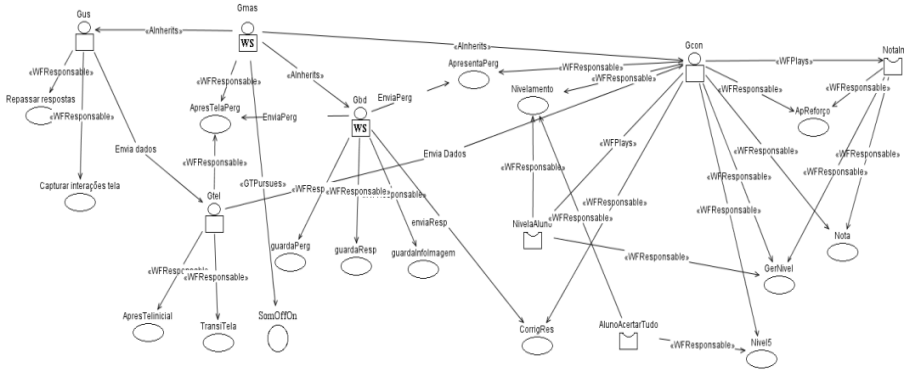

Fig. 3. Modelo de Agentes do Jogo MEDEDUC

O modelo de metas e tarefas (Figura 4) foi modelado no *Sprint* 3 e alterado no *Sprint* 5 e considera a decomposição de metas e tarefas descrevendo as consequências de realizar uma tarefa e o porquê ela deve ser realizada. A meta principal do MEDEDUC é auxiliar no Ensino de Medicina e para isso é decomposto em tarefas como Definir Estrutura do Jogo, Definir ferramenta para modelagem e Definir linguagem. Na Figura 4 são modeladas as tarefas referentes a essas submetas.

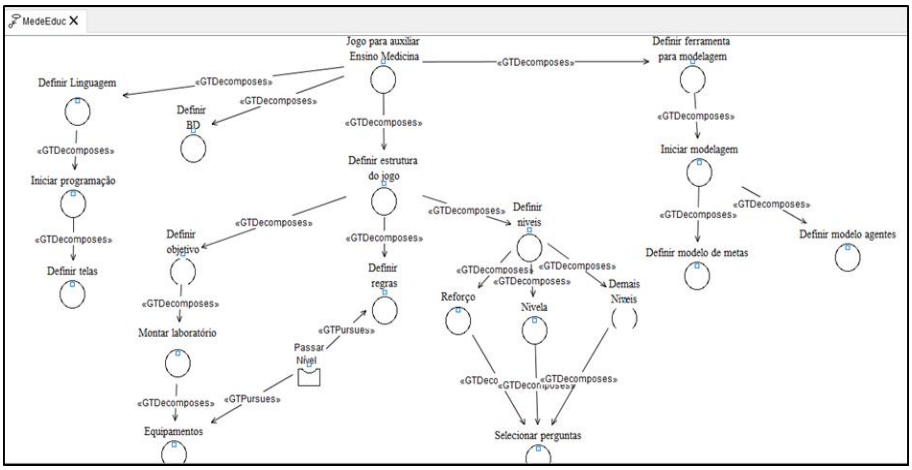

Fig. 4. Modelo de Metas e Tarefas do Jogo MEDEDUC.

O modelo de interação aborda a troca de informações ou pedidos entre agentes, ou entre agentes e usuários humanos. Este começou a ser modelado no *Sprint* 4 e no *Sprint* 6 foi alterado pela inserção do GRASIA, que melhorou a ideia anterior do modelo. O usuário interage com agente Conhecimento através CorrigeRespostaQuestao e ApresentaCorrecao que ele estabelece junto com AgenteTel. Na GRASIASpecification estão os detalhes da interação do jogo.

