Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Fabiana Ribeiro Ferraz Gominho

Uma Proposta de Agilidade para Incorporação em Métodos Orientados a Agentes: um estudo de caso para jogos educacionais

Rio de Janeiro

2021

Fabiana Ribeiro Ferraz Gominho

Uma Proposta de Agilidade para Incorporação em Métodos Orientados a Agentes: um estudo de caso para jogos educacionais

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Orientadora: Prof.ª Dra. Vera Maria Benjamim Werneck
Coorientadora: Prof.ª Dra. Rosa Maria Esteves Moreira da Costa

Rio de Janeiro
2021

CATALOGAÇÃO NA FONTE

UERJ/REDE SIRIUS/BIBLIOTECA CTC/A

G633 Gominho, Fabiana Ribeiro Ferraz.
Uma proposta de agilidade para incorporação em métodos orientados a agentes: um estudo de caso para jogos educacionais/ Fabiana Ribeiro Ferraz Gominho. – 2021.
178 f.: il.

Orientadora: Vera Maria Benjamin Werneck
Coorientadora: Rosa Maria Esteves Moreira da Costa
Dissertação (Mestrado em Ciências Computacionais) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Software - Desenvolvimento - Teses. 2. Sistemas multiagentes - Teses. 3. Jogos educativos - Teses. I. Werneck, Vera Maria Benjamin. II. Costa, Rosa Maria Esteves Moreira da. III.Título.

CDU 004.415.2

Patricia Bello Meijinhos CRB7/5217 - Bibliotecária responsável pela elaboração da ficha catalográfica

Fabiana Ribeiro Ferraz Gominho

Uma Proposta de Agilidade para Incorporação em Métodos Orientados a Agentes: um estudo de caso para jogos educacionais

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Aprovada em 5 de agosto de 2021.
Banca Examinadora:

Prof.ª Dra. Vera Maria Benjamim Werneck (Orientadora)
Instituto de Matemática e Estatística - UERJ

Prof.ª Dra. Rosa Maria Esteves Moreira da Costa (Coorientadora)
Instituto de Matemática e Estatística - UERJ

Prof.ª Dra. Flávia Maria Santoro
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Eduardo Kinder Almentero
Universidade Federal Rural do Rio de Janeiro - UFRJ

Rio de Janeiro
2021

DEDICATÓRIA

Dedico a minha dissertação de mestrado em memória de meu pai, José Jaylson Ferraz Gominho, que em 1999 representou todos os pais na minha formatura de graduação em Ciência da Computação e discursou lindas palavras de incentivo para vida.

"MEUS FILHOS!...

Quando alguém, ao enfrentar uma dificuldade, volta atrás, é o mesmo que dizer que ele não viveu a experiência. Agindo assim, ele não formou uma experiência de vida. Em palavras mais atuais, ele não curtiu aquela situação.

A vivência integral de uma situação, é o único modo de dele realizar o processo de crescimento. Só assim um profissional poderá, após algum tempo, dizer que formou uma experiência no seu campo.

Ele tem que "curtir"as situações que podem resultar em medo, raiva, acovardamento ou o contrário, segurança e sucesso.

Quando não se sofre a experiência, apenas se consegue acumular cultura teórica, livresca e bibliografia. Assim, ninguém mais do que nós, seus pais, que os conhece tão bem, pode afirmar com a mais absoluta convicção, que vocês atingirão, na vida pessoal e profissional, o objetivo colimado.

Queremos ressaltar, mais uma vez, que nós estamos muito orgulhosos de vocês.

Que Deus, pai de todos nós, os abençoe e os guie pelos caminhos da obediência, da bondade, da justiça e da verdade, com a consciência de que, só o amor constrói!..."

Obrigada por sempre me apoiar e ter sido tão participativo na minha vida, te amarei para sempre! Sua filha.

AGRADECIMENTOS

Agradeço a Deus pela perseverança no meu caminhar, e por colocar pessoas tão especiais em minha vida.

Agradeço aos meus pais Jaylson *(in memoriam)* e Maria José, por terem me ensinado a importância da crença em Deus, e terem me iniciado no caminho espiritual.

Agradeço aos meus professores da FGV-Rio, os Drs. Ilan Chamovitz e Tânia Tisser Beyda, pelas cartas de avaliação fornecidas, por terem acreditado na minha persistência nos desafios do mestrado.

Agradeço as minhas orientadoras, Dras. Vera Werneck e Rosa Costa, pela dedicação a minha dissertação, mantendo o trabalho no trilho para que o objetivo fosse cumprido.

Agradeço ao meu companheiro, Antônio, que tantas vezes foi compreensivo, me incentivou, acalmou e animou com palavras carinhosas.

Agradeço a minha família, meus irmãos e cunhadas, Fábio e Estela, Flávio e Venina, que mesmo à distância, me apoiaram e me tranquilizaram em momentos tão difíceis de nossas vidas, para que eu seguisse em frente no mestrado.

Agradeço aos alunos da turma de Inteligência Artificial do segundo semestre de 2019, pela dedicação ao trabalho que levou ao estudo de caso.

Agradeço aos demais professores e alunos, principalmente ao meu querido colega de mestrado, Yuri Gabrich, que me apoiou incansavelmente nos momentos finais da minha dissertação. Conte sempre comigo!

Quando alguém, ao enfrentar uma dificuldade, volta atrás, é o mesmo que dizer que ele não viveu a experiência. A vivência integral de uma situação, é o único modo de realizar o processo de crescimento.

– Meu Pai

RESUMO

GOMINHO, Fabiana Ribeiro Ferraz *Uma Proposta de Agilidade para Incorporação em Métodos Orientados a Agentes*: um estudo de caso para jogos educacionais. 2021. 178 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2021.

Os conceitos do manifesto ágil têm sido amplamente utilizados no mercado e na área acadêmica como uma proposta na resolução de problemas, referentes ao processo de desenvolvimento tradicional. No âmbito das metodologias orientadas a agentes, cada vez mais metodologias ágeis têm sido adaptadas para melhorar o modelo de desenvolvimento de sistema tradicionais. Algumas metodologias orientadas a agentes, já tiveram a sua transformação como a PASSI e a Ingenias, que incorporam o conceito de desenvolvimento ágil. O objetivo desta dissertação de mestrado é propor um conjunto de atividades, artefatos e práticas ágeis, utilizadas no desenvolvimento ágil, para apoiar a proposta de adaptação ágil do modelo cascata orientado a agente: *Organization - based Multiagent Systems Engineering* (O-MaSE). Visando concretizar a proposta ágil, *framework agile* O-MaSE, foram mantidas as fases da metodologia O-MaSE, com as suas atividades e tarefas, e foram adicionados fragmentos dos *frameworks* ágeis: Scrum e *Extreme Programming* (XP). A proposta ágil foi utilizada no estudo de caso, juntamente com outras duas metodologias ágeis orientadas a agentes: a *Agile* Passi e a Ingenias-Scrum, que foram aplicadas por Novo (2018) em seu trabalho. Assim, agregaram-se novas evidências aos dados obtidos em seus estudos, formando uma ampla base para realizar comparações entre os métodos.

Palavras-chave: Método ágil orientado a agente. O-MaSE. Agile PASSI. Ingenias-Scrum.

ABSTRACT

GOMINHO, Fabiana Ribeiro Ferraz *An Agility Proposal for Incorporation in Agents Oriented Methods*: a case study for educational games. 2021. 178 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2021.

The concepts of the agile manifesto have been widely used in the market and in the academic area as a proposal to solve problems related to the traditional development process. Within the scope of agent-oriented methodologies, more and more agile methodologies have been adapted to improve the traditional system development model. Some agent-oriented methodologies have already undergone transformation, such as PASSI and Ingenias, which incorporate the concept of agile development. The objective of this master's dissertation is to propose a set of activities, artifacts and agile practices, used in agile development, to support the proposal of agile adaptation of the agent-oriented cascade model: Organization - based MultiagentSystems Engineering (O-MaSE). In order to implement the agile proposal, agile O-MaSE framework, the phases of the O-MaSE methodology were maintained, with their activities and tasks, and fragments of agile frameworks were added: Scrum andExtreme Programming (XP). The agile proposal was used in the case study, along with two other agile methodologies oriented to agents: agilePassi and Ingenias-Scrum, which were applied by Novo (2018) in his work. Thus, new evidence was added to the data obtained in their studies, forming a broad basis for making comparisons between the methods.

Keywords: agent oriented agile Method. O-MaSE. Agile PASSI. Ingenias-Scrum.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES E SIGLAS

***Agile* PASSI** *Process for Agent Societies Specification and Implementation Agile*

AAGDM Metodologia de Desenvolvimento de Jogos Ágil para Agentes

AAOM Modelagem Orientado a Agentes Ágeis

AOSE *Agent-OrientedSoftware Engineering*

CMS *Conference Management System*

DAS Desenvolvimento Ágil de Software

DPDT Modelo de Documentação do Processo de Design

IA Inteligência Artificial

IDK *INGENIAS Development Kit*

MAS *MultiAgent System*

MDA Manifesto para o Desenvolvimento Ágil

MDD *Model Driven Development*

O-MaSE *Organization - based Multiagent Systems Engineering*

OMACS Modelo de Documentação do Processo de Design Modelo de Organização para Sistemas Computacionais Adaptativos

OOPSLA *Object-oriented Programming, Systems, Languages, and Applications*

PBI *Product Backlog Items*

TDD *Test-Driven Development*

UDP *Unified Development Process*

UML *Unified Modeling Language*

XP *Extreme Programming*

SUMÁRIO

INTRODUÇÃO

Nos últimos anos as metodologias ágeis se disseminaram, criando novas demandas que apoiassem o processo de desenvolvimento de software e ampliassem a eficiência das estratégias adotadas em cada modelo de metodologia. Neste caso, várias propostas de *frameworks* surgiram e contemplavam tanto a coordenação das equipes, quanto a cooperação envolvida na dinâmica das reuniões, o *design* e a qualidade do produto. (GURUSAMY; SRINIVASARAGHAVAN; ADIKARI, 2016; QURESHI; KASHIF, 2017; CHAVES; FREITAS, 2020).

Por outro lado, a área de Sistemas Multiagentes - *MultiAgent System* (MAS) também teve uma grande expansão, com o aumento no número de novas metodologias e relatos de experimentos práticos. (DELOACH; GARCIA-OJEDA, 2014; OKOYE; ADETOKUNBO; OJIEABU, 2017).

Em geral, a modelagem de sistemas Multiagentes é um processo longo e complexo, que contempla todos os elementos necessários para o desenvolvimento de um sistema, capturando e descrevendo todos os requisitos do software. Visando aproveitar a capacidade de visualizar os sistemas Multiagentes mais rapidamente, alguns trabalhos integraram metodologias MAS com abordagens ágeis, como a metodologia *Process for Agent Societies Specification and Implementation Agile* (*Agile* PASSI) de Chella et al. (2006) e a Ingenias Scrum González-Moreno et al. (2014). Entretanto, os *frameworks* ágeis já existentes, que visam simplificar e apoiar o processo de modelagem dos sistemas, não se mostraram eficientes em projetos multiagentes, devido ao nível de complexidade inerentes a esses sistemas e uma maior busca por eficiência nas empresas. (FERREIRA, 2015).

Motivada pela disseminação das metodologias ágeis e possibilidades de adaptação de metodologias orientadas a agentes para a inclusão de agilidade, este trabalho considera uma metodologia orientada a agentes, baseada em conceitos consistentes e bem definidos, a O-MaSE, e integrou aspectos da abordagem ágil. A metodologia O-MaSE foi escolhida, por causa do resultado do estudo de caso de Ferreira (2015), que obteve a melhor solução. E também pela metodologia oferecer suporte no processo de desenvolvimento atual, com abordagem iterativa, incremental e abordagem simples do método Cascata de acordo com Deloach e Garcia-Ojeda (2014).

Visando identificar aspectos positivos e possíveis fragilidades, foi desenvolvido um estudo de caso baseado nos trabalhos de Ferreira (2015) e Novo (2018), que realizaram estudos de caso comparativos entre metodologias de modelagem de sistemas multiagentes, metodologias ágeis tradicionais e metodologias ágeis para sistemas multiagentes. Ferreira (2015) explorou duas metodologias orientadas a agentes, sendo: uma com processo de desenvolvimento tradicional (O-MaSE), uma adaptada à agilidade (*Agile* PASSI), mais o *framework* ágil não orientado a agente (Scrum). E Novo (2018), utilizou três metodologias

orientadas a agentes, sendo: duas adaptadas à agilidade (*Agile* PASSI e Ingenias Scrum) e uma que utiliza o processo de desenvolvimento tradicional (Ingenias).

Neste trabalho foram exploradas duas metodologias ágeis, que foram utilizadas no estudo de caso de Novo (2018): a *Agile* PASSI e a Ingenias Scrum, comparando os resultados com aquele obtido com o *framework Agile* O-MaSE proposto nessa dissertação.

Objetivo Geral

O objetivo geral deste trabalho é propor e avaliar um *framework* ágil orientado a agentes a partir do *framework* tradicional cascata O-MaSE, utilizando as boas práticas ágeis de desenvolvimento de software.

Objetivos Específicos

Os objetivos específicos desta dissertação são:

1. Realizar uma revisão da literatura, ressaltando os trabalhos na área de Sistemas Orientados a Agentes e Desenvolvimento Ágil, que irão apoiar os requisitos da metodologia proposta;

2. Propor um *framework* ágil, para a metodologia O-MaSE, nomeada de *Agile* O-MaSE para ser utilizada no estudo de caso;

3. Aplicar três diferentes metodologias ágeis orientadas a agentes: *Agile* PASSI, Ingenias Scrum e *Agile* O-MaSE no desenvolvimento do mesmo jogo médico educacional, realizado por grupos de estudantes da graduação da turma de Inteligência Artificial, do segundo semestre de 2019;

4. Analisar os resultados do estudo de caso, comparando os resultados da aplicação da *Agile* PASSI e do Ingenias Scrum, com o *framework* proposto: *Agile* O-MaSE;

5. Comparar os dados deste estudo de caso com os dados dos estudos de casos realizados por Ferreira (2015) e Novo (2018).

Organização

Esta dissertação está organizada em mais 5 capítulos, seguidos das referências, apêndices e anexos:

O Capítulo 1 apresenta a fundamentação teórica para dar cobertura e embasamento sobre: desenvolvimento ágil de software, *Extreme Programming*, Agentes de *Softwares* multiagentes, metodologia O-MaSE - que é o foco neste trabalho, metodologias: *Agile* PASSI e Ingenias-Scrum e estudos de caso em Engenharia de Software.

O Capítulo 2 apresenta a Revisão da Literatura, ressaltando os trabalhos na área de sistemas orientados a agentes e desenvolvimento ágil.

O Capítulo 3 apresenta a idealização da proposta da *framework Agile* O-MaSE para ser utilizada no estudo de caso.

O Capítulo 4 apresenta os resultados do estudo de caso com as metodologias *Agile* PASSI, Ingenias Scrum e *Agile* O-MaSE.

O Capítulo 5 apresenta as conclusões e considerações Finais.

1 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos e teorias, que suportam o desenvolvimento desta dissertação, constituindo sua fundamentação conceitual e técnica sobre: fundamentos das metodologias e práticas ágeis de desenvolvimento de softwares, agentes de softwares, sistemas multiagentes e metodologias ágeis orientadas a agentes.

1.1 Desenvolvimento Ágil de *Software*

De acordo com Sommerville (2019), as metodologias de desenvolvimento ágeis tiveram início no final dos anos 90, com o objetivo de reduzir radicalmente o tempo de entrega dos sistemas de *software* e as insatisfações com as despesas gerais, envolvidas nos métodos de *design de software* nas décadas de 80 e 90. Nesse sentido, o processo de desenvolvimento de software precisava evoluir rapidamente para reduzir as despesas da empresas, atendendo às necessidades de requisitos de negócios em constantes mudanças.

De acordo com Highsmith (2001), em fevereiro de 2001, um grupo de 17 pensadores denominados de "A aliança ágil" se reuniram e confeccionaram o Manifesto para o Desenvolvimento Ágil (MDA). O documento contém termos e conceitos gerais, que passaram a guiar o modo ágil de gerenciar projetos, onde o trabalho realizado teve em comum o pensamento:

> Estamos descobrindo maneiras melhores de desenvolver *software* fazendo-o nós mesmos e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar:
>
> 1. **Indivíduos e interações**, mais do que processos e ferramentas;
> 2. ***Software* funcionando**, mais do que documentação abrangente;
> 3. **Colaboração de seus clientes**, mais do que negociação de contratos;
> 4. **Responder a mudanças**, mais do que seguir um plano.
>
> Ou seja, embora haja valor nos itens da direita, valorizamos mais os itens da esquerda (BECK et al., 2001).

Além dos valores, também foram criados 12 princípios que guiam o Desenvolvimento Ágil de Software (DAS):

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor;

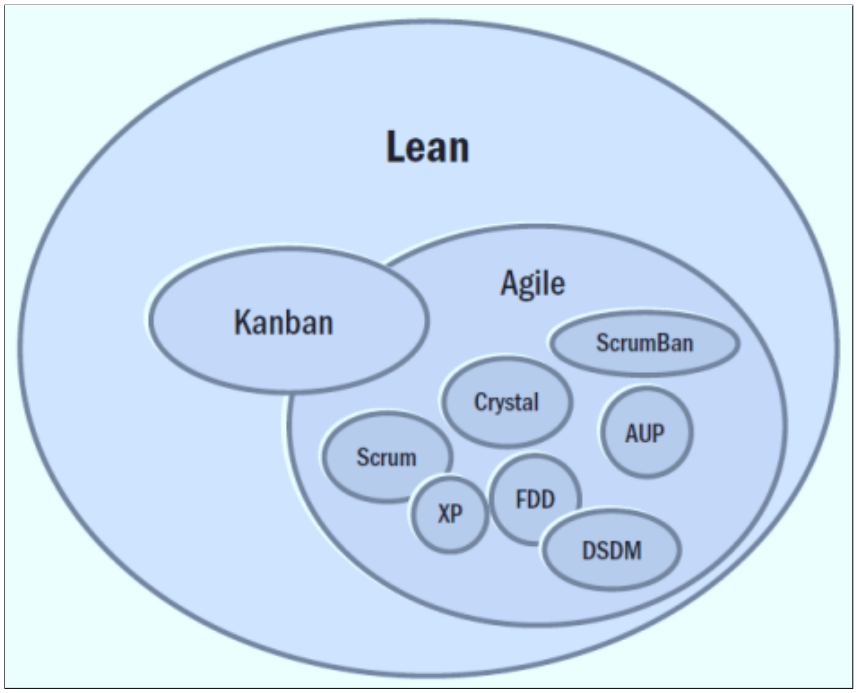
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas;

3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
4. Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;
5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário e confiar que farão seu trabalho;
6. O método mais eficiente e eficaz de transmitir informações para um time de desenvolvimento é através de uma conversa cara a cara;
7. A entrega de software funcional é a medida primária de progresso;
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter, indefinidamente, passos constantes;
9. Contínua atenção à excelência técnica e ao bom design, aumentam a agilidade;
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito;
11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis;
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

Neste caso, a abordagem é iterativa, onde a especificação, a implementação e os testes são intercalados e os resultados do processo de desenvolvimento de software são decididos explorando um processo de negociação entre equipe desenvolvedora e clientes.

Conforme exibido na Figura 1, atualmente existe uma variedade de *frameworks* e metodologias de desenvolvimento de *softwares* ágeis, que reúnem as características gerais descritas no MDA.

Figura 1 - *Frameworks* e Metodologias Ágeis



Fonte: Silva (2021).

1.1.1 *Framework Scrum*

Segundo Schwaber (1995), o termo *Scrum* foi inspirado em um tipo de falta do esporte *Rugby*, onde parte dos jogadores dos dois times se juntam com a cabeça abaixada e se empurram, levando todos os jogadores para frente, até marcar a maior quantidade de pontos possíveis.

O termo *Scrum* surgiu com os pesquisadores Takeuchi e Nonaka (1986) na publicação do artigo The New New Product Development Game – O novo jogo de desenvolvimento de novos produtos, da revista de Havard Business Review.

O artigo descreve técnicas japonesas utilizadas em indústrias automotivas de impressão e fotocopiadoras, que eram empregadas em times[1] pequenos, onde os próprios líderes dessas organizações ajudavam os seus colaboradores a resolver os problemas, tirando os impedimentos, para conseguir entregar os produtos que eram gerados em lotes pequenos, de forma incremental. Os pesquisadores perceberam que a forma de trabalho em que um ajuda o outro, empurrando para frente, se assemelhava ao que acontece na situação *Scrum* do *Rugby*.

Por volta de 1990, Jeff Sutherland ao ler o artigo dos pesquisadores, começou a

[1] É um grupo multidisciplinar de pessoas, responsável por realizar o trabalho de desenvolvimento do produto de ponta a ponta.

aplicar as técnicas descritas no artigo em desenvolvimento de software de uma empresa em que prestava serviço, obtendo bastante sucesso. Em 1994, Jeff Sutherland conhece o Ken Schwaber, que estava desenvolvendo um trabalho similar e juntos escrevem o primeiro artigo sobre o processo de desenvolvimento Scrum. Em 1995, eles apresentam esse artigo numa grande conferência chamada *Object-oriented Programming, Systems, Languages, and Applications* (OOPSLA) e a partir daí, o *Scrum* passou a ter mais colaboradores, se tornando bastante difundido no mercado.

De acordo com Schwaber e Sutherland (2017), o *Scrum* é considerado um *framework* adaptativo, dentro do qual podem ser aplicados vários processos ou técnicas. É utilizado para desenvolver, entregar e manter produtos complexos. Emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos. Ainda de acordo com Schwaber e Sutherland (2017), o *Scrum* é considerado: leve, simples de entender e difícil de dominar[2].

O *Scrum* é fundamentado nas teorias empíricas de controle de processos, onde três pilares apoiam a implementação de controle de processo empírico: transparência, inspeção e adaptação. A transparência está relacionada a aspectos significativos do processo que devem estar visíveis aos responsáveis pelos resultados. A transparência requer que estes aspectos tenham uma definição padrão comum, para que os observadores compartilharem um mesmo entendimento comum do que está sendo visto. Na inspeção os usuários devem, frequentemente, inspecionar os artefatos *Scrum* e o progresso em direção ao objetivo da *Sprint*[3] para detectar variações indesejadas. Porém, as inspeções não devem ser tão frequentes, de tal forma que atrapalhe o andamento dos trabalhos. As inspeções também são mais benéficas quando realizadas por inspetores especializados no trabalho a ser verificado. A adaptação é realizada a partir da inspeção. Se um inspetor identifica que um ou mais aspectos de um processo se desviou dos limites aceitáveis e que o resultado do produto será inaceitável, o processo, ou o material que está sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível de modo a minimizar mais desvios.

Associados aos três pilares, estão os valores do *Scrum*: comprometimento, coragem, foco, transparência e respeito, onde o sucesso no uso do *Scrum* depende das pessoas se tornarem mais proficientes na vivência destes cinco valores.

O *framework Scrum* consiste no time *Scrum* associados a papéis, eventos, artefatos e regras conforme o Quadro 1.

[2] É difícil de dominar porque demanda muita disciplina para manter o *framework* durante o projeto e, assim, obter um bom incremento de produto.

[3] *Sprint* é um time-box de duração consistente, que pode variar de 1 semana a 1 mês. Durante a *Sprint* um incremento de produto é liberado e uma nova *Sprint* é iniciada imediatamente após a conclusão da *Sprint* anterior.

Quadro 1 - Papéis do time *Scrum*.

Papéis do Time *Scrum*	Descrição
Time de desenvolvimento (Development Time)	Um grupo auto-organizado de profissionais, que deve ter no mínimo 3 e no máximo 9 pessoas. Eles são responsáveis pelo desenvolvimento do software e outros documentos essenciais ao projeto.
Dono do produto (*Product Owner*)	Um indivíduo ou o representante de um grupo (partes interessadas), cujo trabalho é identificar os requisitos do produto, priorizá-los para o desenvolvimento, e revisar continuamente a lista de pendências do produto para garantir que o projeto continue atendendo às necessidades de negócio.
Scrum Master	É responsável por garantir que o processo *Scrum* seja seguido e orienta a equipe no uso efetivo do Scrum. O profissional é responsável pela interface com o restante da empresa e por garantir que a equipe do *Scrum* não seja desviada por interferências externas.

Fonte: Sommerville (2019).

As *Sprints* consistem em ciclos de vida compostos pelos eventos: planejamento da *Sprint*, reuniões diárias, o trabalho de desenvolvimento, uma revisão da *Sprint* e uma retrospectiva da *Sprint*. Durante a *Sprint*: (i) Não são feitas mudanças que possam pôr em perigo o objetivo da *Sprint*;(ii) As metas de qualidade não diminuem;(iii) O escopo pode ser clarificado e renegociado entre o *Product Owner* e o Time de Desenvolvimento.

O *framework Scrum* destaca os seguintes eventos necessários para a sua execução: O Planejamento da *Sprint*, Reunião Diária, Revisão da *Sprint* e Retrospectiva da *Sprint*.

O Planejamento da *Sprint* é o evento que realiza a planificação do trabalho a ser realizado na *Sprint*. Este plano é criado com o trabalho colaborativo de todo o Time Scrum;

A Reunião Diária é um evento time-boxed de 15 minutos, mantida no mesmo horário e local todos os dias da *Sprint*, para reduzir a complexidade do Time de Desenvolvimento. Nela o Time de Desenvolvimento planeja o trabalho para as próximas 24 horas. Isso otimiza a colaboração e a performance do time através da inspeção do trabalho desde a última Reunião Diária, e dá previsão do próximo trabalho da *Sprint*;

A Revisão da *Sprint* é realizada no final da *Sprint* para inspecionar o incremento e adaptar o *Backlog* do Produto se necessário. Durante a Revisão da *Sprint* o Dono do Produto explica o que foi feito, o que não foi feito, o trabalho que foi adicionado e removido da *Sprint* e o time *Scrum* colabora com o Dono do Produto sobre o que foi feito na *Sprint*. Com base nisso e em qualquer mudança no *Backlog* do Produto durante a *Sprint*, os participantes colaboram nas próximas entregas, que podem ser feitas para otimizar valor.

A Retrospectiva da *Sprint* é uma oportunidade para o Time *Scrum* inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima *Sprint*.

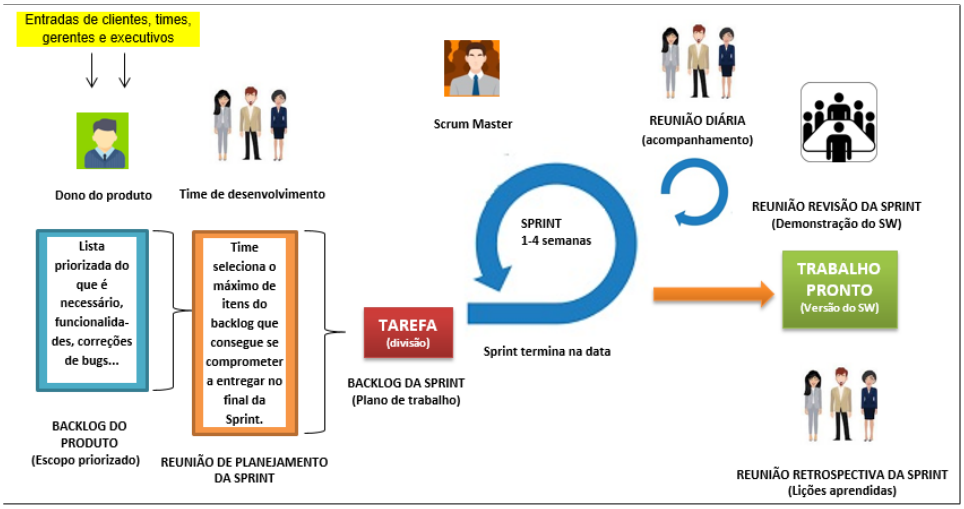
O *Scrum* possui artefatos que representam o trabalho para o fornecimento de transparência e oportunidade para inspeção e adaptação: *backlog* do produto, *backlog* da *sprint* e incremento. O *Backlog* do Produto é uma lista ordenada de tudo que é necessário para o produto. É a única origem dos requisitos para qualquer mudança a ser feita no produto.

O *Backlog* da *Sprint* é um conjunto de itens do *Backlog* do Produto selecionados para a *Sprint*, juntamente com o plano para entregar o incremento do produto e atingir o objetivo da *Sprint*. O *Backlog* da *Sprint* é a previsão do Time de Desenvolvimento sobre qual funcionalidade estará no próximo incremento e sobre o trabalho necessário para entregar essa funcionalidade em um incremento "Pronto"[4].

O Incremento é a soma de todos os itens do *Backlog* do Produto completados durante a *Sprint* e o valor dos incrementos de todas as *Sprints* anteriores. Um incremento é uma parte principal inspecionável de trabalho pronto, que suporta empirismo no final da*sprint*. É um passo na direção de uma visão ou de um objetivo e deve estar na condição de ser utilizado independente do *Product Owner* decidir por liberá-lo ou não.

A Figura 2 apresenta uma visão geral simplificada dos conceitos e atividades do *Scrum*.

Figura 2 - Framework Scrum



Fonte: Adaptado de Marques (2016, fig.2.8).

[4] A definição de "Pronto" para um incremento, pode ser parte da organização, padrões e diretrizes de desenvolvimento da organização, então todos os times *Scrum* devem segui-la, mas caso não seja, cada time *Scrum* deve criar a sua definição de pronto.

1.1.2 *Extreme Programming* (XP)

De acordo com Cohen, Lindvall e Costa (2004), o método ágil XP foi introduzido por Beck e Jeffries et al. em 1998 (The C3 Team, 1998) e ainda mais popularizado por Beck, no livro *Extreme Programming Explained: Embrace Change*, em 1999.

Segundo Beck (2002), o XP é "uma metodologia leve para times de tamanho pequeno a médio, que desenvolva software em face a requisitos vagos que se modificam rapidamente".

O XP é considerado por Pressman (2010) como um dos modelos de engenharia de software ágil, que combina filosofia com um conjunto de princípios de desenvolvimento.

O ciclo do processo XP baseia-se em 4 atividades: planejamento, projeto, codificação e testes (Figura 3).

No planejamento são criadas as histórias de usuário, que é o escopo de trabalho do ciclo, que normalmente, é dimensionado para no máximo 3 semanas de trabalho da equipe de desenvolvimento. As histórias são subdivididas e cada subdivisão recebe uma prioridade diferente.

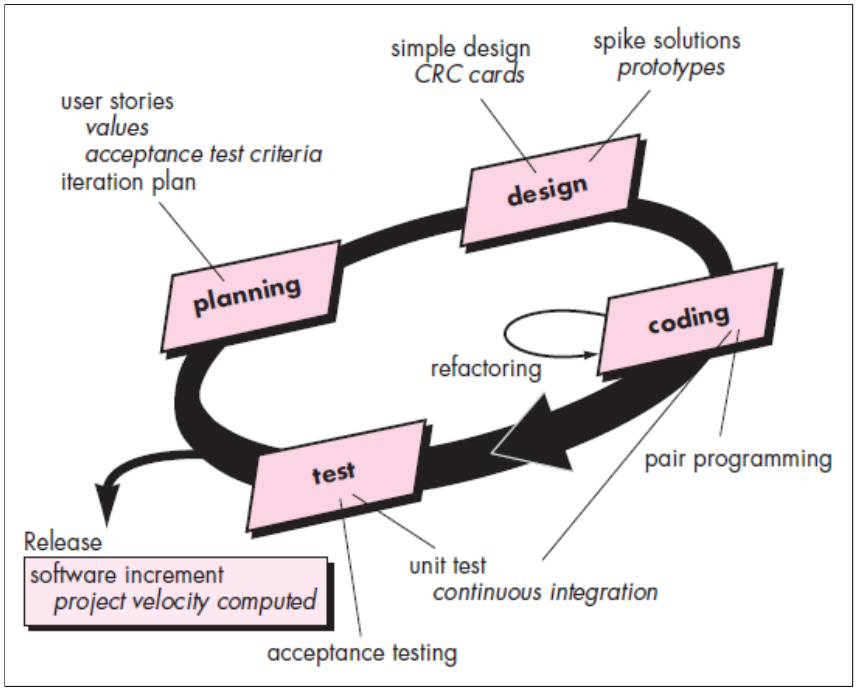
Na etapa de projeto é criado um protótipo do que é o objetivo do ciclo, esse protótipo não possui funcionalidades prontas, mas tem a função de guiar a próxima fase de desenvolvimento. Esse protótipo pode ser validado com o cliente para se alinhar às intenções dele.

A etapa de codificação tem como característica a programação em pares, onde uma pessoa avalia se as regras de codificação estão corretas e a outra se concentra no código a ser criado. A dupla pode trocar de posições durante o processo de codificação para gerar diversas verificações diferentes sobre o que está sendo criado.

A fase de testes se divide em duas partes: primeiro são feitos teste unitários que têm como objetivo avaliar cada função do programa com diferentes entradas e avaliar se as funções irão retornar o valor esperado. Em seguida, são feitos os testes com o cliente, que em ambiente de produção testa a ferramenta e gera *feedbacks* sobre o programa que irá realimentar o próximo ciclo.

O XP possui como valores: simplicidade, transparência, comprometimento, comunicação, foco, coragem e respeito.

Figura 3 - Processo XP, destacando suas 4 atividades principais



Fonte: Pressman (2010).

1.2 Agentes de *Softwares*

Segundo Wooldridge e Jennings (1995), o conceito de Agente tornou-se importante tanto na Inteligência Artificial (IA) quanto na corrente principal da Ciência da Computação. Na literatura existem algumas definições sobre Agentes de Software que ajudam no seu entendimento:

> Agentes são sistemas computacionais que habitam algum ambiente dinâmico complexo, percebem e agem autonomamente neste ambiente e fazendo isto, realizam o conjunto de objetivos e tarefas para os quais foram projetados (MAES, 1995).
>
> Agentes são sistemas computacionais situados em um ambiente, sendo capazes de realizar ações autônomas e flexíveis no intuito de alcançar seus objetivos (JENNINGS; SYCARA; WOOLDRIDGE, 1998).
>
> Agentes são entidades autônomas que podem perceber e agir sobre seu ambiente (RUSSELL; NORVING, 2002).

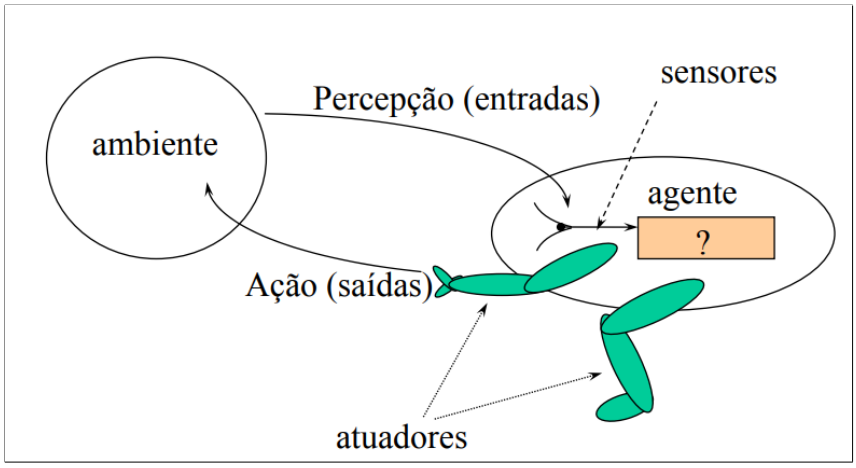
Segundo Russell e Norving (2002), para realizar a percepção e a ação, um agente possui um conjunto de capacidades. Os recursos podem ser usados para capturar habilidades suaves (ou seja, algoritmos), ou habilidades difíceis (ou seja, sensores físicos ou atuadores).

As capacidades podem ser definidas como: (i) um conjunto de sub capacitações, (ii) um conjunto de ações que podem interagir com o ambiente ou (iii) um plano que usa ações de maneiras específicas.

A Figura 4 apresenta uma representação dos elementos básicos de um agente, com

sensore, que recebem estímulos do ambiente no qual ele está inserido, realiza alguma atividade e responde com uma ação ao ambiente.

Figura 4 - Mecanismo de interação do agente com o ambiente



Fonte: Russell e Norving (2002).

Como exemplo, podemos imaginar um agente que realiza a atualização de uma aplicação no servidor de forma autônoma, mas que depende da disponibilização de um arquivo no servidor para que possa acionar a atualização. Ao detectar o arquivo o agente atua na atualização da aplicação e ao terminar volta ao estado de alerta, até receber algum novo estímulo.

As ações de um agente estão ligadas a pré-condições, que definem as situações nas quais o agente pode interagir. Com este exemplo, podemos descrever o ciclo de vida de um agente, que é representado desde o momento em que o agente percebe a alteração no ambiente e realiza a ação de acordo com o seu objetivo.

Russell e Norving (2002) sugerem a seguinte classificação de propriedades do ambientes: Acessível versus Inacessível, Determinística versus Não Determinística, Estático versus Dinâmico e Discreto versus Contínuo.

O ambiente Acessível é aquele em que o agente pode obter informações completas, precisas e atualizadas sobre o estado do ambiente. A maioria dos ambientes do mundo real (incluindo, por exemplo, mundo físico e da Internet) não são acessíveis neste sentido.

O ambiente Determinístico é um ambiente no qual a ação tem um único efeito garantido – não há incerteza sobre o estado que resultará da execução de uma ação, que por sua vez, seria o não determinístico.

Já o ambiente estático é aquele que pode ser assumido como se permanecesse inalterado, exceto pelo desempenho de ações do agente. Em contraste, um ambiente dinâmico é aquele que tem outros processos operando nele, e que, portanto, muda de maneira além do controle do agente. O mundo físico é um ambiente altamente dinâmico, assim como a Internet.

Discreto versus Contínuo: um ambiente é discreto se houver um número de ações e percepções fixas ou finitas. Em síntese, a classe geral de ambientes mais complexos são

aquelas inacessíveis, não determinísticas, dinâmicas e contínuas. Ambientes que possuem essas propriedades são frequentemente referidos como abertos.

Embora esses conceitos sejam empregados pelos pesquisadores da área, eles são bastante abrangentes, e não delimitam o escopo do termo agente. De acordo com Shehory e Sturm (2014) existe também, a definição de alguns atributos que os agentes possuem: autonomia, inteligência, sociabilidade e mobilidade. Esses atributos estão definidos a seguir.

Autonomia é uma das propriedades mais importantes e distintas do agente. Define que os agentes operem sem a intervenção humana e possuam algum tipo de controle sobre suas ações e seu estado interno. Na autonomia, os agentes funcionam sem a intervenção direta de humanos ou outros, e possuem algum tipo de controle sobre suas ações e estado interno.

A estratégia de agentes inteligentes pode ser usada de forma híbrida em conjunção com outras técnicas de IA.

A Sociabilidade de um agente está principalmente associada à comunicação, que deve permitir o envio, recebimento e processamento de mensagens. Essa troca é apoiada por linguagens de comunicação específicas, formatos de mensagens e protocolos para definir os meios e estratégias utilizadas pelos agentes para interagir com outros agentes e humanos.

A Mobilidade é associada à capacidade dos agentes de alterar sua localização lógica ou física. Uma forma de manifestação de mobilidade é que o agente se mova de seu ambiente de execução atual, (o *host*), para outro ambiente de execução. Outra dimensão da mobilidade é encontrada nos casos em que os agentes residem em dispositivos móveis, como *smartphones*.

Nesses casos, mesmo que o agente nunca saia de seu hospedeiro, o próprio dispositivo de hospedagem pode se movimentar, mudando a localização e as condições ambientais. Como resultado, o agente que executa no dispositivo se movimenta e também altera suas condições ambientais.

Em resumo, agentes são autônomos e interativos, baseados em estados internos e externos. Suas atividades incluem planos e condições que guiam a execução de tarefas, podendo agir só ou com outros agentes. Para definir e especificar todas essas dimensões dos sistemas multiagentes existem várias metodologias listadas na literatura.

A metodologia O-MaSE, foco deste trabalho, é apresentada a seguir.

1.2.1 Metodologia O-MaSE (*Organization - based Multiagent Systems Engineering*)

De acordo com Deloach e Garcia-Ojeda (2014) a Engenharia de Software Multiagente baseada em Organização, O-MaSE, é uma nova abordagem na análise e projeto

de sistemas baseados em agentes, por ser projetada desde o início como um conjunto de fragmentos de método para serem usados em um *framework* de engenharia de método.

O objetivo do O-MaSE é permitir que os designers criem processos de desenvolvimento de software orientados a agentes personalizados.

O O-MaSE consiste em três estruturas básicas: um metamodelo, um conjunto de fragmentos de método e um conjunto de diretrizes. O metamodelo O-MaSE define os conceitos-chave necessários para projetar e implementar sistemas multiagentes. Os fragmentos de método são tarefas executadas para produzir um conjunto de produtos de trabalho, que podem incluir modelos, documentação ou código. As diretrizes definem como os fragmentos do método estão relacionados entre si.

O-MaSE também fornece um conjunto de Diretrizes de Construção de Método, que estabelece como os fragmentos do método O-MaSE podem ser combinados para formar processos compatíveis com O-MaSE.

O Quadro 2 mostra as diretrizes de construção do método para as tarefas do O-MaSE. Essas Diretrizes de Construção de Método são definidas em termos de uma pré-condição e uma pós-condição.

Segundo Deloach e Garcia-Ojeda (2014) o O-MaSE foi projetado do zero, ou seja, não partiu de outra metodologia. E possui um conjunto de fragmentos que podem ser montados pelos desenvolvedores para atender aos requisitos específicos de seu projeto.

O-MaSE não se compromete com um conjunto predefinido de Fases. Em vez disso, define explicitamente as Atividades e Tarefas - Quadro 3 e permite que os engenheiros de método organizem as Atividades de diferentes maneiras com base na necessidade do projeto.

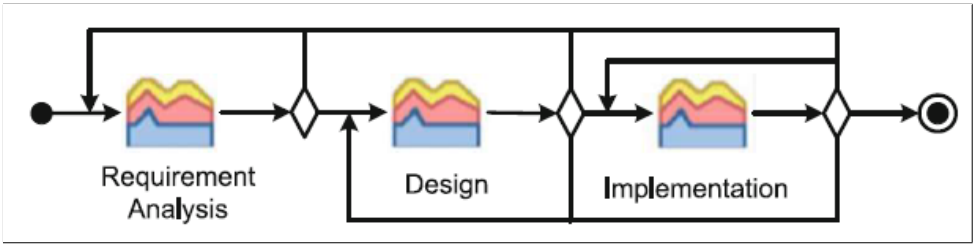
Ainda segundo Deloach e Garcia-Ojeda (2014), o fato de o O-MaSE não se comprometer com nenhum conjunto específico de fases, causava um problema por não ter um modelo documentado para descrição de seus processos de design no Modelo de Documentação do Processo de Design (DPDT) criado por Cossentino (2012).

Para minimizar este problema, na literatura foi assumido uma abordagem tradicional em cascata, Figura 5, com uma visão geral do O-MaSE definidas três fases principais: Análise de Requisitos, Projeto e Implementação. Uma visão geral das atividades do O-MaSE está descrita no Quadro 3.

Ao utilizar o O-MaSE em um projeto real, o designer de processo é livre para definir seu próprio conjunto de Fases e Iterações e atribuir Atividades e Tarefas, conforme apropriado.

Como essa adaptação é exclusiva para cada sistema que está sendo desenvolvido, não há regras rígidas sobre quais atividades devem ser colocadas em quais fases.

Figura 5 - *Frameworks* O-MaSE



Fonte: Deloach e Garcia-Ojeda (2014).

Quadro 2 - Diretrizes de construção do método

Task	Pre-condition	Post-condition
Requirements Specification	True	Requirements Spec
Model Goals	Requirements Spec ∨ ((Goal Model ∨ GMoDS) ∧ Role Model)	Goal Model
Refine Goals	Goal Model	GMoDS
Model Domain	Requirements Spec	Domain Model
Model Organization Interfaces	Requirements Spec ∧ GMoDS	Organization Model
Model Roles	GMoDS ∧ Organization Model	Role Model
Define Roles	Role Model	Role Description
Model Agent Classes	GMoDS ∨ Role Model ∨ Organization Model	Agent Class Model
Model Protocols	Role Model ∨ Agent Class Model	Protocol Model
Model Policies	GMoDS ∨ Organization Model ∨ Role Description ∨ Agent Class Model	Policy Model
Model Plans	(GMoDS ∧ Role Model) ∨ (GMoDS ∧ Agent Class Model)	Plan Model
Model Capabilities	Role Model ∧ Agent Class Model ∨ Domain Model	Capability Model
Model Actions	Capability Model ∧ Domain Model	Action Model
Code Generation	(Plan Model ∨ Protocol Model) ∧ (Capability Model ∨ Action Model)	Source Code

Fonte: Deloach e Garcia-Ojeda (2014).

O metamodelo O-MaSE define os principais conceitos e relacionamentos usados para definir sistemas multiagentes. É baseado em uma abordagem organizacional e inclui noções que permitem a decomposição hierárquica, holônica[5] e baseada em equipe das organizações.

O metamodelo O-MaSE foi derivado do Modelo de Documentação do Processo de Design Modelo de Organização para Sistemas Computacionais Adaptativos (OMACS) de Deloach e Miller (2010), que captura o conhecimento necessário da estrutura organizacional e dos recursos de um sistema para permitir a organização e reorganização em tempo de execução.

A decisão chave nos sistemas baseados no OMACS é qual agente atribuir a qual

[5] Holônica é uma palavra de origem inglesa (holonic) e seu significado no texto é definir um tipo de decomposição como parte de um todo ou uma decomposição adaptativa.

Quadro 3 - Visão geral do O-MaSE

Entity	Task	Work Product	Role
Requirements Gathering	Requirements Specification	Requirements Spec	Requirements Engineer
Problem Analysis	Model Goals	Goal Model	Goal Modeler
	Refine Goals		
	Model Domain	Domain Model	Domain Modeler
Solution Analysis	Model Organization	Organization Model	Organization Modeler
	Interfaces		
	Model Roles	Role Model	Role Modeler
	Define Roles	Role Description Document	
	Define Role Goals	Role Goal Model	
Architecture Design	Model Agent Classes	Agent Class Model	Agent Class Modeler
	Model Protocols	Protocol Model	Protocol Modeler
	Model Policies	Policy Model	Policy Modeler
Low Level Design	Model Plans	Agent Plan Model	Plan Modeler
	Model Capabilities	Capabilities Model	Capabilities Modeler
	Model Actions	Action Model	Action Modeler
Code Generation	Generate Code	Source code	Programmer

Fonte: Deloach e Garcia-Ojeda (2014).

função para atingir qual objetivo. Como mostra a Figura 6, uma organização é composta por seis tipos de entidades: metas, funções, agentes, agentes organizacionais, um modelo de domínio e políticas. O Quadro 4 descreve cada uma dessas entidades.

Embora uma variedade de interpretações sutis de metas exista na inteligência artificial e nas comunidades de agentes, O-MaSE define uma meta como um objetivo da organização, que geralmente é descrito em termos de algum estado do mundo desejado.

Um papel/função define uma posição dentro de uma organização cujo comportamento se espera que atinja uma determinada meta ou conjunto de metas.