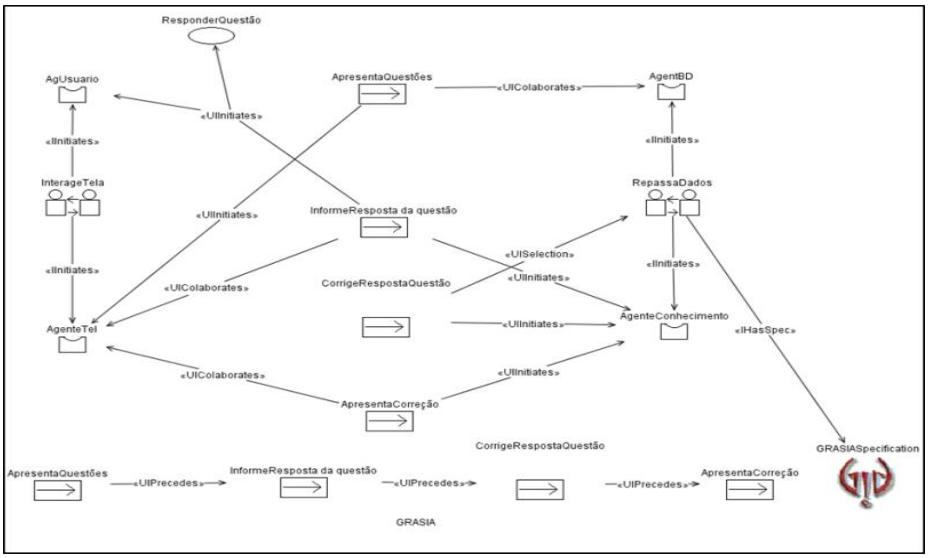

Fig. 5. Modelo de Interação do Jogo MEDEDUC

E, por fim, o modelo de ambiente definiu as entidades com as quais o sistema multiagentes interage. Esse modelo só foi construído no *Sprint* 6 e foi o modelo no qual a equipe teve mais dificuldade em projetar, pois é difícil reconhecer as partes do modelo no desenvolvimento do jogo que são recursos. O único recurso identificado foi o Banco de Dados utilizado (MySQL) e o *AssetsImagens*, que contém as imagens do programa utilizado para desenvolver, ou seja, a ferramenta Unity.

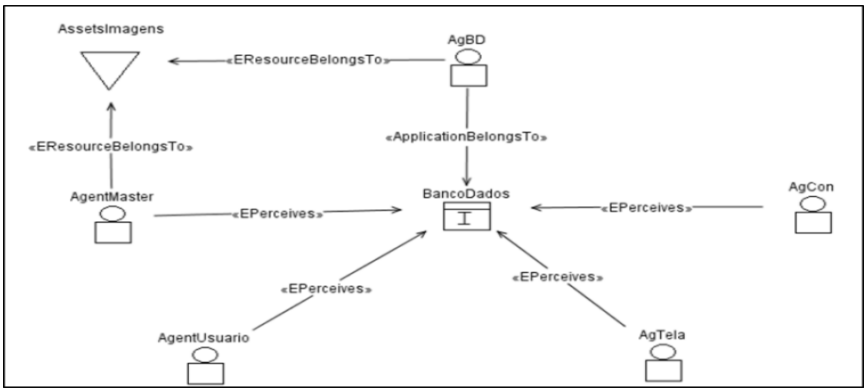

Fig. 6. Modelo de Ambiente do Jogo MEDEDUC

O produto final, Jogo MEDEDUC, foi apresentado para o *SCRUM Master Team* e *Product Owner* que consideraram os 22 requisitos (Tabela 1) estipulados na fase de preparação. Para pontuar os requisitos foram considerados a nota 0 para requisito não entregue, 1 para requisito parcialmente entregue e 2 para requisito entregue. O grupo

não teve atrasos nas entregas dos produtos e obteve 42 pontos em um total de no máximo 44 para requisitos entregues. Somente o requisito 10 não foi considerado nos modelos, nem implementado na versão final do jogo. Esse requisito se refere ao retorno do jogo com uma justificativa relatando o porquê da opção escolhida estar incorreta. Este retorno deveria ser em formato de áudio, vídeo, imagens e texto. Para uma primeira versão, esse resultado foi considerado como muito bom, pois 95,5% dos requisitos foram implementados e toda modelagem documentada, além dos *sprints* e revisões destes.