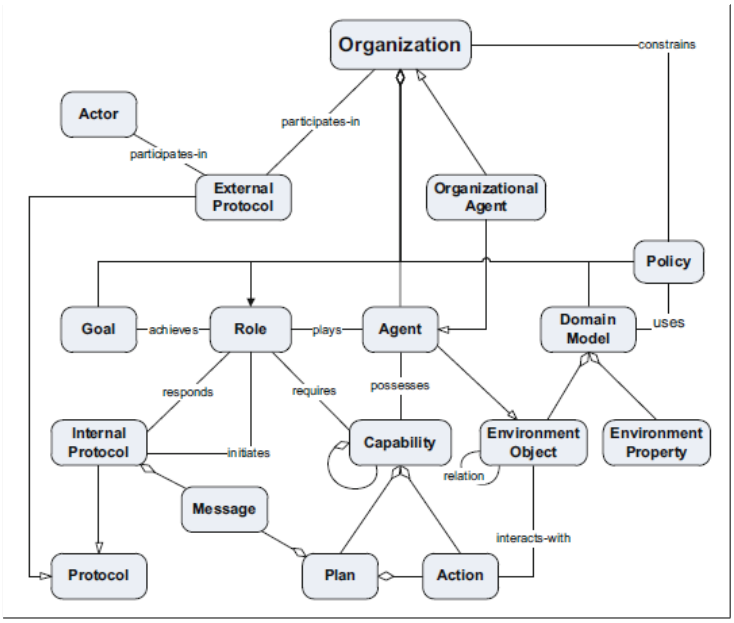
O Modelo de Domínio é usado para capturar os principais elementos do ambiente no qual os agentes irão operar. Esses elementos são capturados como Tipos de objeto de domínio do ambiente, que inclui agentes e os relacionamentos entre esses tipos de objeto. Também pode ser usado para capturar as propriedades gerais do ambiente que podem descrever como os objetos se comportam e interagem.

Um *designer* pode usar entidades definidas no modelo O-MaSE (objetivos, funções, agentes, etc.) junto com entidades definidas no Modelo de Domínio para especificar políticas organizacionais para restringir como uma organização pode se comportar em uma situação particular.

As políticas são frequentemente usadas para especificar as propriedades de vivaci-

dade[6] e segurança[7] do sistema que está sendo projetado.

Figura 6 - Metamodelo O-MaSE



Fonte: Deloach e Garcia-Ojeda (2014).

Quadro 4 - Entidades de metamodelo

Entity	Definition
Goal	A desirable state; goals capture organizational objectives
Role	Capture behavior that achieves a particular goal or set of goals
Agent	Autonomous entities that perceive and act upon their environment; agents play roles in the organization
Organizational Agent	A sub-organization that functions as an agent in a higher-level organization
Capability	Soft abilities (algorithms) or hard abilities of agents
Domain model	Captures the environment including objects and general properties describing how objects behave and interact
Policy	Constrain organization behavior often in the form of liveness and safety properties
Protocol	Define interaction between agents, roles, or external Actors; they may be internal or external
Actor	Actors that exist outside the system and interact with the system
Plan	Abstractions of algorithms used by agents; plans are specified in terms of actions with the environment and messages in protocols

Fonte: Deloach e Garcia-Ojeda (2014).

[6] Propriedades de vivacidade (*liveness*) - expressam "comportamentos que queremos que (sempre) ocorram", em outras palavras essa propriedade estabelece a necessidade ou a possibilidade de que "comportamentos desejáveis que queremos que aconteçam".

[7] Propriedades de segurança (*safety*) - expressam "comportamentos que não queremos que ocorram", em outras palavras "comportamentos indesejáveis não acontecem". Essa propriedade se relaciona com a invariância de uma determinada característica, como por exemplo, a ausência de deadlock.

Os protocolos definem as interações entre funções ou entre a organização e atores externos. Os protocolos são geralmente definidos como padrões de comunicação entre tais entidades. Um protocolo pode ser de dois tipos, Externo ou Interno. Os protocolos externos especificam as interações entre a organização e os atores externos (ou seja, humanos ou outros aplicativos de software). Os protocolos internos especificam as interações entre os agentes que desempenham funções específicas na organização. Tanto as mensagens quanto as ações podem ser usadas para definir os protocolos. As mensagens são normalmente usadas para comunicações; no entanto, as ações podem ser usadas para modificar o ambiente como um meio de comunicação.

A primeira etapa do uso do O-MaSE para definir um sistema, é estabelecer um processo de trabalho que seja compatível com o O-MaSE. Para isto, a abordagem mais simples é realizar uma análise ascendente dos produtos de trabalho necessários para produzir o sistema desejado. E nesse caso, a decisão principal é que tipo de sistema precisamos desenvolver e quais são os produtos finais de trabalho, necessários para apoiar a implementação desse sistema.

A partir daí, trabalhamos para trás para determinar quais outros produtos de trabalho são necessários para produzir os produtos finais.

Na fase de Análise de Requisitos, pode-se obter os produtos de trabalho, conforme mencionado no Quadro 5.

Existem muitas maneiras de capturar e categorizar requisitos para uso em sistemas. O O-MaSE assume que as técnicas de coleta de requisitos com foco em multiagentes ou tradicionais, são suficientes e portanto, não estipulam uma estrutura de documento específica.

Quadro 5 - Produtos de trabalho da fase de análise de requisitos

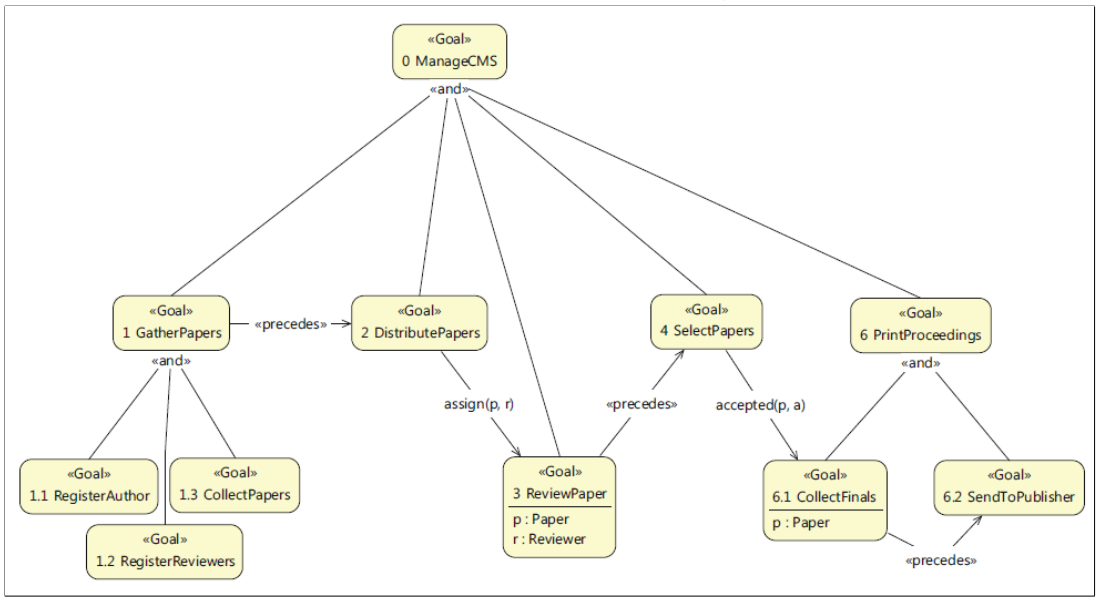
Name	Description	Kind
System Description Specification	describes the technical requirements for a particular agent-oriented software	Structural
Goal Model	captures the purpose of the organization as a goal tree; includes goal attributes, precedence and triggering relationships	Behavioral
Domain Model	defines the language that can be used when defining the operation of the system	Structural
Organizational Model	documents the interaction between the organization and the external actors	Structural
Role Model	depicts organization roles, the goals they achieve and interactions between roles/external actors	Structural

Fonte: Deloach e Garcia-Ojeda (2014).

As figuras 7 e 8 são exemplos de modelos da fase de Análise de Requisitos. O modelo de Meta ou Objetivo, Figura 7, captura o propósito da organização como um objetivo Árvore comportamental: inclui atributos de meta, precedência e relações de acionamento.

No exemplo da Figura 7 é apresentada a versão inicial do modelo de Meta/Objetivo *Conference Management System* (CMS), cujo o objetivo geral do Gerenciar CMS é dividido em cinco sub-objetivos: Reunir Documentos, Distribuir Artigos, Revisar Artigos, Selecionar Artigos e Imprimir Procedimentos.

Figura 7 - Exemplo da versão inicial do Modelo de Meta/Objetivo



Fonte: Deloach e Garcia-Ojeda (2014).

As metas de Coleta de Documentos são decompostas posteriormente em duas submetas, Registrar Autor e Coletar Artigos, enquanto a meta Imprimir Procedimentos também é decomposta nas submetas Coleta Final e Enviar ao Editor.

O CMS permite que novas instâncias de objetivo sejam criadas por meio de um acionador de evento, que é denotado por uma seta entre dois objetivos rotulados por uma assinatura de evento, como assign (p, r) na seta entre os objetivos Distribuir Artigos e Artigo de Revisão.

Nesse caso, quando uma atribuição ocorre durante a busca da meta do Distribuir Artigos, uma nova meta do Artigo de Revisão é criada. Usamos isso para criar uma meta de Artigo de Revisão para cada revisor designado para revisar um determinado artigo.

Da mesma forma, o evento aceito durante a busca da meta do Selecione o Artigo criará uma nova meta do Coletar Finais, para coletar cada artigo aceito para o processo.

A relação de precedência GMoDS (denotada por uma seta marcada com «precede») é usada no modelo de objetivo para garantir a sequência adequada de ações no sistema.

Assim, uma vez que os artigos devem ser reunidos antes de serem distribuídos para revisão, o objetivo Reunir Documentos precede ao objetivo Distribuir Artigos (ou seja, Reunir Documentos deve ser alcançado antes que Distribuir Artigos possa começar a sua sequência).

Da mesma forma, todas as revisões devem ser realizadas (objetivo Revisar Artigos) antes que os artigos possam ser selecionados (Selecionar Artigos) para a conferência, e

todos os objetivos Coletar Finais devem ser alcançados antes que a meta Enviar ao Editor seja realizada.

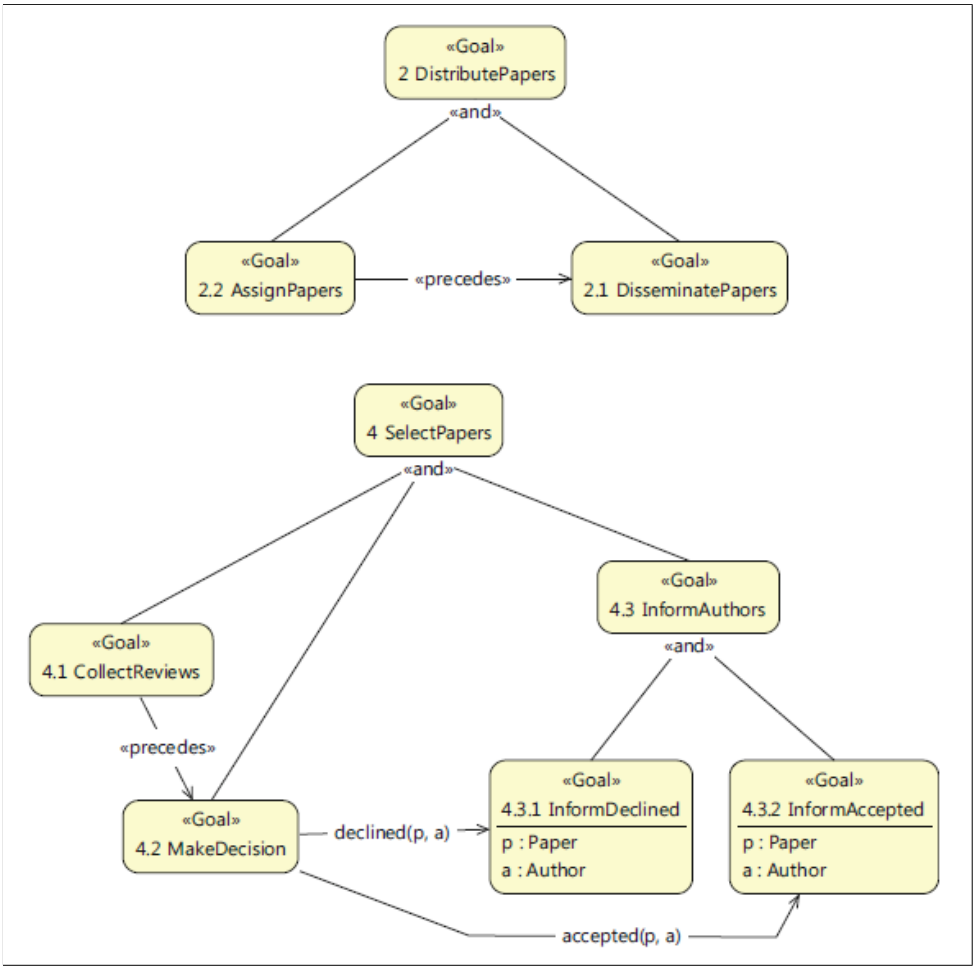
A organização do PC é, na verdade, projetada para atingir três objetivos de nível superior: Enviar ao Editor, Distribuir Artigos e Selecionar as Metas.

Os objetivos Distribuir Documentos e Selecionar Documentos são decompostos posteriormente, conforme mostrado na Figura 8. A meta Distribuir Artigos é decomposta nas metas Atribuir Artigos e Divulgar-Artigos.

Embora a descrição do CMS forneça várias maneiras de atribuir papéis, há apenas um único objetivo que orienta esse processo de atribuição.

O modo como o PC realmente atinge o objetivo da atribuição é definido pelo processo que eles usam. Portanto, este aspecto da definição do CMS deve ser capturado como um processo, que no caso do O-MaSE seria capturado como variações de um plano.

Figura 8 - Exemplo do Modelo de Meta/Objetivo refinado



Fonte: Deloach e Garcia-Ojeda (2014).

O objetivo do Selecione os Artigos é decomposto em vários sub-objetivos: Colete Avaliações, Tome Decisões e Informe os Autores.

Uma vez que todas as revisões devem ser coletadas antes da tomada de uma decisão, existe uma relação de precedência entre as metas Coletar Revisões e Tomar Decisão.

À medida que uma decisão é tomada em cada artigo, um evento, que é recusado ou aceito, aciona uma meta Informar Recusado, ou Informar Aceito para aquele artigo. Além disso, um evento aceito aciona uma meta Coletar Finais no modelo de meta CMS.

O modelo de Domínio, Figura 9, define a linguagem que pode ser usada pelo designer para garantir que todos falem sobre a mesma coisa.

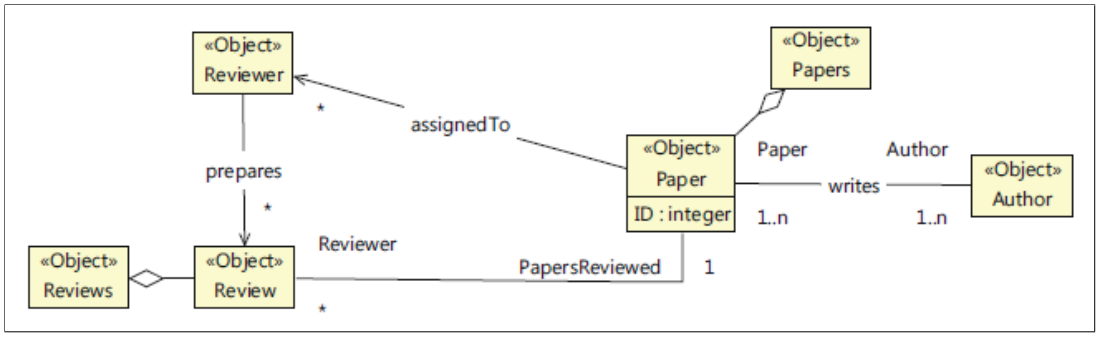
Também é essencial para definir formalmente a operação do sistema, desde os atributos no modelo de Meta/Objetivo até as informações passadas por meio de protocolos do sistema para as políticas do sistema.

No exemplo existem quatro objetos principais de interesse no domínio CMS: Autor, Artigo, Revisor e Revisão. Cada um deles e seus relacionamentos são mostrados explicitamente no Modelo de Domínio. Além disso, uma agregação de artigos individuais (Artigos) e resenhas (Revisões) também é definida.

O modelo usa a notação *Unified Modeling Language* (UML) como padrão, para definir as multiplicidades permitidas entre objetos. Por exemplo, um Autor deve ter pelo menos um Autor, e para ser válido, um Autor deve ter pelo menos um Artigo.

Também mostra que um revisor prepara uma revisão, e cada revisão tem exatamente um artigo que pode ser sobrescrito. O Modelo de Domínio permite a definição dos atributos do Objeto. Conforme mostrado, cada objeto Paper possui um ID.

Figura 9 - Exemplo de Modelo de Domínio

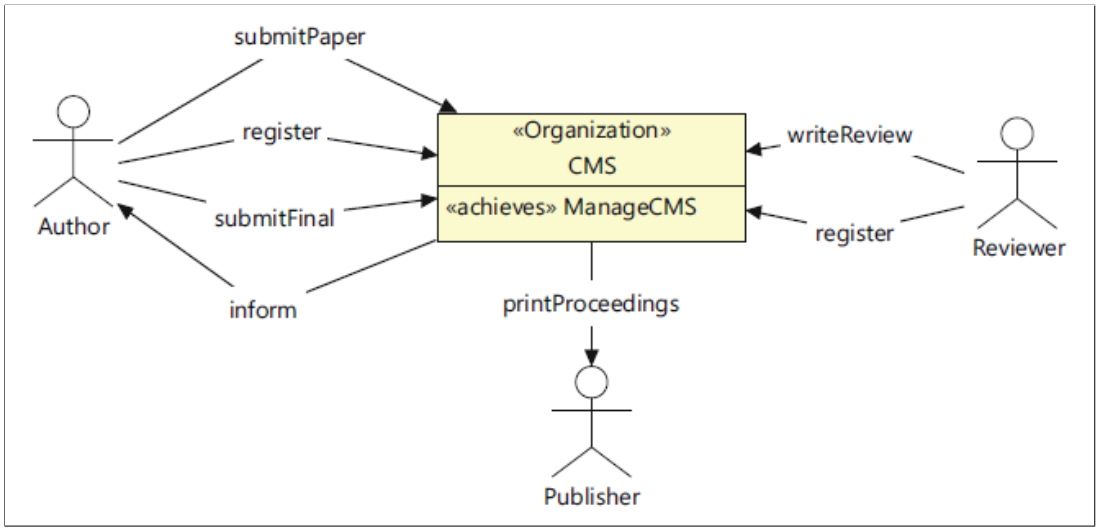


Fonte: Deloach e Garcia-Ojeda (2014).

No exemplo do Modelo de Organização para o CMS, Figura 10, é mostrado que a decisão de tornar o PC uma sub organização aparece na ausência do PC como um ator externo.

Como descrito, uma vez que o PC é uma sub organização, estamos essencialmente considerando-o como parte do sistema nesse nível. Os três atores externos mostrados, Autor, Editor e Revisor, todos interagem com o sistema por meio dos protocolos fornecidos.

Figura 10 - Exemplo de Modelo de Organização



Fonte: Deloach e Garcia-Ojeda (2014).

À medida que a organização é refinada em Modelos de Função e Classe de Agente, esses atores externos e protocolos devem aparecer de maneira consistente.

O modelo de Papel, Figura 11, mostra que existem seis papéis definidos para realizar os objetivos definidos no modelo de Metas/Objetivos. Cada papel é projetado para atingir um único objetivo, conforme indicado pelo atributo «atinge» em cada função. Duas exceções são a função Coletor de papel, que é projetada para coletar as cópias inicial e final dos documentos, e a função de Registrador, que é projetada para registrar autores e revisores.

A fase de Projeto consiste em duas atividades: Projeto de Arquitetura e Projeto de Baixo Nível. Uma vez que os objetivos, ambiente, comportamento e interações do sistema são conhecidos, o Projeto de Arquitetura é usado para criar uma descrição de alto nível dos principais componentes do sistema e suas interações.

A descrição de Alto Nível é então usada para conduzir o Projeto de Baixo Nível, onde a especificação detalhada do sistema interno o comportamento do agente é definido.

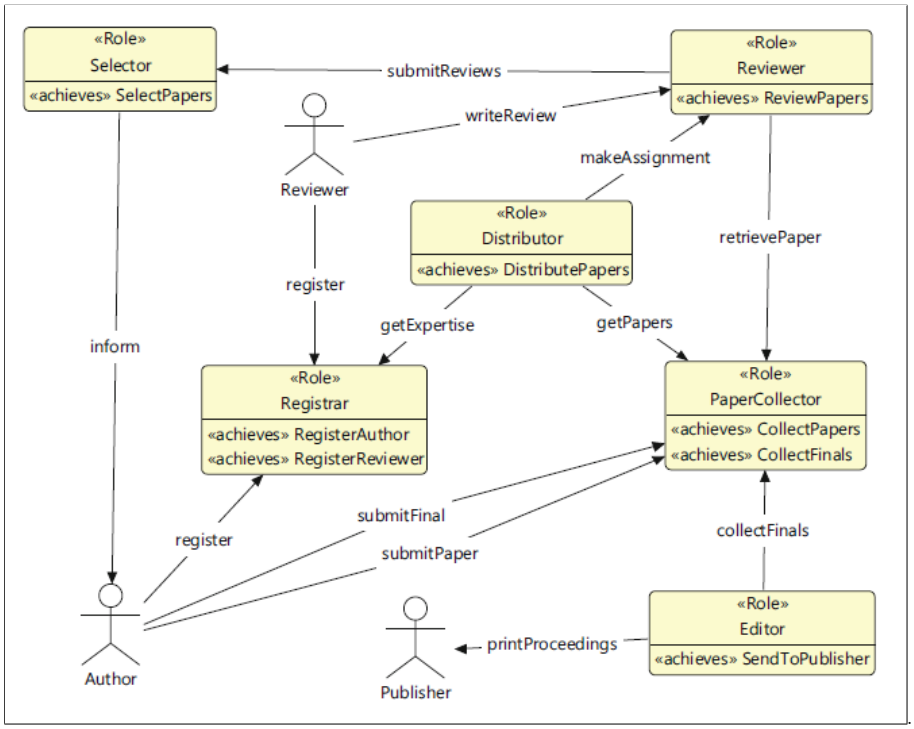
A especificação de Baixo Nível é então usada para implementar os agentes individuais durante a fase de Implementação.

O projeto de arquitetura consiste em três tarefas, conforme mostrado na Figura 12. A tarefa Modelar Classes de Agentes identifica os tipos de agentes na organização e os protocolos entre eles.

As classes de agentes podem ser definidas pelas funções que desempenham ou pelos recursos que possuem, o que define implicitamente as funções que podem desempenhar. Assim, uma classe de agente fornece um modelo para um tipo de agente no sistema.

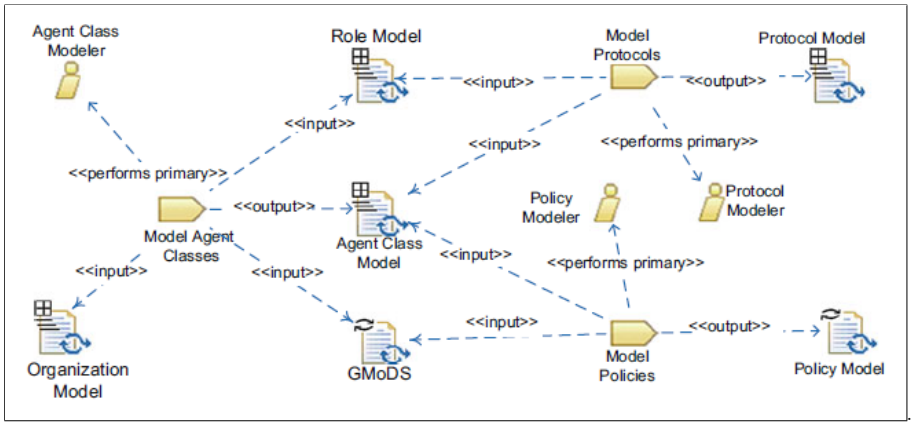
A tarefa Modelar Protocolos especifica os detalhes das interações entre agentes ou funções.

Figura 11 - Exemplo de Modelo de Papel



Fonte: DeLoach e Garcia-Ojeda, 2014

Figura 12 - Diagrama de Atividades do Projeto de Arquitetura



Fonte: Deloach e Garcia-Ojeda (2014)

Como os protocolos podem ser especificados em Modelos de Organização, Modelos de Função e Modelos de Classe de Agente, o engenheiro de método pode decidir qual conjunto de protocolos definir.

Se os protocolos do Modelo de Meta forem definidos por meio de modelos de protocolo, as classes de agente que desempenham essas funções devem herdar esses protocolos.

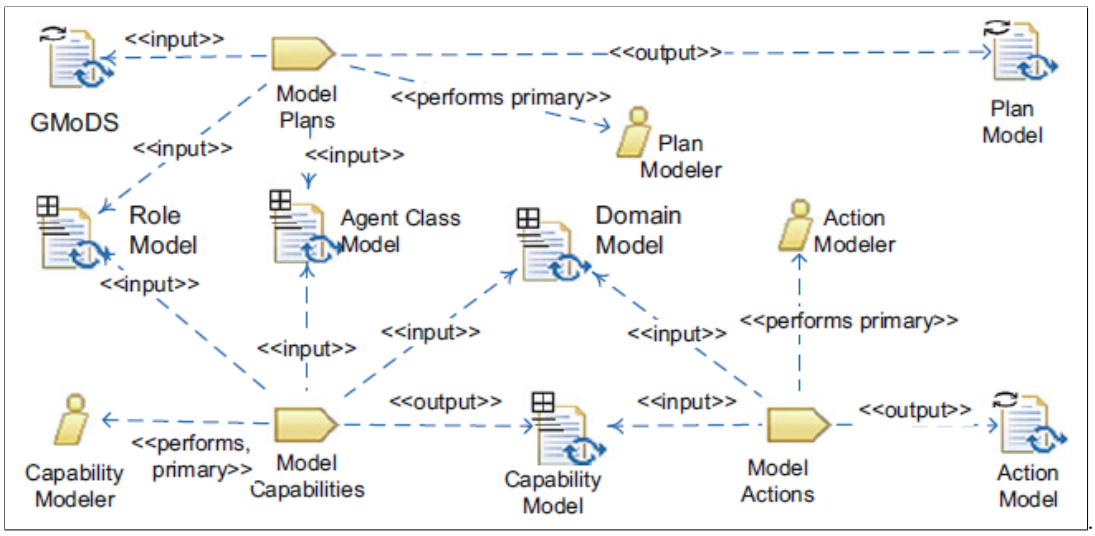
O Modelo de Protocolo define os tipos de mensagens enviadas entre as duas entidades e é semelhante aos modelos de interação UML.

A tarefa Modelar Políticas define um conjunto de regras que descrevem como uma organização deve se comportar.

Em geral, as políticas são usadas para restringir o comportamento do agente e podem ser aplicadas em tempo de projeto ou em tempo de execução. A forma como as políticas são aplicadas é uma decisão crítica que afeta a maneira como o Modelo de Política é usado durante o desenvolvimento.

O Projeto de Baixo Nível consiste em três tarefas: Modelar Capacidades, Modelar Ações e Modelar Planos, conforme mostrado na Figura 13.

Figura 13 - Diagrama de Atividade de Projeto de Baixo Nível



Fonte: Deloach e Garcia-Ojeda (2014)

A tarefa Modelar Capacidades define a estrutura interna das capacidades possuídas pelos agentes na organização, que podem ser modeladas como uma Ação ou um Plano. Uma ação é uma funcionalidade possuída por um Agente e definida usando a tarefa Modelar Ações. Um plano é uma definição algorítmica de um recurso e é definido usando a tarefa Modelar Planos.

Conforme o Quadro 6, existem seis produtos de trabalho produzidos na fase de Projeto: Modelo de Classe de Agente, Modelo de Protocolo, Modelo de Política, Modelo de Capacidade, Modelo de Plano e Modelo de Ação.

O modelo de classe de agente é mostrado na Figura 14. Existem três agentes e uma sub organização (um agente organizacional) definidos para implementar as seis funções

definidas no modelo de função.

Quadro 6 - Produtos do trabalho da fase de Projeto

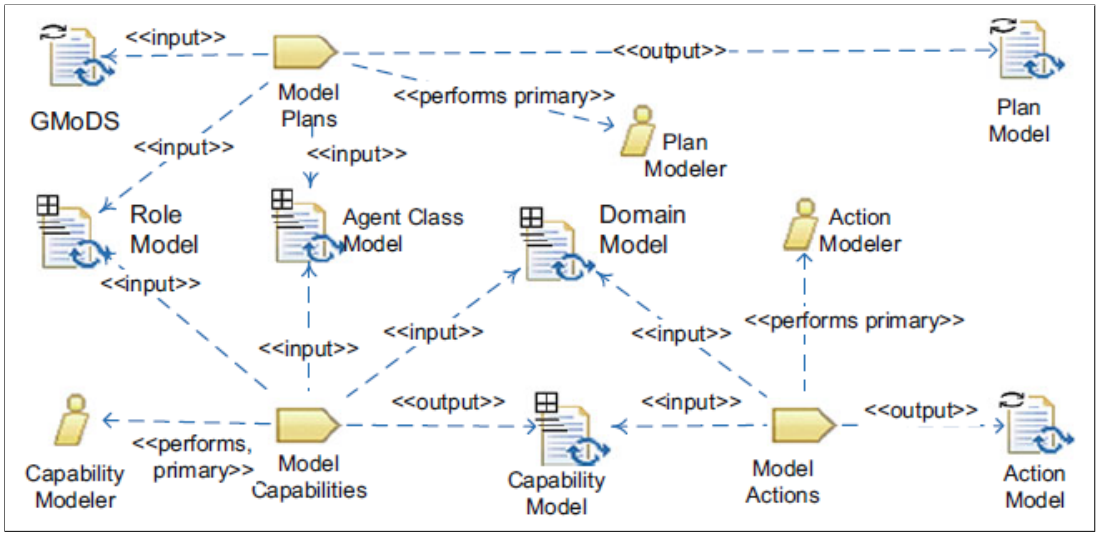
Name	Description	Kind
Agent Class Model	defines the agent classes and sub-organizations that will populate the organization.	Structural
Protocol Model	represents the different relations/interaction between external actors and agents/roles.	Structural
Policy Model	describes all the rules/constraints of the system	Behavioral
Capability Model	defines the internal structure of the capabilities possessed by agents in the organization.	Structural
Plan Model	captures how an agent can achieve a specific type of goal using a set of actions (which includes sending and receiving messages).	Behavioral
Action Model	defines the low-level actions used by agents to perform plans and achieve goals.	Behavioral

Fonte: Deloach e Garcia-Ojeda (2014).

O agente de banco de dados desempenha a função Coletor de papel, o agente de Registro desempenha a função de registrador e o agente Revisor desempenha a função de Revisor.

Os protocolos definidos no Modelo de Papel são cada um herdado pelo Modelo de Classe de Agente com base nas funções atribuídas aos vários agentes.

Figura 14 - Exemplo de Modelo de Classe de Agente



Fonte: Deloach e Garcia-Ojeda (2014).

Um exemplo de Modelo de Política é exibido na Figura 15, que usa uma linguagem que inclui fórmula temporal com quantificação.

A linguagem usada para especificar políticas vem de entidades definidas nos vários modelos, por exemplo, objetos definidos no Modelo de Domínio, os papéis definidos no Modelo de Papéis e os Agentes definidos no Modelo de Classe de Agente.

A atividade Modelar Protocolos define os detalhes internos de cada protocolo identificado nos modelos Papel e Classe de Agente. Neste ponto, todos os protocolos no CMS

ou na organização do PC são modelados.

Figura 15 - Exemplo de Modelo de Política

$$\forall p : Paper, m : Member, a : Author\ a = m \wedge writes(a, p) \Rightarrow m\neg \in assignedTo(p, m)$$

$$\forall p : Paper, m : Member\ submitReview(m, p) \Rightarrow m \in assignedTo(p, m)$$

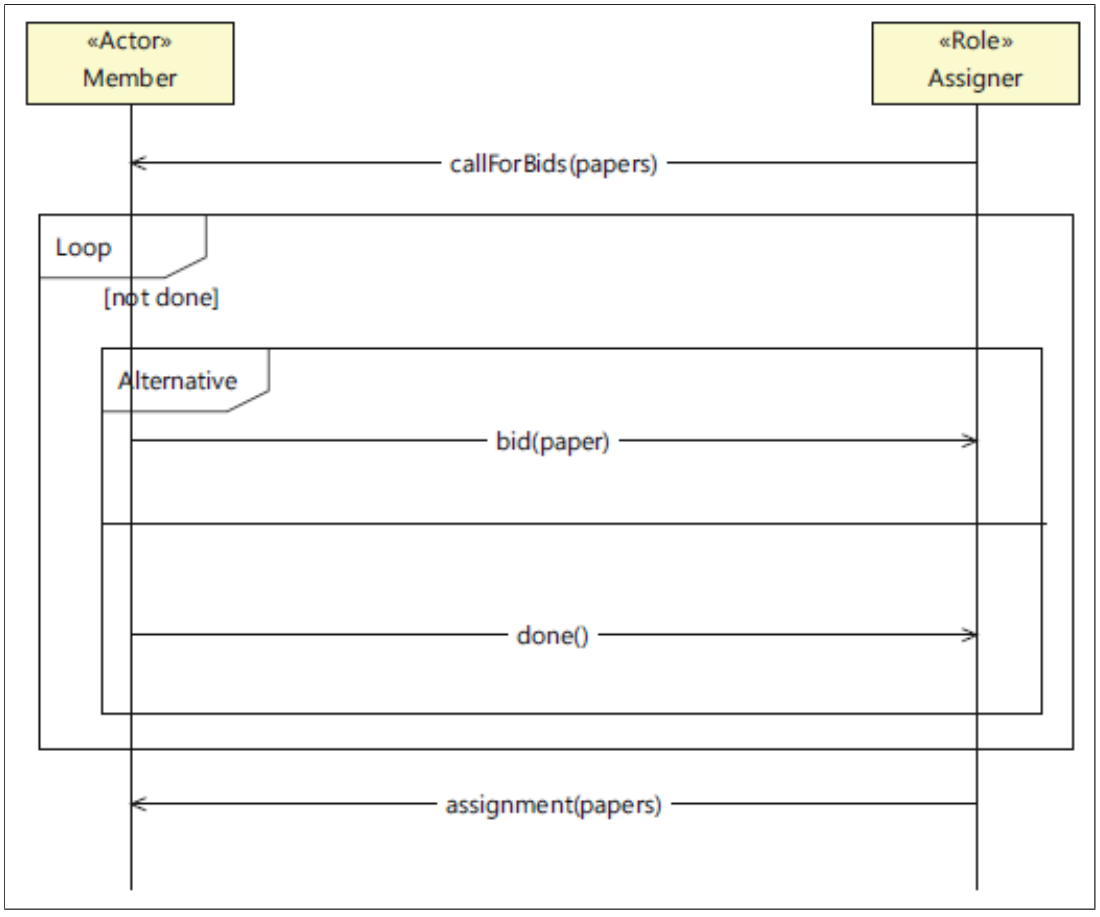
Fonte: Deloach e Garcia-Ojeda (2014).

Um exemplo de um Modelo de Protocolo é apresentado na Figura 16, que mostra o protocolo de licitação entre o ator Membro e o agente Cedente.

Primeiro, a função Cedente PC envia a mensagem de chamada de lances para o membro com uma lista dos papéis disponíveis. Em seguida, o Membro decide qual lance colocar em cada um dos papéis do conjunto de papéis recebidos.

Para cada papel em que o Membro gostaria de licitar, uma mensagem de lance é enviada de volta ao Designador.

Figura 16 - Exemplo de Modelo de Protocolo



Fonte: Deloach e Garcia-Ojeda (2014).

Quando o Membro tiver concluído a licitação dos papéis, ele envia uma mensagem concluída ao Designador.

Assim que o Designador receber todos os lances de todos os Membros, o Designador

decide as designações finais para cada Membro e uma mensagem de designação é enviada ao Membro e o protocolo é concluído.

O modelo de capacidade captura a estrutura interna das capacidades dos agentes da organização. Cada capacidade pode ser modelada como uma ação ou um plano.

Uma ação é uma funcionalidade possuída por um Agente e definida usando um Modelo de Ação.

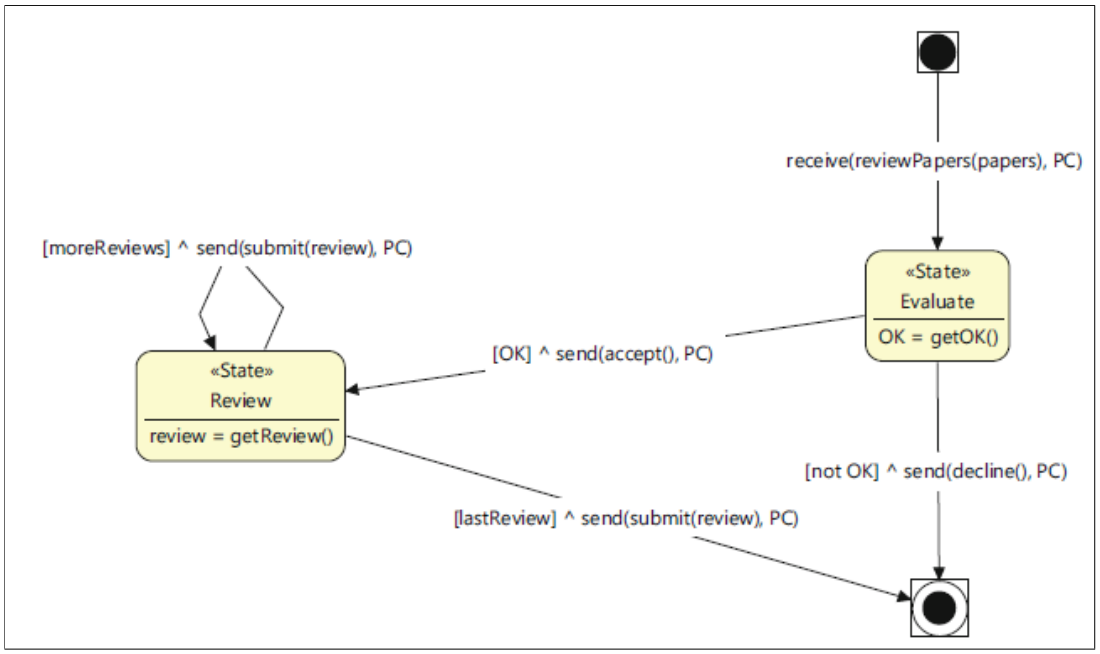
Um plano é uma definição algorítmica (definida por meio de uma máquina de estado) de um recurso que usa ações e implementa protocolos. Cada plano é definido usando um Modelo de Plano.

Normalmente, é necessário um plano para cada tipo de meta que um agente pode atingir. Assim, uma vez que os agentes são definidos pelos papéis que desempenham, deve-se olhar para os objetivos que podem ser alcançados por cada papel que o agente pode desempenhar.

Um exemplo está na Figura 17 com o agente Revisor. Uma vez que o modelo de classe de agente define que o agente Revisor só pode desempenhar a função de Revisor, só precisamos examinar os objetivos que podem ser alcançados pelo papel do Revisor.

No Modelo de Papéis podemos ver que a função de Revisor é projetada para atingir apenas a meta de Artigos de Revisão. Portanto, precisamos apenas definir um único plano para definir totalmente o comportamento do agente Revisor.

Figura 17 - Exemplo de Modelo de Plano de Revisão



Fonte: Deloach e Garcia-Ojeda (2014).

Finalizando a metodologia O-MaSE, o que foi projetado é traduzido em código. O objetivo desta fase é pegar todos os modelos projetados criados durante a fase de Projeto e convertê-los em código que os implemente corretamente.

Existem várias abordagens para a geração de código com base na plataforma de

tempo de execução e na linguagem de implementação escolhida.

Nesta fase existe um único Papel, o Programador que é responsável por escrever o código com base nos vários modelos produzidos durante a fase de Projeto.

A saída da tarefa Gerar Código é o código-fonte do aplicativo, embora não seja coberto atualmente no processo, a criação do sistema termina com o teste, avaliação e implantação do sistema.

1.2.2 Metodologia *Agile* PASSI (*Process for Agent Societies Specification and Implementation*)

De acordo com Chella et al. (2006), o *Agile* PASSI foi criado para dar uma abordagem mais versátil no design de software. A nova metodologia foi criada explorando experiências feitas com o PASSI convencional.

A abordagem foi realizada com a combinação de uma das principais metodologias ágeis, XP. Segundo Chella et al. (2006), foi decidido criar uma nova metodologia ágil, específica para os seus propósitos, onde o principal objetivo não era apenas identificar/-criar uma metodologia ágil, mas tinham vários requisitos específicos que deveriam ser respeitados.

Ainda de acordo com Chella et al. (2006), o método ágil XP, não foi uma boa solução porque violava pelo menos um de seus requisitos: a reutilização de experiências e habilidades que foram amadurecidas com o PASSI convencional.

O processo *Agile* PASSI é composto de quatro fases: requisitos, sociedade de agentes, código e testes, conforme exibido na Figura 18.

Na fase de Requisitos, o modelo de requisito do sistema é dividido em duas etapas: Planejamento e Descrição dos Requisitos do Subdomínio.

A fase de Sociedade de Agentes consiste numa visão dos agentes envolvidos na solução, suas interações e seu conhecimento sobre o mundo. É composto em duas etapas: Descrição da Ontologia de Domínio e Identificação do Agente.

Na fase de Código um modelo de domínio de solução é implementado no nível do código.

A fase de Testes consiste na criação do plano de teste antes da codificação e os Testes logo após a codificação.

Na etapa de Planejamento da fase de Requisitos, é realizada a preparação das iterações, privilegiando as comunicações entre a equipe de desenvolvimento de componentes, através de uma análise de riscos e requisitos.É realizado, por exemplo, com o auxílio das histórias de usuários, semelhantes aos cenários típicos da programação XP. Durante esta atividade, a equipe de desenvolvimento decide quais requisitos do sistema devem ser feitos e em qual ordem.

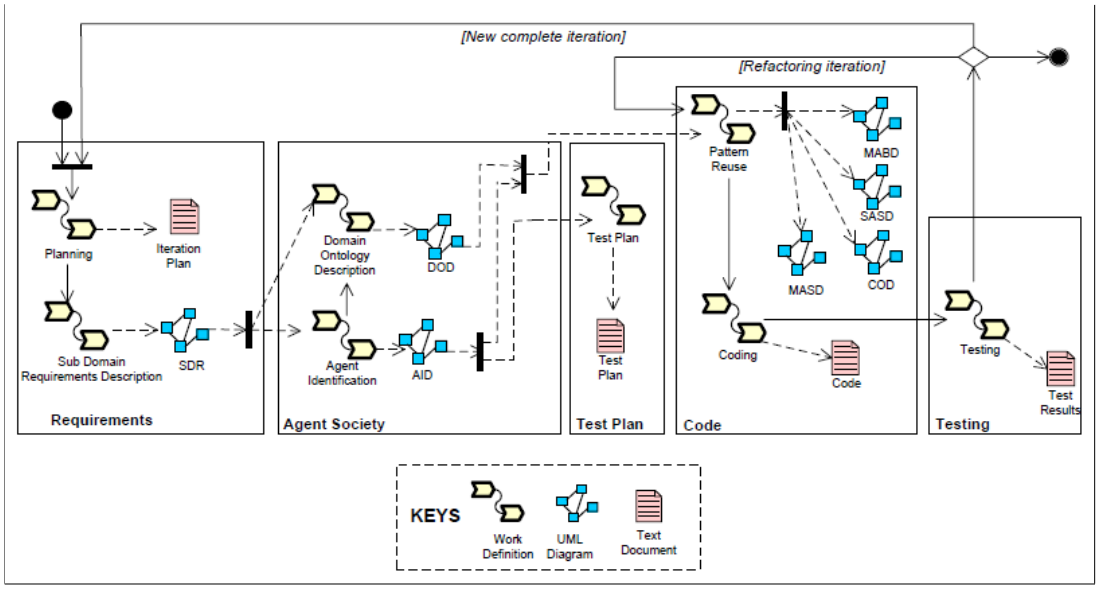
Na etapa Descrição dos Requisitos do Subdomínio, diagramas de casos de uso da

UML comuns, são usados para representar uma descrição funcional do sistema. O termo "sub" refere-se à oportunidade de dividir o problema inteiro em subproblemas. O resultado é uma decomposição do problema em vários problemas mais fáceis que serão realizados em iterações sequenciais.

Na fase de Sociedade de Agente, o desenvolvimento do modelo de sociedade de agentes, envolve duas atividades: identificação do agente e descrição da ontologia de domínio. A Identificação do Agente obtém como entrada os diagramas de casos de uso selecionados para a iteração atual. No diagrama de Identificação do Agente, os casos de uso devem ser particionados em grupos separados de responsabilidade (os agentes do sistema). No diagrama, isso significa agrupar um ou mais casos de uso em pacotes estereotipados com o rótulo do agente, conforme a Figura 19.

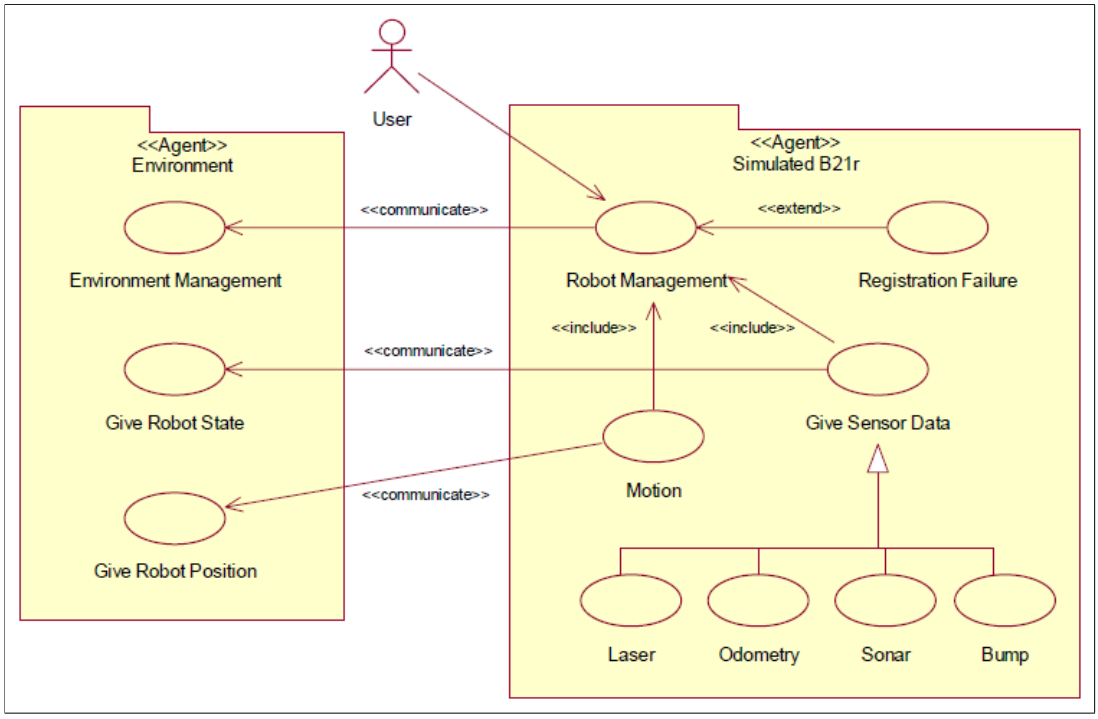
A atividade Descrição da Ontologia de Domínio, Figura 20, visa capturar o domínio no qual o sistema trabalha identificando entidades como classes UML.

Figura 18 - Processo Agile PASSI



Fonte: Chella et al. (2006).