A Figura 7 apresenta algumas telas do jogo: Tela de Perguntas e Tela Final do Consultório Completo.

Fig. 7.a.Tela de Perguntas. b.Tela final do consultório completo

5 CONCLUSÃO

Este trabalho descreveu a adoção de um método ágil orientado a agentes para o desenvolvimento de um jogo médico educacional. Neste caso, utilizou-se o método ágil INGENIAS SCRUM.

Em Ferreira [3] foi apresentado o desenvolvimento do MEDEDUC com três métodos de desenvolvimento distintos (O-MaSE, AgilePASSI e SCRUM). Na conclusão desse estudo o método O-MaSE, foi que obteve o melhor resultado no atendimento dos requisitos (72% e os outros ficaram em torno de 55%). O processo SCRUM foi o que melhor entregou um produto final funcionando, mas com uma modelagem limitada em termos de agentes. Os resultados deste primeiro estudo apontaram a necessidade de integrar a metodologia SCRUM com uma metodologia orientada a agentes, para evitar as dificuldades observadas na identificação dos papéis dos agentes.

Assim, comparando com a experiência de Ferreira [3] o uso do método INGENIAS SCRUM fomentou uma experiência bastante produtiva, pois o sistema foi desenvolvido em dois meses, sendo modelados e implementados 22 requisitos, que são aproximadamente, 96% dos requisitos definidos, superando significativamente o obtido em Ferreira [3], e ainda, em menor tempo.

A partir deste trabalho novos trabalhos e pesquisas poderão ser desenvolvidos tanto no escopo do jogo MEDEDUC em experiências e testes com alunos da área da saúde, como na incorporação de inteligência no jogo, ou ainda, com a expansão da exploração de integração de desenvolvimento ágil com metodologias orientadas a agentes para a criação de outros produtos.

Referências

1. Sturm, A., &Shehory, O. The landscape of agent-oriented methodologies. In Agent-Oriented Software Engineering (pp. 137-154). Springer Berlin Heidelberg(2014).
2. Henderson-Sellers, Brian; GIORGINI, Paolo. Agent-oriented Methodologies. Hershey: Idea Group, p. 429. (2005).
3. Ferreira, Vitor. Comparação de Desenvolvimento Orientado a Agentes para Jogos Educacionais: Um estudo de caso. Dissertação (Mestrado em Ciências Computacionais) - Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. 163 f. (2015).
4. Cossentino, Massimo; et al. Handbook on Agent-Oriented Design Processes. Berlim: Springer Berlin, (2014).
5. Schwaber, Ken. Agile project management with Scrum. Microsoft press, 2004.
6. González-Moreno, J. C., Gómez-Rodríguez, A., Fuentes-Fernández, R., & Ramos-Valcárcel, D. INGENIAS SCRUM. In Handbook on Agent-Oriented Design Processes (pp. 219-251). Springer Berlin Heidelberg.(2014).
7. Boeker M, Andel P, Vach W, Frankenschmidt A. Game-based e-learning is more effective than a conventional instructional method: a randomized controlled trial with third-year medical students. PloS one. Dec 5;8(12):e82328.(2013).
8. Kashiwakura, Eduardo Yukio. Jogando e aprendendo: um paralelo entre videogames e habilidades cognitivas. São Paulo: Puc, (2008).
9. Bigdeli S, Kaufman D. Digital games in medical education: Key terms, concepts, and definitions. Med J Islam Repub Iran;31:52. (2017). https://doi.org/10.18869/mjiri.31.52.
10. Pettit RK, McCoy L, Kinney M, Schwartz FN. Student perceptions of gamified audience response system interactions in large group lectures and via lecture capture technology. BMC medical education. 22;15(1):92. (2015).
11. Steghoefer, Jan-Philipp; et al. Combining PosoMAS Method Content with Scrum: Agile Software Engineering for Open Self-Organising Systems. Scalable Computing: Practice and Experience, vol. 16, no. 4, (2016).