Figura 19 - Diagrama de Identificação de Agentes



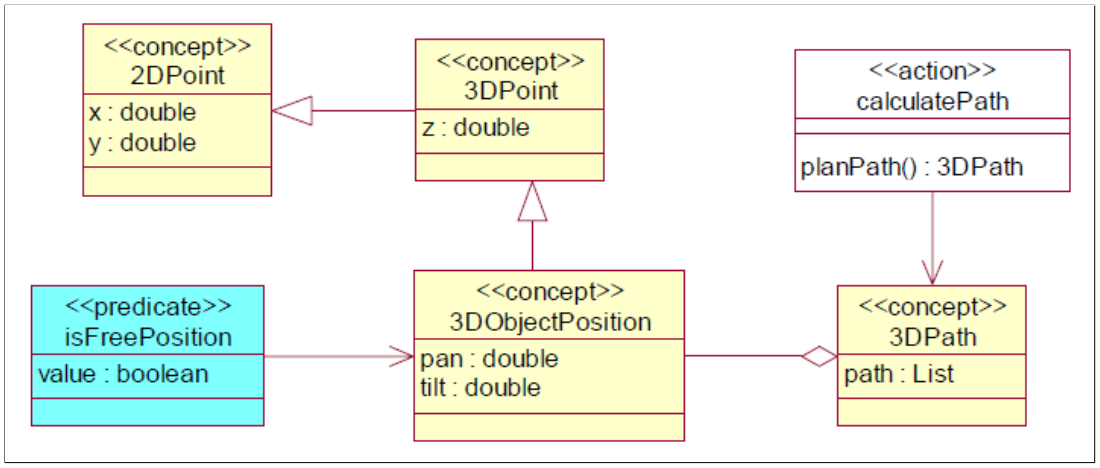
Fonte: Chella et al. (2006).

A ontologia é descrita em termos de conceitos (cor de preenchimento: amarelo), predicados (cor de preenchimento: azul claro) e ações (cor de preenchimento: branco). Os elementos da ontologia podem ser relacionados usando três relacionamentos padrão da UML: generalização, associação e agregação.

As duas atividades da Fase de Requisitos podem ser realizadas em paralelo, enquanto é decidido quais funcionalidades serão atribuídas a diferentes agentes, pode-se também identificar e representar os seus conhecimentos.

Na Fase de Código é construído o modelo de código, que inclui duas atividades: reutilização de padrões e codificação.

Figura 20 - Diagrama da Ontologia de Domínio



Fonte: Chella et al. (2006).

Na Reutilização de Padrões, os desenvolvedores tentam reutilizar os padrões de projeto dos agentes, obtendo partes do código reutilizável que é documentado com uma visão estrutural e comportamental.

A Reutilização de Código é feita com o auxílio do Agent Factory, Chella, Cossentino e Sabatucci (2003), uma ferramenta adotada no PASSI convencional, que permite a criação de um sistema multi-agente, reutilizando as funcionalidades de um repositório de padrões.

A ferramenta fornece um suporte para a compilação automática de uma quantidade relevante de código (não apenas esqueletos de classe, mas também partes internas dos métodos especificados nos padrões reutilizados) e executa a engenharia reversa do código modificado manualmente.

De acordo com Chella et al. (2006), essa abordagem baseada em padrões melhora a qualidade do projeto, aumenta a quantidade de código produzido rapidamente e reduz o tempo e os custos gerais de desenvolvimento.

Para produzir uma documentação mínima da fase de Projeto, Chella et al. (2006) criaram um adicionador (add in) para a ferramenta MetaEdit+ [8].

Esta ferramenta transforma as informações armazenadas no Diagrama de Identificação do Agente e nos modelos estruturais e comportamentais, gerados pela ferramenta Agent Factory, em quatro tipos de documentos: Code (COD), Multi-Agent System Definition (MASD), Multi-Agent Behavior Descript (MABD), Single-Agent System Definition (SASD):

- COD tem um diagrama de classes representando agentes, suas comunicações e parâmetros relacionados (linguagem do conteúdo, protocolo de interação do agente e

[8] De acordo com a empresa MetaCase https://metacase.com, a ferramenta MetaEdit+ permite que sejam melhorados radicalmente a produtividade e a qualidade do desenvolvimento, gerando código completo diretamente a partir de modelos.

ontologia referida);

- MASD tem um diagrama de classes em que representamos todo o sistema no nível de abstração social e multiagente. O diagrama representa cada agente com uma classe e as tarefas do agente como métodos da classe;

- MABD tem um diagrama de atividades que representa o fluxo de controle e as comunicações entre todos os agentes;

- SASD tem um diagrama de classe diferente para cada agente, a fim de representar sua estrutura interna e todas as suas tarefas da maneira mais detalhada.

Na etapa de Codificação os programadores completam manualmente o código produzido anteriormente, pondo em prática todas as regras da programação extrema.

A Fase de Testes envolve a fase de codificação, é dividida em duas partes (Plano de Testes e Testes), onde o Plano de Testes ocorre antes da fase de Codificação e os Testes após.

De acordo com as regras da XP, a fase de testes começa antes da codificação e deve ser contínua. O projetista e os programadores são responsáveis por planejar um ou mais testes que o agente deve satisfazer quando codificado.

Após a conclusão do código, durante a fase de teste, o teste real é executado de acordo com os planos de teste definidos anteriormente. Essa atividade representa uma forma de controlar o trabalho dos programadores, pois se pelo menos um teste falhar, o componente estará sujeito a um refinamento e uma refatoração[9] até que todos os testes sejam satisfeitos.

Quando a fase de testes terminar com êxito, uma versão de trabalho do agente é liberada. Esta pode não ser uma versão final, mas é considerada, podendo ser usada como um protótipo para uma demonstração ao cliente.

1.2.3 Metodologia Ingenias-Scrum

De acordo com González-Moreno et al. (2014) Ingenias-Scrum é baseado no processo de desenvolvimento: Scrum. O processo adota o plano iterativo e rápido apresentado originalmente pela metodologia e usa algumas das atividades e a maioria dos produtos de trabalho da proposta da INGENIAS com o *Unified Development Process* (UDP).

A nova abordagem também é baseada no metamodelo INGENIAS, mas é mais focada no desenvolvimento de código do que na especificação do sistema. Aproveita o INGENIAS Agent *framework* (IAF), que faz parte do *INGENIAS Development Kit*

[9] Refatoração é o processo de modificar um sistema de software para melhorar a estrutura interna do código sem alterar seu comportamento externo.

(IDK).

Como esta abordagem usa os mesmos metamodelos que os baseados em UDP, não há grandes diferenças nos modelos do Ingenias, porém a organização dos produtos de trabalho e o tempo gasto para obter os resultados finais são bastante diferentes.+

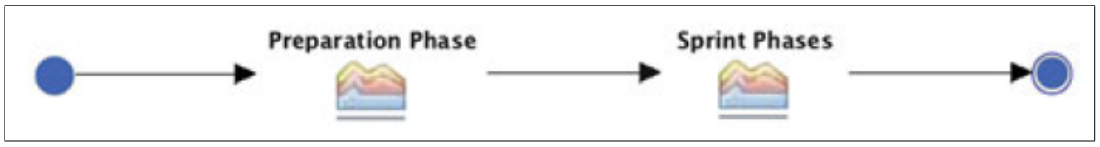
É uma metodologia da engenharia de software orientada a agente - *Agent-OrientedSoftware Engineering* (AOSE) de propósito geral, que adota um desenvolvimento dirigido por modelo - *Model Driven Development* (MDD).

O desenvolvimento é organizado em torno da definição de modelos que são semi-automaticamente transformados em diferentes produtos, por exemplo: código, teste e documentação.

Segundo González-Moreno et al. (2014) a estrutura do processo *Scrum* é especialmente adequada para desenvolvimentos de engenharia de conhecimento com base no uso de MAS.

O processo Ingenias *Scrum* é composto pelas fases de Preparação e *Sprints*, conforme a Figura 21. De acordo com o *framework Scrum* original, a fase de Preparação compreende as seguintes atividades: iniciar o *backlog* do produto, planejar a liberação e tarefas de preparação.

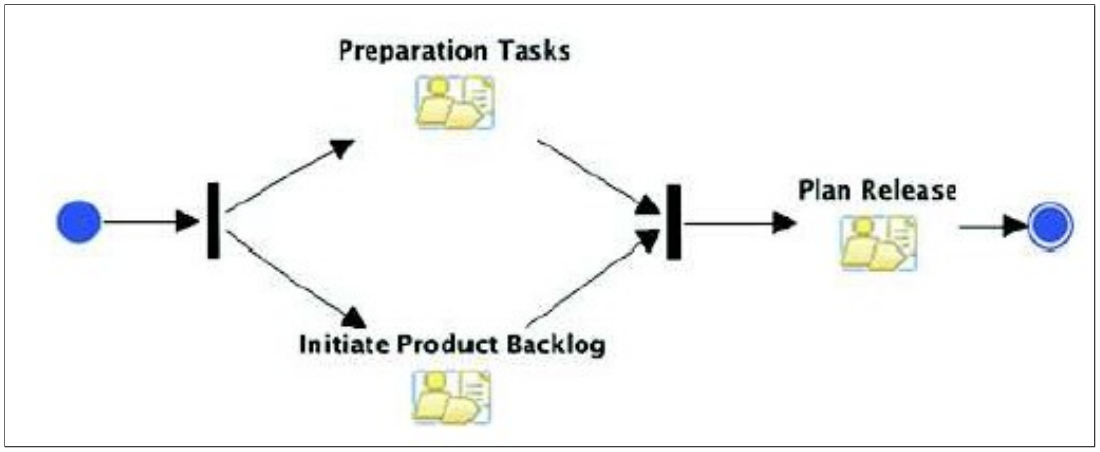
Figura 21 - Processo Ingenias Scrum



Fonte: González-Moreno et al. (2014).

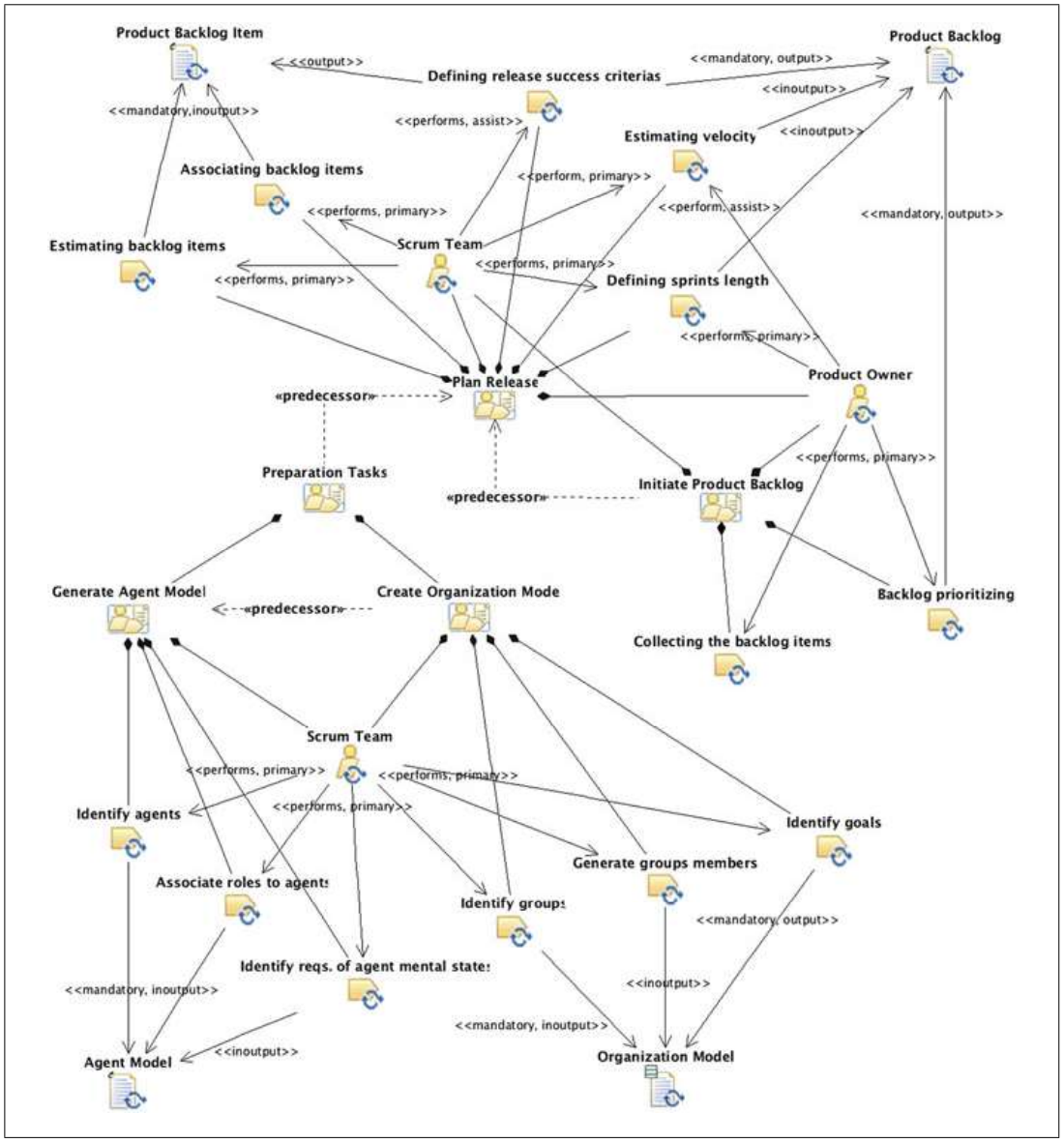
A Figura 22 mostra o fluxo de trabalho da fase de preparação, e a Figura 23 mostra a estrutura da fase em termos das tarefas e produtos de trabalho a serem realizados.

Figura 22 - Atividades da fase de Preparação



Fonte: González-Moreno et al. (2014).

Figura 23 - Fase de preparação descrita em termos de atividades e produtos de trabalho



Fonte: González-Moreno et al. (2014).

As fases de *Sprint* cobrem a execução de todas as *sprints* necessárias para liberar o produto final.

O *backlog* do produto é uma lista priorizada de recursos que contém descrições curtas de todas as funcionalidades desejadas para o produto.

Para Ingenias, estes requisitos são descritos utilizando o IDK, seja adaptando uma especificação de um projeto anterior ou definindo um novo.

Ao usar Scrum, não é necessário iniciar um projeto com um longo, esforço inicial para documentar todos os requisitos.

Normalmente, um Time *Scrum* e seu *Product Owner* começam escrevendo tudo o que eles podem pensar facilmente, e isso constitui a primeira versão do *backlog* do produto.

Isso quase sempre é mais do que suficiente para um primeiro *sprint*(quando documentado de acordo com o IDK.

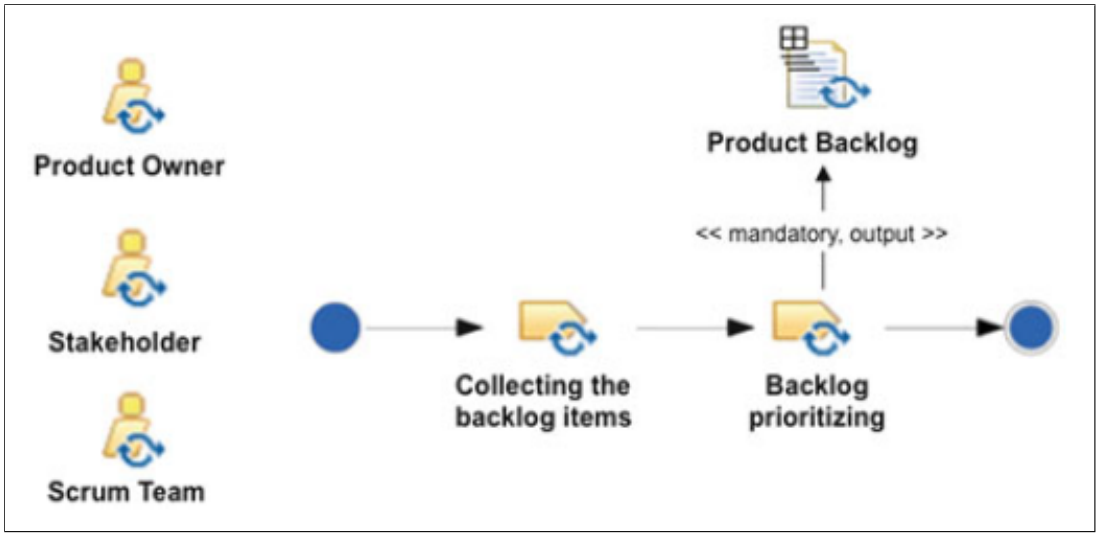
O *backlog* do produto na especificação INGENIAS, pode crescer e mudar conforme mais informações são aprendidas sobre o produto e os clientes. O fluxo de tarefas necessárias para realizar esta atividade é descrito na Figura 24 e detalhado no Quadro 7.

A atividade Preparação das Tarefas é realizada pelo time *Scrum* para preparar as próximas reuniões com o Dono do Produto e as Partes Interessadas.

O time cria uma especificação INGENIAS em andamento que visa reorganizar, formalizar e estruturar as especificações preliminares no *backlog* do produto. Isso implica que essa atividade é mais complexa que a do *Scrum* original.

Requer a execução de várias tarefas comuns às atividades do INGENIAS. Cria o Modelo da Organização e o Modelo do Agente.

Figura 24 - Fluxo de trabalho da atividade de Inicialização do *Backlog* do Produto



Fonte: González-Moreno et al. (2014).

Quadro 7 - Descrição das tarefas da atividade de inicialização do *backlog* do produto

Activity	Task	Description	Involved roles
Initiate *Product Backlog*	Collecting the backlog items	The *Product Owner* establishes a first list of requirements (a requirement becomes a *Product Backlog Item*) that she/he wish to be implemented for the end of the release. After this, a workshop is organized in which the Product Owner participates, as well as, the whole team and the stakeholders (if necessary). The Product Owner introduces the backlog draft and the assistants may suggest additional items.	Product Owner Scrum Team Stakeholder
	Backlog prioritizing	The full list of backlog items is prioritized by the Product Owner. Items with higher priority must be addressed first in the release. If a backlog contains a lot of items, the definition of *Themes*, and the association of one of them to each item, is suggested. After this, the Product Owner prioritizes the *themes*.	Product Owner Scrum Team Stakeholder

Fonte: González-Moreno et al. (2014).

Na liberação do Plano de Release as atividades anteriores entregam uma especificação inicial do INGENIAS, conforme o Quadro 8. Com essas informações, o *Scrum* Master, em conjunto com o Dono do produto, pode estabelecer a liberação do plano, que define as *sprints* necessárias para atender às metas.

Várias tarefas devem ser executadas, incluindo a definição dos critérios de sucesso da liberação, a estimativa dos itens da lista de pendências e a velocidade da*sprint*, a definição do comprimento das *sprints* e a associação dos itens da lista de pendências.

O Backlog do Produto é um conjunto de itens, chamado *Product Backlog Items* (PBI). Um PBI na abordagem INGENIAS-Scrum é um produto de trabalho de uma composição de várias partes (Estruturado + Estrutural + Comportamental) [10].

Na proposta original, cada PBI pode ser descrito como texto livre e refere-se a um dos seguintes tipos de itens:

- Features: uma característica geralmente é descrita com as histórias do usuário que compreendem uma descrição curta e simples da funcionalidade desejada da perspectiva do usuário. Um exemplo: "Como autor, posso enviar um artigo para a conferência e serei informado sobre sua recepção e o número de identificação atri-

[10] A análise estruturada é uma atividade de construção de modelos. Utiliza uma notação que é própria ao método de análise estruturada com a finalidade de retratar o fluxo e o conteúdo das informações utilizadas pelo sistema, dividir o sistema em partições ambientais e comportamentais e descrever a essência daquilo que será construído. A UML é uma linguagem que realiza a modelagem de estruturas que irão compor uma aplicação. Ela divide seus diagramas em estruturais e comportamentais. São exemplos de estruturais os diagramas: de classe, de objeto, de componentes e de pacotes. E os comportamentais descrevem as ações que o sistema deve realizar para responder da melhor forma aos eventos definidos no modelo ambiental. São exemplos: diagramas de caso de uso, de máquina de estados, de atividades e de interação.

Quadro 8 - Descrição das tarefas da atividade plano de release

Activity	Task	Description	Involved roles
Plan Release	Defining *release success criteria*	This task pursues to establish the criteria to consider a Release successfully finished by the Product Owner. The Scrum Team (the Scrum Master, in particular) must take into account that there are 2 types of releases: *End-Date Driven Release*, which must be available to the end users before a deadline; and *Feature Driven Release*, in which the release finishes when all of the requirements are implemented.	Product Owner Scrum Team
	Estimating *backlog items*	Estimation must be performed by the team item per item. It is a good idea to start by the highest priority items. In this activity, it is usual to use as range of values for estimations the *Fibonacci Numbers*: 1, 2, 3, 5, 8, and 13. Regarding the *estimation techniques*, the collective ones are preferred.	Scrum Team
	Defining *Sprint length*	Although historically the length of a sprint is 30 days, INGENIAS-Scrum recommends 15 days, although a week or 21 days are also acceptable lengths, according to the difficulty of the work and the human resources available.	Scrum Team
	Estimating *velocity*	The *velocity* (i.e., the number of items that could be finished) is calculated as the sum of all items checked and approved as fully implemented in a Sprint. This activity makes an estimation of the expected team velocity. It should be done automatically because it is supposed that the team has worked together in previous projects using the INGENIAS tools and the JADE platform. Nevertheless, the velocity can be manually estimated for the first time and adjusted in the next sprints.	Scrum Team
	Associating *backlog items* to *sprints*	This task takes into consideration the parameters obtained from the previous tasks. A slight adjustment of items prioritization may be done at this stage, so the velocity can be better adjusted to the team features. It is not required to make this association for all sprints of the release at the beginning, but it should be done for the 2 or 3 first ones.	Product Owner Scrum Team

Fonte: González-Moreno et al. (2014).

buído a ele".

- Bugs: apesar de se tratar de um erro de na aplicação, em geral, um bug não apresenta nenhuma diferença real de uma nova *feature* e será tratado como se fosse uma mudança de funcionalidade, o modo de escrever será o mesmo.

- Trabalho técnico: esse tipo de item apresenta aspectos detalhados relacionados à implantação do sistema. Um exemplo de trabalho técnico seria: "Use uma versão até 2.0 do sistema de gerenciamento de e-mail".

- Aquisição de conhecimento: Este item está relacionado à obtenção das habilidades e conhecimentos necessários para o projeto. O item reflete algum tipo de conhecimento externo necessário para entender melhor o problema ou aplicar uma solução específica.

Um exemplo pode ser pedir aos membros da equipe o estudo e/ou a seleção entre várias bibliotecas Java para determinar a mais precisa para o sistema ou encontrar uma solução para o problema restrito ao uso dessa biblioteca.

As histórias de usuários geralmente determinam vários objetivos que devem ser satisfeitos pelos agentes que desempenham certas funções.

Os erros devem ser documentados, modificando a descrição da história do usuário relacionada ou adicionando uma nova.

Em relação aos trabalhos técnicos e itens de aquisição de conhecimento, uma boa prática é documentá-los, definindo um teste ou prova para uma parte do sistema ou especificando a API concreta necessária para resolver o problema em conjunto com o agente encarregado de aplicar tal solução.

Em geral, um PBI deve incluir os seguintes atributos gerais: (i) Valor estimado, frequentemente calculado em relação a outros itens usando prioridades; (ii) Custo estimado de desenvolvimento; (iii) Eventualmente, o tema (ou seja, domínio funcional) ao qual ele pertence; (iv) Eventualmente, seu tipo, que pode ser definido como: objetivo do sistema do projeto; bug resultante do processo; ou requisito não funcional; (v) Os critérios para aceitação do teste.

Por meio dessa fase do processo de desenvolvimento, um PBI pode alcançar os seguintes estados: criado, estimado, planejado (associado a uma sprint futura), associado ao sprint atual (a implementação está em andamento) e concluído.

No final da fase de Preparação, todo o PBI identificado em conjunto com seu estado deve ter pelo menos um objetivo/meta associado.

As Fases de Sprint

Na abordagem INGENIAS-Scrum, é sugerido menos de 21 dias para um sprint, porque o comprimento de sprint mais comum para a estrutura Scrum é de 2 semanas e o uso das ferramentas acionadas pelo modelo INGENIAS promove ciclos curtos.

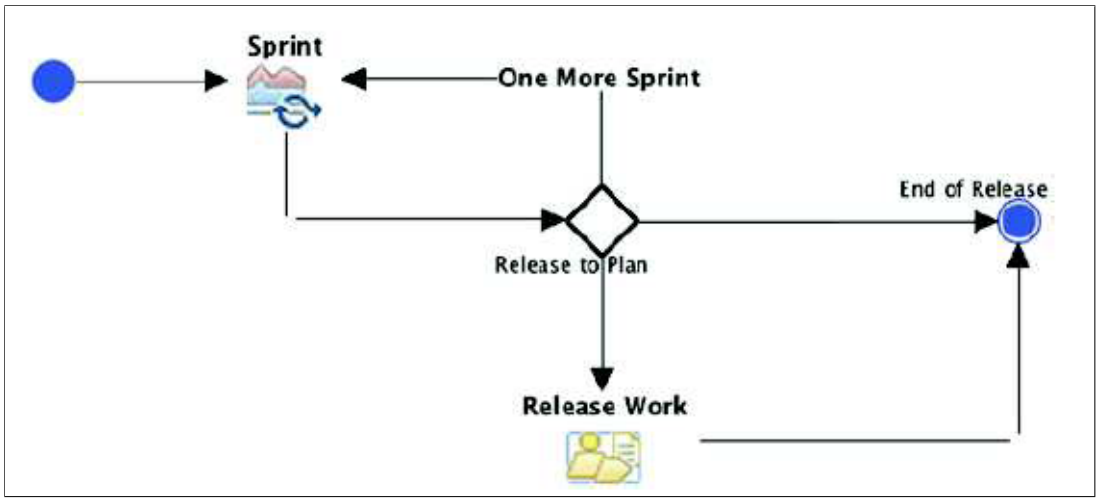
O objetivo usual de cada sprint é implementar e testar o código, que abrange pelo menos um dos objetivos. A Figura 25 e a Figura 26 mostram os fluxos das atividades a serem realizadas durante a fases de Sprint.

A metodologia Ingenias-Scrum é composta pelos papéis: *Product Owner*, Time *Master*, Time Scrum e *Stakeholder*, conforme o Quadro 9.

Nos trabalhos diários a equipe realiza uma lista de pendências para atingir as metas da sprint. A recomendação do processo Ingenias-Scrum para esta atividade é mesclar e distribuir, de acordo com as necessidades, as tarefas apresentadas: (i) Refinar modelos organizacionais com relações sociais, que incluem o refinamento dos modelos Organização, Agente e Ambiente; (ii) Criar o modelo de tarefas e objetivos; (iii) Mostrar execução de tarefas usando o modelo de interação; (iv) Criar um modelo de componente; (v) Criar um modelo de implantação; (vi) Especificar modelos de código a serem aplicados; (vii) Validar o código.

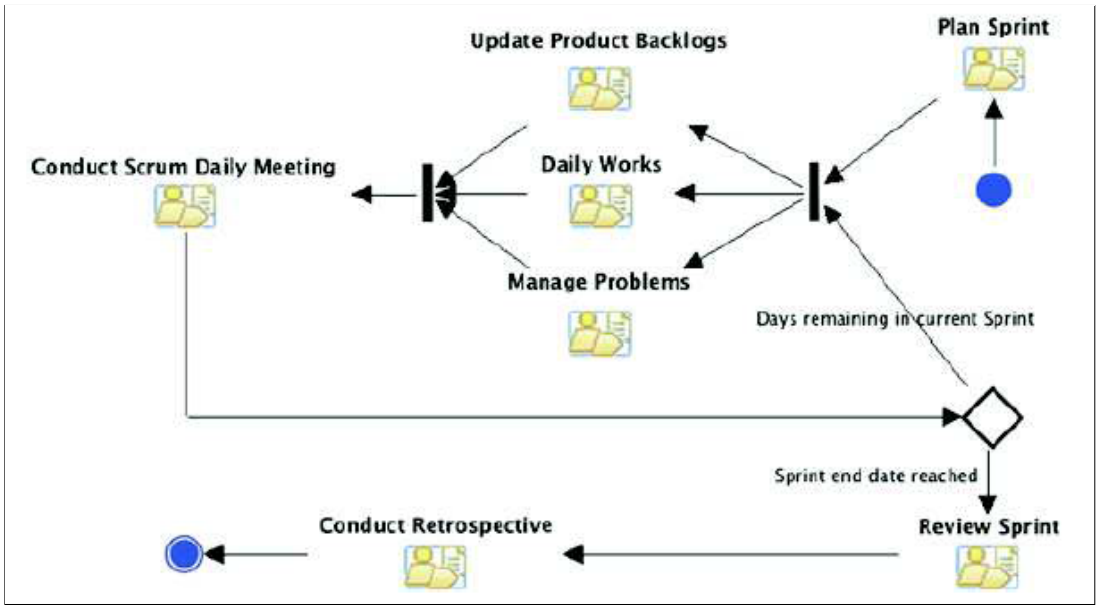
Exemplos do modelo organizacional, modelo de metas, modelo de tarefas e o modelo de agentes estão na sequência (Figura 27, Figura 28, Figura 29 e Figura 30).

Figura 25 - O fluxo de atividades das Fases de Sprint



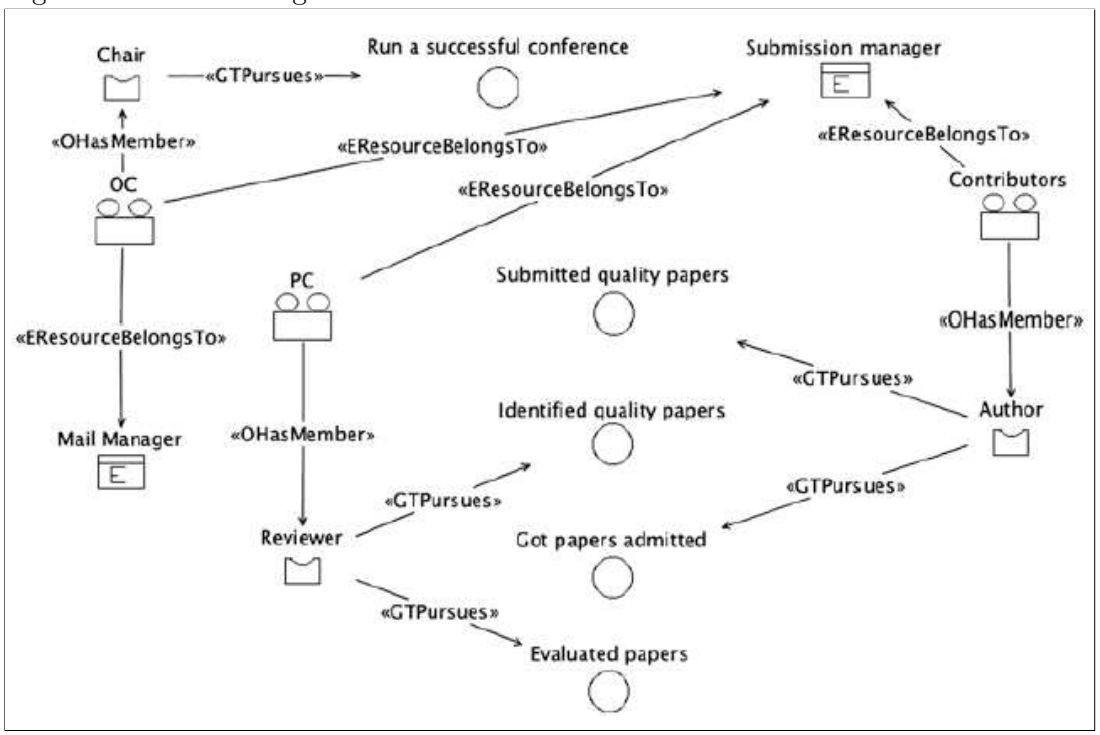
Fonte: González-Moreno et al. (2014).

Figura 26 - O fluxo de trabalho da atividade Sprint



Fonte: González-Moreno et al. (2014).

Figura 27 - Modelo Organizacional



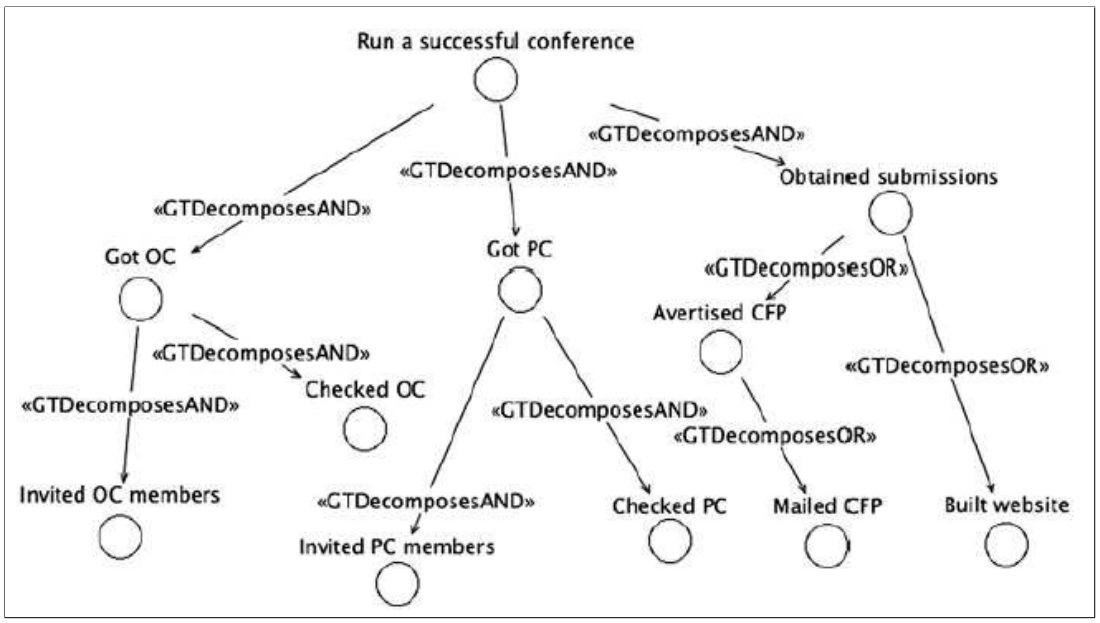
Fonte: González-Moreno et al. (2014).

Quadro 9 - Papéis do Ingenias-Scrum

Papéis do Ingenias-Scrum	Descrição
Dono do produto (Product Owner)	Durante cada fase da Sprint, o Product Owner é responsável por analisar as mudanças sugeridas para adicioná-las ao backlog e priorizar os itens adicionados. Isso é feito antes do próximo Sprint Planning. Idealmente, este trabalho é feito 1 ou 2 dias antes da Revisão da Sprint, durante uma sessão de brainstorming com o Time Scrum.
Time Master	As fases iterativas da Sprint executam o desenvolvimento do produto. O Time Master é o responsável por ajudar o Time Scrum a trabalhar de forma autônoma e constantemente se aprimorando, fazendo os seguintes tipos de tarefas: • Tarefas periódicas, cujo objetivo principal é organizar e promover o trabalho colaborativo durante as reuniões: Daily Scrum, Sprint planning, Sprint review e Retrospectiva; • Tarefas de eventos, cujo objetivo é remover impedimentos. Esta tarefa baseia-se principalmente em levar em consideração eventos anteriores para resolver problemas recorrentes o mais rápido possível, protegendo a equipe de distrações externas; • Tarefa em segundo plano, na qual o Master Team tenta garantir que a equipe permaneça produtiva e focada no objetivo do projeto: desenvolver itens de backlog em estreita colaboração com o Product Owner.
Time Scrum	Na abordagem INGENIAS, a equipe deve estar entre 3 e 5 pessoas. O Time Scrum é o principal responsável pelo Incremento do Produto e sua participação é fundamental nas atividades: Planejar Sprint, Atualizar o Backlog do Produto, Trabalhos Diários e Conduzir Reuniões Diárias Scrum. Além disso, a participação do Time Scrum é importante na atividade Review Sprint e Conduct Retrospective.
Stakeholder	Como na Fase de Preparação, não é necessária a participação dos Stakeholders em nenhuma das Fases do Sprint. Eles podem participar desempenhando uma função secundária nas atividades Atualizar Backlog do Produto, Conduzir Retrospectiva e Revisar Sprint. Sua participação é consultiva, eles fazem sugestões sobre os objetivos que o sistema deve satisfazer e as soluções certas do ponto de vista do cliente.

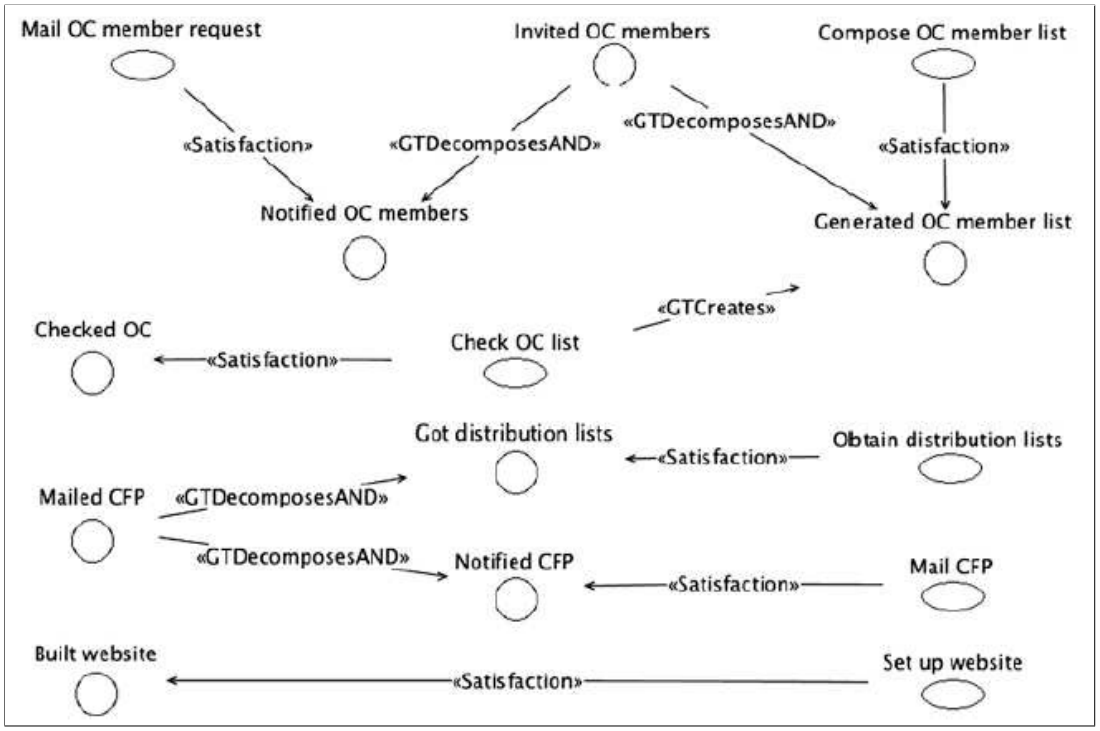
Fonte: Sommerville (2019).

Figura 28 - Modelo de metas selecionada em diferentes submetas



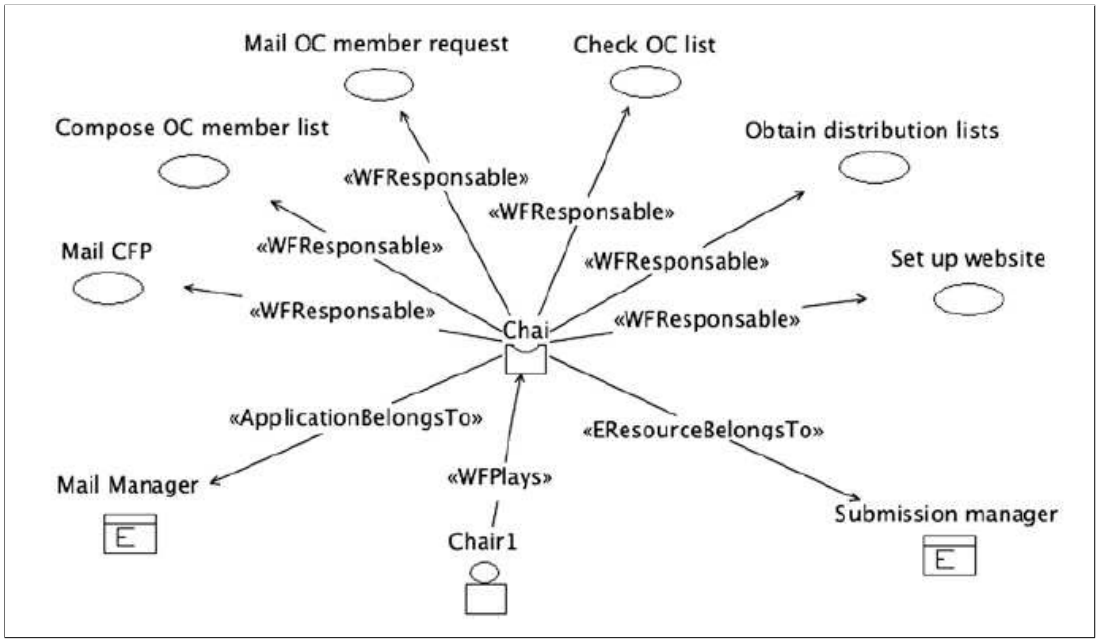
Fonte: González-Moreno et al. (2014).

Figura 29 - Modelo de tarefas para atingir os objetivos



Fonte: González-Moreno et al. (2014).

Figura 30 - Modelo de agentes



Fonte: González-Moreno et al. (2014).

1.3 **Estudo de Caso na Engenharia de Software**

De acordo com Runeson et al. (2012) um conceito de estudo de caso em Engenharia de Software é:

> uma investigação empírica que recorre a várias fontes de evidência para investigar uma instância (ou um pequeno número de instâncias) de um fenômeno contemporâneo de engenharia de software dentro de seu contexto da vida real, especialmente quando a fronteira entre fenômeno e contexto não pode ser claramente especificado.

Os estudos de caso são utilizados para pesquisar projetos, atividades ou atribuições. Os dados são coletados para uma finalidade específica ao longo do estudo. Com base na coleta de dados, análises estatísticas podem ser realizadas. O estudo de caso normalmente visa rastrear um atributo específico ou estabelecer relacionamentos entre diferentes atributos e o nível de controle é mais baixo por ser considerado um estudo observacional. De acordo Wohlin et al. (2012), a utilização de Estudo de Caso na Engenharia de Software, não deve ser utilizado apenas para avaliar, como e porque certos fenômenos acontecem, mas também para a avaliar a diferença entre dois métodos de design.

Avaliar qual método é o mais adequado em uma determinada situação. Para garantir a validade do estudo de caso é necessário criar uma base sólida para avaliar o resultado. Existem três maneiras de organizar o estudo:

1. Uma comparação dos resultados do uso do novo método, em relação a uma linha de base da empresa é uma solução. A empresa deve coletar dados de projetos padrão e calcular características como produtividade média e taxa de defeitos. Então é possível comparar os resultados do estudo de caso com os números da linha de base;

2. O projeto em estudo utiliza o novo método e o projeto irmão o atual. Ambos os projetos devem ter as mesmas características, ou seja, os projetos devem ser comparáveis;

3. Se o método se aplicar a componentes individuais do produto, ele poderá ser aplicado aleatoriamente a alguns componentes e não a outros. Isso é muito semelhante a um experimento, mas como os projetos não são sorteados aleatoriamente da população de todos os projetos, não é um experimento.

Ao realizar o Estudo de Caso, é necessário minimizar os efeitos dos fatores de confusão. Um fator de confusão, é um fator que torna impossível distinguir os efeitos entre dois fatores.

Por exemplo, pode ser difícil dizer se um resultado melhor depende da ferramenta ou da experiência do usuário na ferramenta. O fator de confusão pode envolver problemas com o aprendizado de como usar uma ferramenta ou método ao tentar avaliar seus benefícios, ou usar uma equipe muito entusiasmada ou cética.

Existem prós e contras nos Estudos de Caso, que são muito valiosos porque incorporam qualidades que um experimento não consegue visualizar, por exemplo escala, complexidade, imprevisibilidade e dinamismo. Um estudo de caso pequeno ou simplificado raramente é um bom instrumento para descobrir princípios e técnicas de engenharia de software.

Os pesquisadores não estão no controle total de uma situação de estudo de caso. O problema é que não podemos ter certeza sobre os efeitos devidos aos fatores de confusão. O bom de não ter o controle total da situação é que as mudanças imprevisíveis frequentemente dizem muito sobre os problemas que estão sendo estudados.

Ao conduzir um estudo de caso, existem cinco etapas importantes do processo a serem seguidas: (i) Design do estudo de caso; (ii) Preparação para coleta de dados; (iii) Coleta de dados; (iv) Análise dos dados coletados; (v) Relatórios.

Na etapa 1 - Design do estudo de caso, os objetivos são definidos e o estudo de caso é planejado. O plano para o estudo de caso deve conter pelo menos os seguintes elementos: (i) Objetivo: o que alcançar? (ii) O caso: o que é estudado? (iii) Teoria: quadro de referência (iv) Perguntas de pesquisa: o que saber? (v) Métodos: como coletar dados? (vi) Estratégia de seleção: onde buscar dados?

O objetivo do estudo pode ser: exploratório, descritivo, explicativo ou aprimoramento. O objetivo é naturalmente formulado de maneira mais geral e menos preciso do que em projetos de pesquisa fixos. É inicialmente mais como um ponto de foco que evolui durante o estudo.

As questões de pesquisa estabelecem o que é necessário saber para cumprir o objetivo do estudo. Similar ao objetivo, as questões de pesquisa são desenvolvidas durante o estudo e são restritas a questões de pesquisa específicas durante as iterações do estudo.

Na engenharia de software, o caso pode ser um projeto de desenvolvimento de software, que é a escolha mais simples. Alternativamente, pode ser um indivíduo, um grupo de pessoas, um processo, um produto, uma política, um papel na organização, um evento, uma tecnologia, etc.

O projeto, o indivíduo, o grupo etc. também podem constituir uma unidade de análise dentro de um caso. Os estudos sobre "programas de brinquedos" ou similarmente são excluídos devido à falta de contexto da vida real.

O uso de teorias para desenvolver a direção da pesquisa não está bem estabelecido no campo da engenharia de software, no entanto, a definição do quadro de referência do estudo torna o contexto da pesquisa do estudo de caso claro e ajuda tanto os que realizam a pesquisa quanto os que revisam os resultados dela.

Na falta de teoria, o quadro de referência pode ser alternativamente expresso em termos de ponto de vista da pesquisa e do histórico dos pesquisadores.

Os estudos de caso da teoria fundamentada não têm, naturalmente, uma teoria específica.

As principais disposições sobre métodos de coleta de dados são definidas no tempo de projeto para o estudo de caso, embora decisões detalhadas sobre procedimentos de coleta de dados sejam tomadas posteriormente.

Lethbridge, Sim e Singer (2005) definem três categorias de métodos: direta (por exemplo, entrevistas), indireta (por exemplo, instrumentação de ferramenta) e independente (por exemplo, análise de documentação).

A seleção de casos é particularmente importante ao replicar estudos de caso. Um estudo de caso pode ser literalmente replicado, ou seja, o caso é selecionado para prever resultados semelhantes, ou é teoricamente replicado, ou seja, o caso é selecionado para prever resultados contrastantes por razões previsíveis.

Na etapa 2 - Preparação de Coleta de Dados, são definidos procedimentos e protocolos para coleta de dados.

O protocolo é um documento continuamente alterado que é atualizado quando os planos para o estudo de caso são alterados e serve a vários propósitos:

- Serve como um guia na condução da coleta de dados, evitando assim que o pesquisador deixe de coletar os dados que estavam planejados para serem coletados.

- Os processos de formulação do protocolo tornam a pesquisa concreta na fase de planejamento, o que pode ajudar o pesquisador a decidir quais fontes de dados usar e quais perguntas fazer.

- Outros pesquisadores e pessoas relevantes podem revisá-lo para dar feedback sobre os planos. O feedback de outros pesquisadores sobre o protocolo pode, por exemplo, diminuir o risco de perder fontes de dados relevantes, perguntas da entrevista ou

papéis a serem incluídos na pesquisa e questionar a relação entre as perguntas da pesquisa e as perguntas da entrevista.

- Pode servir como um registro ou diário onde toda a coleta e análise de dados é registrada junto com as decisões de mudança com base na natureza flexível da pesquisa. Esta pode ser uma importante fonte de informação quando o estudo de caso for relatado posteriormente.

A fim de acompanhar as mudanças durante o projeto de pesquisa, o protocolo deve ser mantido sob alguma forma de controle de versão.

Na etapa 3 - Coleta de Dados, execução com coleta de dados no caso estudado. É importante usar várias fontes de dados em um estudo de caso para limitar os efeitos de uma interpretação de uma única fonte de dados.

Em um estudo de caso, também é importante levar em consideração pontos de vista de diferentes funções e investigar diferenças, por exemplo, entre diferentes projetos e produtos.

Normalmente, as conclusões são tiradas analisando as diferenças entre as fontes de dados.

De acordo com Lethbridge, Sim e Singer (2005), as técnicas de coleta de dados podem ser divididas em três níveis. O primeiro grau são métodos diretos significam que o pesquisador está em contato direto com os sujeitos e coleta dados em tempo real. Por exemplo: entrevistas, grupos de foco, pesquisas Delphi por Dalkey e Helmer (1963) e observações com "protocolos de pensar em voz alta" Owen, Brereton e Budgen (2006).

Métodos de primeiro grau são geralmente mais caros de aplicar do que métodos de segundo ou terceiro grau, uma vez que requerem um esforço significativo tanto do pesquisador quanto dos sujeitos.

O Segundo grau corresponde a métodos indiretos onde o pesquisador coleta diretamente os dados brutos, sem realmente interagir com os sujeitos durante a coleta de dados. Os exemplos são o registro do uso de ferramentas de engenharia de software e observações por meio de gravação de vídeo.

Uma vantagem dos métodos de primeiro e segundo grau é que o pesquisador pode, em grande medida, controlar exatamente quais dados são coletados, como são coletados, em qual a forma de coleta dos dados, qual o contexto etc.

O terceiro grau uma análise independente de artefatos de trabalho onde já estão disponíveis e, às vezes, são usados dados compilados.

Este é o caso, por exemplo, quando documentos como especificações de requisitos e relatórios de falhas de uma organização são analisados ou quando dados de bancos de dados organizacionais, como contabilidade de tempo, são analisados.

Os métodos de terceiro grau são, em sua maioria, mais baratos, mas não oferecem o mesmo controle ao pesquisador; portanto, a qualidade dos dados também não está sob

controle, nem em relação à qualidade dos dados originais, nem ao seu uso para fins de estudo de caso. Em muitos casos, o pesquisador deve, até certo ponto, basear os detalhes da coleta de dados em que os dados estão disponíveis.

Na etapa 4 – Análise dos Dados Coletados, pode ser análise de dados quantitativos ou qualitativos.

Para dados quantitativos, a análise geralmente inclui análise de estatísticas descritivas, análise de correlação, desenvolvimento de modelos preditivos e teste de hipóteses. Todas essas atividades são relevantes na pesquisa de estudo de caso.

Na análise qualitativa o objetivo básico é tirar conclusões dos dados, mantendo uma cadeia de evidências clara.

A cadeia de evidências significa que um leitor deve ser capaz de seguir a derivação de resultados e conclusões a partir dos dados coletados.Isso significa que devem ser apresentadas informações suficientes de cada etapa do estudo e de todas as decisões tomadas pelo pesquisador.

Na etapa 5 – Relatório, comunica as conclusões do estudo e é a principal fonte de informação para julgar a qualidade do estudo.

Os relatórios podem ter diferentes públicos, como pesquisadores pares, formuladores de políticas, patrocinadores de pesquisas e profissionais da indústria. Isso pode levar à necessidade de escrever relatórios diferentes para públicos diferentes.

A pesquisa de Estudo de Caso é amplamente discutida pelos seguintes pesquisadores: Robson (2002), Stake (1995) e Yin (2009), e especificamente para engenharia de software por Pfleeger (1994), Kitchenham, Pickard e Pfleeger (1995), Verner et al. (2009), Runeson et al. (2012), e Runeson e Skoglund (2009).

2 REVISÃO DA LITERATURA

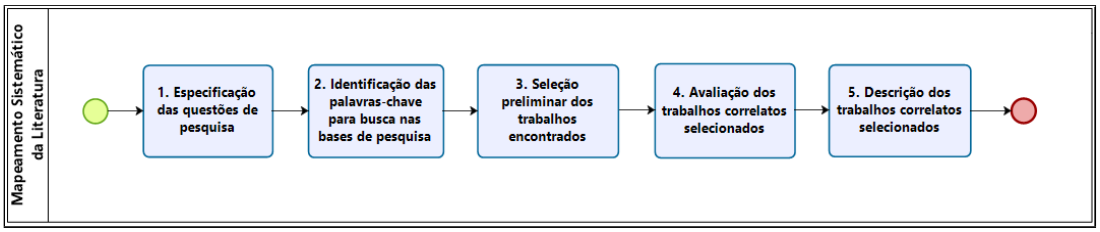
Este capítulo aborda o planejamento da Revisão da Literatura, apresentando o protocolo que foi produzido durante o processo para realização da revisão, e no final é apresentada a síntese dos principais trabalhos correlatos com esta dissertação.

2.1 Processo de Revisão da Literatura

A realização dessa revisão utilizou as fases de um mapeamento sistemático (KITCHENHAM, 2007): o planejamento da revisão, a condução da revisão e a apresentação dos resultados.

O planejamento do processo dessa revisão segue o fluxo de seis atividades, conforme descrito na Figura 31.

Figura 31 - Processo de Revisão da Literatura



Fonte: A autora.

2.1.1 Especificação das Questões de Pesquisa

O propósito desta revisão foi identificar quais trabalhos descrevem metodologias orientadas a agentes e desenvolvimento ágil, especialmente, aqueles que usam as metodologias O-MaSE, Ingenias-scrum e *Agile* PASSI.

Seria importante identificar principalmente, algum trabalho que tivesse utilizado a metodologia O-MaSE de forma ágil, já que tínhamos o objetivo de propor uma adaptação ágil nessa metodologia, para ser utilizada no estudo de caso junto as metodologias Ingenias-Scrum e *Agile* PASSI. As metodologias Ingenias-scrum e *Agile* PASSI foram pesquisadas para identificar se havia uma proposta mais recente.

Partindo deste propósito foi utilizada a seguinte questão de pesquisa:

Quais trabalhos publicados a partir de 2014 contêm metodologias orientadas a agentes O-MaSE, ou ingenias-scrum, ou *Agile* PASSI com desenvolvimento ágil?

2.1.2 Identificação das Palavras-chave para Buscas nas Bases de Pesquisa

A segunda atividade do processo de revisão foi a Identificação das Palavras-chave para Buscas nas Bases de Pesquisa. As palavras-chave são extraídas das questões de pesquisa. O conjunto de palavras-chave pode conter termos escritos de maneiras diferentes, para obter trabalhos que possam estar com termos variados sobre o mesmo assunto.

- Palavras-chave 1: ("multi-agent"OR "multiagent"OR "multi agent"OR "agent oriented"OR "agent-oriented") AND ("Agility"OR "Agile") AND ("Methodology"OR "Method") AND ("O-MaSE"OR "OMASE"OR "INGENIAS SCRUM"OR "Ingenias Scrum"OR "Ingenias-Scrum"OR "Agile Passi");
- Palavras-chave 2: (“multi-agent” OR “multiagent” OR “multi agent” OR “agent oriented” OR “agent-oriented”) AND ("Agility"OR "Agile") AND ("Methodology"OR "Method");

O conjunto de palavras-chave foram buscadas em Bases de Dados Científicas (IEEE e ACM) e no Indexador de Artigos (Google Scholar) descritos no Quadro 10.

Quadro 10 - Bases de Dados

Bases de Dados	Endereços Eletrônicos
Google Scholar	http://scholar.google.com/
IEEE Xplore Digital Library	http://ieeexplore.ieee.org/
ACM Digital Library	http://dl.acm.org/

Fonte: A autora.

A pesquisa no Google Acadêmico, trouxe uma grande quantidade de informações que não condiziam com o estudo, sendo necessário restringir o filtro delimitando pelos métodos trabalhados neste trabalho, que corresponde ao conjunto de palavras chaves 1, porém continuou retornando alguns resultados que não condiziam com a pesquisa.

Nas bases da IEEE e da ACM foram aplicadas as palavras-chave 2, uma vez que as palavras-chave 1 não geraram resultado significativo.

A busca foi realizada no dia 15/11/19 e os resultados obtidos estão disponíveis na Tabela 1.

Tabela 1 - Resultado das buscas na diferentes bases de dados

Bases de Dados	Resultados das Buscas
Google Scholar	65
IEEE Xplore Digital Library	15
ACM Digital Library	6
Total	**86**

Fonte: A autora.

2.1.3 Seleção Preliminar dos Trabalhos Encontrados

A terceira atividade da revisão foi a seleção preliminar dos trabalhos encontrados. A partir do resultado das buscas das bases de dados, Tabela 1, são aplicados critérios de inclusão e exclusão a fim de selecionar os trabalhos que tenham mais interseção com a dissertação. Os Critérios de Inclusão são:

- Publicações realizadas desde 2014;
- O artigo deve estar disponível para acesso em arquivo .pdf;
- O artigo deve estar escrito em português ou inglês.

Os Critérios de Exclusão são:

- Retirada dos artigos duplicados;
- A existência de um artigo mais recente, ou mais completo (tratando do mesmo trabalho), elimina o artigo anterior;
- O trabalho não pode ser uma tese, dissertação ou monografia.

O resultado da terceira fase está disponível na Tabela 2.

Tabela 2 - Resultado da Seleção preliminar dos trabalhos encontrados

Bases de Dados	Resultados das Buscas
Google Scholar	39
IEEE Xplore Digital Library	14
ACM Digital Library	6
Total	**59**

Fonte: A autora.

2.1.4 Avaliação dos Trabalhos Correlatos Selecionados

A quinta atividade foi a avaliação dos trabalhos correlatos selecionados, para finalizar a análise dos trabalhos. Após a realização da atividade anterior, os seguintes critérios foram considerados: (i) O artigo não deve tratar somente de sistemas multiagentes sem abordar metodologias ágeis também; (ii) O artigo não deve tratar somente de metodologias ágeis; (iii) O artigo deve tratar de sistema multiagentes e desenvolvimento ágil; (iv) O trabalho deve apresentar alguma proposta ágil com orientação a agentes, independentemente do nível de complexidade.

A quantidade dos trabalhos resultantes desta fase é apresentada na Tabela 3, e a síntese dos 11 trabalhos estão descritos no próximo subcapítulo.

Tabela 3 - Resultado da Avaliação dos Trabalhos Correlatos Selecionados

Bases de Dados	Resultados das Buscas
Google Scholar	5
IEEE Xplore Digital Library	4
ACM Digital Library	2
Total	**11**

Fonte: A autora.

2.1.5 Descrição dos Trabalhos Correlatos Selecionados

Na quinta e última etapa do processo de mapeamento, os principais trabalhos correlatos são descritos considerando cada base de pesquisa. O Quadro 11, Quadro 12 e Quadro 13 apresentam as sínteses desses trabalhos.

Quadro 11 - Trabalhos encontrados na base Google Scholar

Base Google Scholar		
Título	Autor(es)	Link
Engineering Multi-Agent Systems Anno 2025.	Mascardi e Weyns (2019)	https://link.springer.com/chapter/10.1007/978-3-030-25693-7_1
Modelando Requisitos de um Jogo Educacional Médico usando a Metodologia INGENIAS SCRUM.	Novo et al. (2018)	http://wer.inf.puc-rio.br/∼wer/WERpapers/artigos/artigos_WER18/WER_2018_paper_22.pdf
PosoMAS: an extensible, modular SE process for open self-organising systems.	Steghofer et al. (2014)	https://link.springer.com/chapter/10.1007/978-3-319-13191-7_1
Developing an educational medical game using Agile-PASSI multi-agent methodology.	Ferreira et al. (2015)	https://ieeexplore.ieee.org/abstract/document/7167504
Ingenias-scrum,Handbook on Agent-Oriented Design Processes.	González-Moreno et al. (2014)	https://link.springer.com/book/10.1007%2F978-3-642-39975-6

Fonte: A autora.

Quadro 12 - Trabalhos encontrados na base IEEE Xplore

IEEE Xplore		
Título	Autor(es)	Link
Enhancing Requirements Engineering in Agile Methodologies by Agent-Oriented Goal Models: Two Empirical Case Studies.	Tenso et al. (2017)	https://ieeexplore.ieee.org/document/8054864
The Evaluation of Agile Demand Response: An Applied Methodology.	Babar et al. (2017)	https://ieeexplore.ieee.org/document/7926413
Simulation of pair programming using multi-agent and MBTI personality model.	Noori e Kazemifard (2015)	https://ieeexplore.ieee.org/document/7426665
Towards Agent-based Agile approach for Game Development Methodology.	Al-azawi, Ayesh e Obaidy (2014)	https://ieeexplore.ieee.org/document/6916626

Fonte: A autora.

Quadro 13 - Trabalhos encontrados na base ACM Digital Library

ACM Digital Library		
Título	Autor(es)	Link
Teamworking Strategies of Scrum Team: A Multi-Agent Based Simulation.	Wang (2018)	https://dl.acm.org/citation.cfm?id=3297179
A Multi-agent Game for Studying Human Decision-making.	Yu et al. (2014)	https://dl.acm.org/citation.cfm?id=2616113

Fonte: A autora.

O trabalho apresentado por Mascardi e Weyns (2019) traz uma reflexão sobre a função e o potencial da engenharia de MAS em uma seleção das principais particularidades, que caracterizam a prática moderna da engenharia de software: desenvolvimento ágil, computação na nuvem, blockchain distribuídos, sistemas ciberfísicos, Internet das Coisas e computação verde. Também, destaca as oportunidades para os desafios, as particularidades identificadas e conclui com uma série de questões éticas que o MAS e os engenheiros de sistemas modernos enfrentarão nos próximos anos.

O trabalho apresentado por Novo et al. (2018) apresenta os resultados da aplicação do método ágil orientado a agente, o INGENIAS Scrum, no desenvolvimento de um jogo educativo na área da saúde. Esse trabalho é usado para realizar uma das comparações do estudo de caso desta dissertação.

Steghofer et al. (2014) apresentam o método PosoMAS, o processo para sistemas multiagentes abertos e auto-organizados, com o ciclo de vida ágil iterativo-incremental do Scrum. Mostra como a nova metodologia foi aplicada em um projeto e as lições aprendidas. No final, exibe um quadro comparando o PosoMas com outras metodologias de engenharia de software orientada a agentes ágeis.

O trabalho de Ferreira et al. (2015) apresenta o desenvolvimento do jogo MedEduc, que foi realizado utilizando a metodologia ágil orientada a agente *Agile* PASSI e enfatiza as vantagens do uso da metodologia ágil para sistemas multiagentes. Esse trabalho foi outro utilizado como uma das comparações realizadas no estudo de caso desta dissertação.

González-Moreno et al. (2014) apresentam a definição de um processo ágil para a metodologia INGENIAS, baseado no conhecimento do processo de desenvolvimento Scrum. A nova abordagem também é baseada no metamodelo INGENIAS, mas é menos focada na especificação e mais no desenvolvimento de código.

O trabalho apresentado por Tenso et al. (2017) afirma que o artefato História de Usuário, utilizado em métodos ágeis, não é o suficiente para entender o panorama geral dos requisitos do sistema, mesmo em métodos que tentam resolver esse problema, e neste caso, propõem um novo método de Modelagem Orientado a Agentes Ágeis (AAOM).

Babar et al. (2017) formulam uma metodologia aplicada para uma resposta ágil à demanda de sistemas multiagentes utilizando micromodelos matemáticos, onde é proposto

uma técnica simples de Q-learning de maneira descentralizada. O trabalho também inclui a exploração de trocas de regras complexas, no contexto do mercado.

Noori e Kazemifard (2015) realizam uma avaliação da programação em pares, para determinar a influência da personalidade dos programadores e a dificuldade do problema na eficiência dos pares, onde é utilizado um sistema baseado em agente para simular o par e o indicador Myers-Briggs é usado para medir a personalidade de cada membro dos pares.

O trabalho apresentado por Al-azawi, Ayesh e Obaidy (2014) realiza uma investigação nos métodos de desenvolvimento de jogos e fornece um novo método de desenvolvimento de jogos baseado em modelos de desenvolvimento preditivo e adaptativo, criando uma cooperação com AOSE, que introduz uma metodologia híbrida denominada Metodologia de Desenvolvimento de Jogos Ágil para Agentes (AAGDM).

Wang (2018) apresenta uma simulação baseada em JADE, utilizando a modelagem baseada em agentes para simular os vários fatores que geram problemas, que envolvem o desempenho da equipe ágil e que afeta a entrega do software a cada sprint na estrutura ágil Scrum.

O trabalho apresentado por Yu et al. (2014) descreve a demonstração de um jogo multiagente para treinar estudantes na prática da engenharia de software ágil, onde os dados de comportamento relacionados ao processo de tomada de decisão de recursos são coletados de maneira discreta. Os dados podem fornecer aos pesquisadores, novas informações sobre o processo de tomada de decisão humana e ajuda-los a comparar os modelos propostos.

3 PROPOSTA ÁGIL DO O-MASE: *FRAMEWORK AGILE* O-MASE

Essa dissertação tem como proposta incorporar agilidade em um método orientado a agentes, que utiliza um processo tradicional de desenvolvimento.

Os resultados do estudo de caso de Ferreira (2015) identificaram a importância de se ter um metodologia orientada a agentes no desenvolvimento de Sistemas Multiagentes e que a melhor solução de sua pesquisa foi a que utilizou O-MaSE. Por isso, a metodologia O-MaSE foi escolhida para incorporação de agilidade.

Outra necessidade identificada na análise das metodologias ágeis orientadas a agentes encontradas na literatura, foi a combinação de diferentes soluções existentes no desenvolvimento ágil, por exemplo as metodologias *Agile* PASSI e INGENIAS-Scrum só incorporaram soluções especificas do XP e Scrum respectivamente.

Este capítulo detalha a proposta do *framework Agile* O-MaSE que além de incorporar fragmentos de mais de um método ágil também adiciona um valor e dois princípios MDA citados em 1.1.

3.1 Processo de Desenvolvimento do *Agile* O-MaSE

A visão geral do *framework Agile* O-MaSE com suas fases, artefatos e decisões é apresentada na Figura 32, destacando em vermelho os artefatos e fluxos propostos no Agile O-MaSE. O processo de desenvolvimento do *Agile* O-MaSE tem quatro fases: (i) Análise e Planejamento dos Requisitos; (ii) Projeto de Arquitetura e (iii) Codificação e (iv) Qualidade do *Software*.

As três primeiras fases do *framework Agile* O-MaSE foram adaptadas da metodologia O-MaSE, com as suas atividades e modelos sendo estendidas para uso do processo ágil.

A fase de Projeto detalhado da metodologia O-MaSE foi retirada do *framework Agile* O-MaSE, pois após a definição da arquitetura do sistema multiagentes, este é codificado. Assim a etapa de modelagem foi simplificada. Com a definição de seus agentes, comportamento e comunicação, começa-se a fase de codificação.

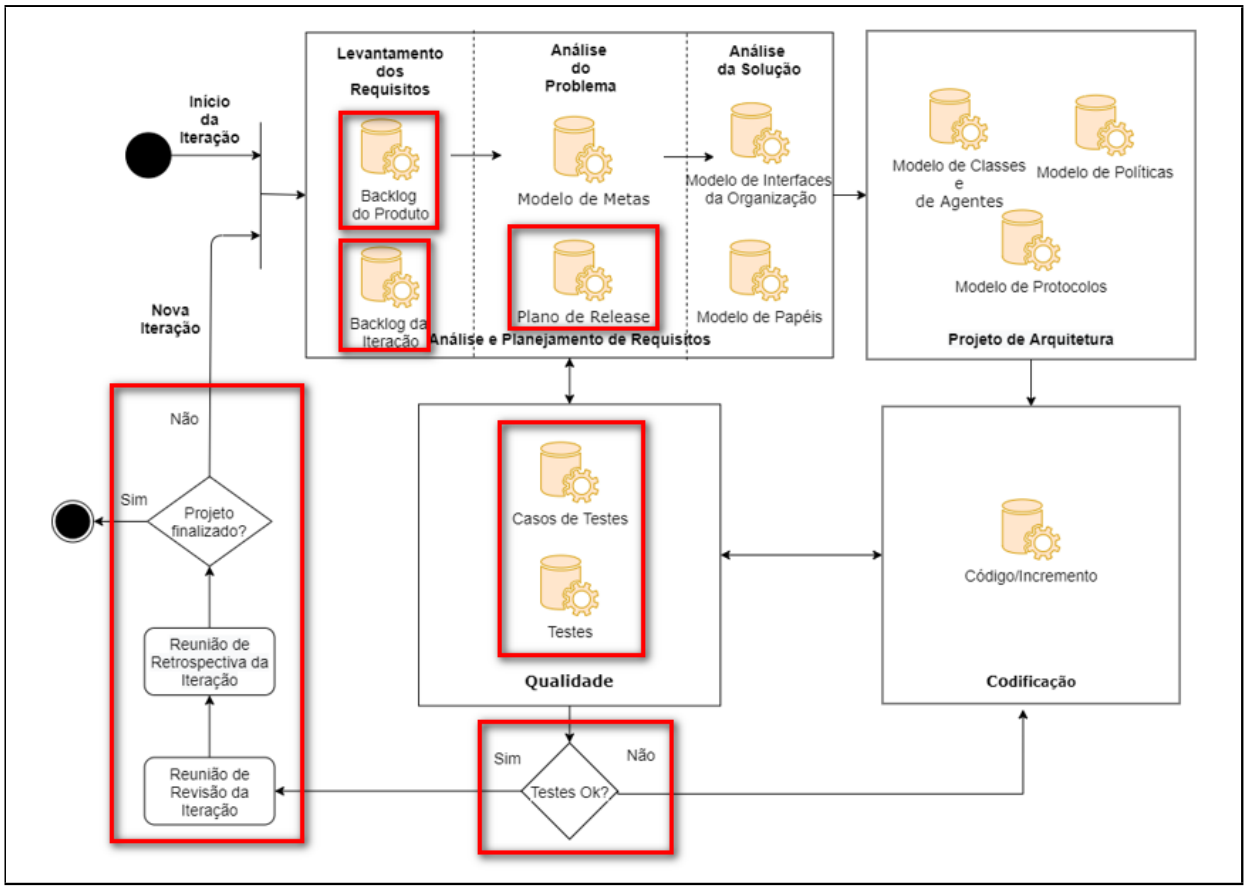
A fase de Qualidade do Software, que foi adicionada ao processo de desenvolvimento *Agile* O-MaSE está relacionada ao MDA. Essa fase teve como base o *framework* XP, que se destaca por ter introduzido como parte de seu método a abordagem ao desenvolvimento, *Test-Driven Development* (TDD), essa abordagem também foi utilizada em outro método ágil orientado a agentes, o *Agile* PASSI.

A inclusão dessa fase de qualidade teve como motivação o 2º Valor do MDA - "*Software* em funcionamento mais do que documentação abrangente" e os princípios "Entregar

Software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos" (3º Princípio do MDA) e "A entrega de Software funcional é a medida primária de progresso" (7º Princípio do MDA). A Figura 33 mostra essa junção desse valor e dos princípios do MDA mais o Desenvolvimento Orientado a Testes do inglês TDD.

As seguintes cerimônias foram incorporadas ao *Agile* O-MaSE: reunião de revisão da interação ou Sprint (reunião de entrega) e reunião de retrospectiva da interação ou Sprint. Essas reuniões caracterizam e controlam as iterações ágeis.

Figura 32 - Framework Agile O-MaSE



Fonte: A autora.

3.1.1 Análise e Planejamento dos Requisitos do *Agile* O-MaSE

A fase de Análise e Planejamento dos Requisitos da metodologia O-MaSE tem três atividades: (i) Levantamento dos Requisitos, (ii) Análise do Problema e (iii) Análise da Solução.

Para incorporação de boas práticas ágeis, nessa fase foram inseridos os seguintes produtos oriundos do *framework* ágil Scrum: *backlog* do produto, *backlog* da iteração e o plano de *release*.

Nessa atividade de requisitos é construído o *backlog* do produto com todos os requi-

Figura 33 - Motivação para Fase de Qualidade do Framework Agile O-MaSE

Fonte: A autora.

sitos funcionais e não funcionais que devem ser desenvolvidos na construção do software. Deve conter inclusive os *bugs* que forem encontrados nas versões anteriores com o cliente, para acompanhamento das correções.

O *backlog* do produto pode ser atualizado a qualquer momento e deve conter: item, descrição, tema, prioridade, tipo, estimativa de complexidade, iteração e situação.

Além disso nessa atividade foi inserido o produto *Backlog* da Iteração, que é construído de acordo com a meta/objetivo da iteração.

O *backlog* da iteração deve conter: meta da iteração, item, história do usuário, tarefas da história, duração das tarefas, distribuição das tarefas e situação da tarefa.

Na atividade de Análise do Problema além do modelo de metas, o plano de *release* é construído e deve constar: definição do critério de sucesso da *release*, estimativa dos itens do *backlog* do produto, definição do tamanho e quantidade de iterações, estimativa da velocidade, e associação dos itens do *backlog* do produto as iterações.

Na atividade Análise da Solução são definidos os modelos de papéis, da organização e da interface.

3.1.2 Projeto da Arquitetura

Na fase de Projeto da Arquitetura é realizada a modelagem da solução computacional do software.

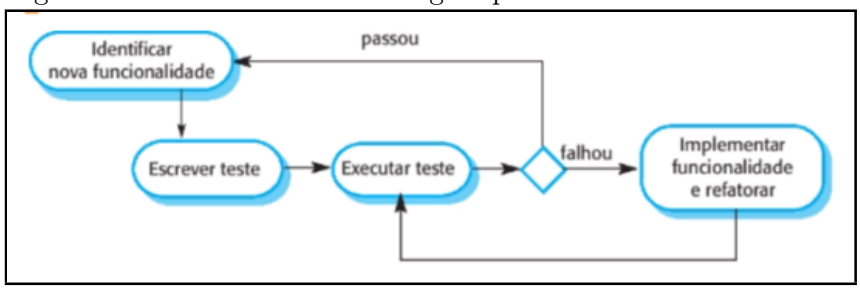
Nela são definidos os modelos de Classes de Agentes, de Protocolo e de Políticas Organizacionais. Após esses modelos caraterizados na O-MaSE como Projeto da Arquitetura, o código é implementado.

3.1.3 Codificação e Qualidade de Software

As fases de Codificação e Qualidade são realizadas juntas. A fase de qualidade do *Agile* O-MaSE foi definida com base no TDD, que é um conjunto de práticas voltadas para testes, focando a qualidade, onde o teste é escrito primeiro, antes da implementação do código.

O software é desenvolvido de forma incremental, em conjunto com um teste para esse incremento, assim, o próximo incremento não é construído até que o desenvolvimento do incremento atual passe no teste. O desenvolvedor ao escrever o teste já o implementa como um teste automatizado. O desenvolvimento de código e os testes são intercalados a cada iteração conforme a Figura 34. (BECK, 2002) e (JEFFRIES; MELNIK, 2007).

Figura 34 - Desenvolvimento Dirigido por Testes



Fonte: Sommerville (2019)

O TDD auxilia os programadores a clarear suas ideias sobre o que um segmento de código supostamente deve fazer Sommerville (2019). Para escrever um teste, é necessário entender a que ele se destina, e com esse entendimento é mais fácil escrever o código necessário.

O desenvolvedor tendo conhecimento ou compreensão incompleta para escrever os testes, o TDD não ajudará, mas como consequência o desenvolvedor identifica que não está compreendendo os requisitos o suficiente, para escrever o código necessário.

A fase de testes, no *framework Agile* O-MaSE é chamada de Qualidade e define duas atividades: Casos de Testes e Testes. Os Casos de Testes são escritos a cada iteração, seguindo os modelos definidos e os requisitos selecionadas no *backlog* da iteração, que se baseiam no que foi definido como meta/objetivo da iteração.

Os Testes, são realizados a cada funcionalidade desenvolvida e só serão liberados na entrega da iteração as funcionalidades que estiverem prontas, incluindo os testes.

As funcionalidades que não passarem nos testes e que não derem tempo de serem corrigidas dentro do time-box da iteração, devem ser atualizadas na situação das histórias no *backlog* da iteração, como *bug*, para que ela seja corrigida e entregue na próxima iteração.

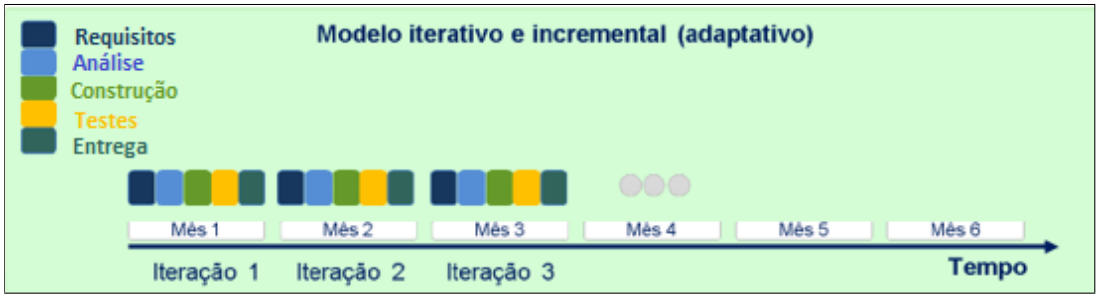
3.2 Modelos do *Agile* O-MaSE

Na metodologia O-MaSE são propostos dez modelos, sendo quatro na fase de Análise e Planejamento de Requisitos e seis na Fase de Projeto. No *Framework Agile* O-MaSE foram reduzidos a seis modelos. Na primeira fase o modelo de Metas, de Interface da Organização e de Papeis foram mantidos, pois estes apoiam a definição dos agentes do sistema. Na fase de projeto só os modelos relacionados ao Projeto da Arquitetura foram mantidos: Modelo de Classes de Agentes, Modelo de Protocolos e Modelo de Políticas. Assim o *framework Agile* O-MaSE ficou com 6 modelos conforme descrito no Quadro 14.

3.3 Consideração Final

Para finalizar o *framework Agile* O-MaSE, foi adicionado o modelo adaptativo que é clássico nos *frameworks* ágeis, para formalizar as iterações e incrementos como mostra a Figura 35, onde a cada iteração é realizado um grupo de requisitos, análise, construção, testes e entrega, assim sucessivamente até que sejam finalizados os requisitos ou até o prazo acordado no projeto.

Figura 35 - Modelo Adaptativo



Fonte: Material da TI.exames adaptado pela a autora.

O Quadro 14 apresenta as características das metodologias envolvidas no estudo de caso deste trabalho.

Quadro 14 - Comparação das metodologias: *Agile* Passi, Ingenias-Scrum e *Agile* O-MaSE

	Agile Passi	***Ingenias-Scrum***	**O-MaSE *Agile***
Origem	*Passi* e *XP*	*Ingenias* e *Scrum*	*O-MaSE,* *Scrum e* *XP*
Abordagem do Processo Original	Tradicional e Ágil	Tradicional e Ágil	Tradicional e Ágeis
Orientado a Agentes	Sim	Sim	Sim
Perspectiva de Desenvolvimento	Bottom-Up	Hybrid	Top-Down
Fases	- Requisitos, - Sociedade de agentes, - Código e - Teste	- Preparação e - *Sprints*	- Análise de Requisitos, - Projeto de Arquitetura, - Codificação e - Qualidade
Modelos Desenvolvidos	- Modelo de requisitos do sistema, - Modelo de sociedade de agente, - Modelo de testes e - Modelo de código	- Modelo de organização, - Modelo de agentes e ambiente, - Modelo de metas e tarefas, - Modelo de interação, - Modelo de componente e - Modelo de desenvolvimento	- Modelo de metas, - Modelo de interfaces da organização, - Modelo de papéis, - Modelo de classes de agentes, - Modelo de protocolo e - Modelo de políticas
Artefatos	- Histórias de usuários, - Diagrama de caso de uso, - Casos de uso, - Diagrama de ontologia de domínio (diagrama de classes); - Plano de testes e - Protótipo (código)	- *Backlog* do produto: Itens de *Backlog* do produto e Histórias de usuários, - Diagrama de organização (opcional), - Diagrama de agentes e ambiente, - *Backlog da Sprint*, - Diagrama de metas e tarefas, - Diagrama de interação e - Código	- *Backlog* do produto, - *Backlog* da iteração, - Modelo de metas, - Modelo de interfaces da organização, - Modelo de papéis, -Modelo de classes de agentes, - Modelo de protocolo, - Modelo de políticas, - Casos de testes e - Incremento (Código)
Papeis	Não define	- Dono do Produto (*Product Owner*), - *Scrum Master* e - Time *Scrum*	- Dono do Produto (*Product Owner*), - *Scrum Master* e - Time *Scrum*
Tamanho da Equipe	De 2 a 12 pessoas	Até 15 pessoas	Até 15 pessoas

Fonte: A autora.

4 ESTUDO DE CASO

Este capítulo descreve o estudo de caso de acordo com as etapas apresentadas na seção 1.3 desta dissertação, segundo Wohlin et al. (2012): 1. Design do estudo de caso; 2. Preparação para coleta de dados; 3. Coleta de dados e 4. Análise dos dados coletados.

4.1 Etapa 1 – Design do Estudo de Caso

O estudo de caso foi realizado na graduação do curso Ciência da Computação, na turma de IA com 14 alunos numa Universidade púbica do Estado do Rio de Janeiro.

Para introduzir os alunos no objetivo do estudo de caso, que foi desenvolver um jogo médico educacional utilizando uma abordagem ágil com orientação a agentes, foi realizada uma aula sobre metodologia ágil para nivelar o conhecimento dos alunos com relação ao tema agilidade.

Os alunos assinaram um termo de consentimento para participar do estudo de caso, com a garantia da confidencialidade e privacidade de seus dados pessoais, como nome e nota na disciplina, onde todo material entregue durante o estudo de caso só seria divulgado para fins acadêmicos, sem a exposição direta dos participantes.

Foi explicado o propósito do trabalho sobre as metodologias, cronogramas e detalhes do jogo a ser desenvolvido. Os alunos dividiram-se por livre escolha entre eles, em 4 grupos: 2 grupos com 4 alunos e 2 grupo com 3 alunos, onde cada grupo escolheu seu representante, que sorteou a metodologia que iriam trabalhar. A divisão dos grupos ocorreu da seguinte maneira:

- Grupo A recebeu o cronograma da metodologia *Agile* PASSI, Figura 37;
- Grupo C recebeu o cronograma da metodologia Ingenias-Scrum, Figura 38;
- Grupo E e F receberam o cronograma da metodologia *Agile* O-MaSE, Figura 36.

Os cronogramas continham todos modelos e artefatos que deveriam ser entregues ao final de cada iteração, assim como, o incremento de software referente ao objetivo da iteração.

A lista de alunos inscritos na matéria de IA, apresentava uma quantidade alunos que sugeria um planejamento de dois grupos para cada metodologia, porém houveram muitos alunos que desistiram da disciplina, formando inicialmente apenas 1 grupo por metodologia.

No decorrer das aulas, apareceram mais alunos sendo necessário a formação de mais um grupo, onde foi passada a nova metodologia: *Agile* O-MaSE. A escolha pela

Agile O-MaSE foi para garantir que a metodologia fosse utilizada, já que não houve a possibilidade de termos 2 grupos de cada metodologia, criando assim o grupo F:

- Grupo F recebeu o cronograma da metodologia *Agile* O-MaSE, Figura 36.

O projeto foi dividido em três iterações/sprints num *time-box* de 2 semanas, que foi realizado no período de 10 de outubro a 05 de dezembro de 2019. Os encontros aconteciam todas as quintas-feiras, das 08:50hs às 10:30hs.

Figura 36 - Cronograma Agile O-MaSE

Id	EDT	Nome da tarefa	Início	Término	Nomes dos recursos
1	**1**	**Projeto de Desenvolvimento do Jogo MedEduc - Metodologia Agile O-MaSE**	**Qui 17/10/19**	**Qui 28/11/1**	**Grupo E;Grupo F**
2	**1.1**	**Iteração 1**	**Qui 17/10/19**	**Qui 31/10/1**	
3	**1.1.1**	**Fase de Análise de Requisitos**			
4	1.1.1.1	Atividade Levantamento dos Requisitos: Backlog do Produto e Backlog da Iteração			
5	1.1.1.2	Atividade Análise do Problema: Modelar e Refinar Metas (Diagrama) e Plano de Release			
6	1.1.1.3	Atividade Análise da Solução: Modelo de Interface com a Organização (Diagrama) e Modelo de Papeis (Diagrama)			
7	**1.1.2**	**Fase de Qualidade**			
8	1.1.2.1	Casos de Testes			
9	**1.1.3**	**Fase Projeto de Arquitetura**			
10	1.1.3.1	Modelar Classes de Agentes, Modelar Políticas e Modelar Protocolos			
11	**1.1.4**	**Fase de Codificação**	**Qui 24/10/19**	**Qui 24/10/1**	
12	1.1.4.1	Gerar Incremento			
13	1.1.4.2	Testar a codificação			
14	**1.1.5**	**Fase de Qualidade**			
15	1.1.5.1	Realizar Testes			
16	1.1.6	Reunião de Revisão da Iteração - Apresentação dos modelos e Protótipo	Qui 31/10/19	Qui 31/10/1	Todos
17	1.1.7	Reunião de Restropectiva da Iteração	Qui 31/10/19	Qui 31/10/1	Grupo E;Grupo F
18	**1.2**	**Iteração 2**	**Qui 31/10/19**	**Qui 31/10/1**	
19	**1.2.1**	**Fase de Análise de Requisitos**	Qui 31/10/19	Qui 31/10/1	
20	**1.2.2**	**Fase de Qualidade**			
21	**1.2.3**	**Fase Projeto de Arquitetura**			
22	**1.2.4**	**Fase de Codificação**	**Qui 07/11/19**	**Qui 07/11/1**	
23	**1.2.5**	**Fase de Qualidade**			
24	1.2.6	Reunião de Revisão da Iteração - Apresentação dos modelos e Protótipo	Qui 14/11/19	Qui 14/11/1	Todos
25	1.2.7	Reunião de Restropectiva da Iteração	Qui 14/11/19	Qui 14/11/1	Grupo E;Grupo F
26	**1.3**	**Iteração 3**	**Qui 14/11/19**	**Qui 14/11/1**	
27	**1.3.1**	**Fase de Análise de Requisitos**	**Qui 14/11/19**	**Qui 14/11/1**	
28	**1.3.2**	**Fase de Qualidade**			
29	**1.3.3**	**Fase Projeto de Arquitetura**			
30	**1.3.4**	**Fase de Codificação**	**Qui 21/11/19**	**Qui 21/11/1**	
31	1.3.5	Reunião de Revisão da Iteração - Apresentação dos modelos e Protótipo	Qui 28/11/19	Qui 28/11/1	Todos
32	1.3.6	Reunião de Restropectiva da Iteração	Qui 28/11/19	Qui 28/11/1	Grupo E;Grupo F

Fonte: A autora.

Figura 37 - Cronograma Agile Passi

Id	EDT	Nome da tarefa	Início	Término	Nomes dos recursos
1	**1**	**Projeto de Desenvolvimento do Jogo MedEduc - Metodologia Agile Passi**	**Qui 17/10/19**	**Qui 28/11/19**	**Grupo A;Grupo B**
2	**1.1**	**Iteração 1**	**Qui 17/10/19**	**Qui 31/10/19**	
3	**1.1.1**	**Fase de Requisitos**	**Qui 17/10/19**	**Qui 17/10/19**	
4	1.1.1.1	Modelo de Requisitos: Plano de Iteração (histórias de usuários) e Descrição dos requisitos de sub-domínios (diagrama de caso de uso)			
5	**1.1.2**	**Fase de Sociedade de Agentes**			
6	1.1.2.1	Modelo de Sociedade de Agente: Identificação dos agentes (casos de uso)			
7	1.1.2.2	Modelo de Sociedade de Agente: Descrição da ontologia de domínio (diagrama de classes			
8	**1.1.3**	**Fase de Plano de Testes**	**Qui 24/10/19**	**Qui 24/10/19**	
9	1.1.3.1	Plano de testes			
10	**1.1.4**	**Fase de Codificação**			
11	1.1.4.1	Modelo de Código: Refatoração do código e Codificação			
12	1.1.4.2	Testes do código			
13	1.1.5	Reunião de entrega - Apresentação dos modelos e Protótipo da Iteração	Qui 31/10/19	Qui 31/10/19	Todos
14	**1.2**	**Iteração 2**	**Qui 31/10/19**	**Qui 14/11/19**	
15	**1.2.1**	**Fase de Requisitos**	**Qui 31/10/19**	**Qui 31/10/19**	
16	**1.2.2**	**Fase de Sociedade de Agentes**			
17	**1.2.3**	**Fase de Plano de Testes**	**Qui 07/11/19**	**Qui 07/11/19**	
18	**1.2.4**	**Fase de Codificação**			
19	1.2.5	Reunião de entrega - Apresentação dos modelos e Protótipo da Iteração	Qui 14/11/19	Qui 14/11/19	Todos
20	**1.3**	**Iteração 3**	**Qui 14/11/19**	**Qui 28/11/19**	
21	**1.3.1**	**Fase de Requisitos**	**Qui 14/11/19**	**Qui 14/11/19**	
22	**1.3.2**	**Fase de Sociedade de Agentes**			
23	**1.3.3**	**Fase de Plano de Testes**	**Qui 21/11/19**	**Qui 21/11/19**	
24	**1.3.4**	**Fase de Codificação**			
25	1.3.5	Reunião de entrega - Apresentação dos modelos e Protótipo da Iteração	Qui 28/11/19	Qui 28/11/19	Todos

Fonte: A autora.

Figura 38 - Cronograma Ingenias Scrum

Id	EDT	Nome da tarefa	Início	Término	Nomes dos recursos
1	**1**	**Projeto de Desenvolvimento do Jogo MedEduc - Metodologia Ingenias-Scrum**	**Qui 10/10/19**	**Qui 28/11/19**	**Grupo C;Grupo D**
2	**1.1**	**Fase de Preparação**	**Qui 10/10/19**	**Qui 17/10/19**	
3	1.1.1	Atividade de Iniciação do Backlog do Produto: itens do backlog do produto (PBI)	**Qui 10/10/19**	**Qui 10/10/19**	Dono do Produto;TIme Scrum
4	1.1.2	Atividade de Preparação: Modelo de Organização (opicional) e Modelo de Agente e Ambiente: diagrama de Agentes interagindo com o ambiente			TIme Scrum
5	1.1.3	Atividade Planejar Release: plano de release			Scrum Master;Dono do Produto;TIm
6	**1.2**	**Fase de Sprints**	**Qui 17/10/19**	**Qui 28/11/19**	
7	**1.2.1**	**Sprint 1**	**Qui 17/10/19**	**Qui 31/10/19**	
8	1.2.1.1	Atividade planejar a sprint: backlog da Sprint	**Qui 17/10/19**	**Qui 17/10/19**	
9	**1.2.1.2**	**Trabalho diário:**			
10	1.2.1.2.1	Atualizar o Backlog do Produto			
11	1.2.1.2.2	Refinar o Modelo Organizacional e Modelo de Agente e Ambiente			
12	1.2.1.2.3	Criar Modelo de Metas e Tarefas			
13	1.2.1.2.4	Mostrar a execução das tarefas usando o Modelo de Interação			
14	1.2.1.2.5	Criar um Modelo de Componente e Modelo de um Desenvolvimento			
15	1.2.1.2.6	Especificar template de Códigos para ser seguido - Codificação	Qui 24/10/19	Qui 24/10/19	
16	1.2.1.2.7	Validar o código			
17	1.2.1.3	Reunião de Revisão da Sprint - Apresentação dos modelos e Incremento da Iteração (Funcionalidades implementadas)	Qui 31/10/19	Qui 31/10/19	Todos
18	1.2.1.4	Reunião de Restropctiva da Sprint	Qui 31/10/19	Qui 31/10/19	Grupo C;Grupo D
19	**1.2.2**	**Sprint 2**	**Qui 31/10/19**	**Qui 14/11/19**	
20	1.2.2.1	Atividade planejar a sprint: backlog da Sprint	Qui 31/10/19	Qui 31/10/19	
21	1.2.2.2	Trabalho diário	Qui 07/11/19	Qui 07/11/19	
22	1.2.2.3	Reunião de Revisão da Sprint - Apresentação dos modelos e Incremento da Iteração	Qui 14/11/19	Qui 14/11/19	Todos
23	1.2.2.4	Reunião de Restropctiva da Sprint	Qui 14/11/19	Qui 14/11/19	Grupo C;Grupo D
24	**1.2.3**	**Sprint 3**	**Qui 14/11/19**	**Qui 28/11/19**	
25	1.2.3.1	Atividade planejar a sprint: backlog da Sprint	Qui 14/11/19	Qui 14/11/19	
26	1.2.3.2	Trabalho diário			
27	1.2.3.3	Reunião de Revisão da Sprint - Apresentação dos modelos e Incremento da Iteração	Qui 21/11/19	Qui 21/11/19	Todos
28	1.2.3.4	Reunião de Restropctiva da Sprint	Qui 28/11/19	Qui 28/11/19	Grupo C;Grupo D

Fonte: A autora.

4.1.1 Objetivo

O objetivo geral do estudo de caso foi desenvolver o mesmo jogo médico educacional, utilizando três diferentes metodologias, sendo todas ágeis e orientadas a agentes: *Agile* PASSI, Ingenias Scrum e a proposta: *Agile* O-MaSE. E a partir desta prática realizar uma pesquisa exploratória.

O jogo foi escolhido por ter sido utilizado em dois estudos de casos anteriores, Ferreira (2015) e Novo (2018), para que seja possível a comparação do resultado do estudo de caso deste trabalho com as metodologias utilizadas nos trabalhos dos dois outros pesquisadores.

4.1.2 Descrição do Caso

O caso estudado é o desenvolvimento de um jogo médico educacional, que tem como objetivo ensinar de maneira lúdica, jovens estudantes da carreira médica.

O jogo é dividido em 5 níveis de dificuldades, sendo 1 - o mais baixo e 5 - o mais alto, onde todos os níveis continham 10 perguntas. A cada nível ultrapassado, o jogador ganha um equipamento médico para montar o seu consultório.

Logo no início da aplicação, o jogador faz o seu cadastro de login e senha, para ter acesso ao jogo. Na sequência realiza um nivelamento, que também consiste em 10 perguntas, para que o sistema possa sugerir qual nível de conhecimento o jogador se encaixava.

Para todas as questões o jogador recebe o resultado do que foi respondido, se conseguiu acertar e quando não acertava o jogo dever informar qual era a resposta correta, explicando de maneira didática, através de vídeo, imagem ou texto descritivo.

O jogo contém 22 requisitos funcionais, descritos no Backlog do Produto (Apêndice A), que foram previamente priorizados pela mestranda que fazia o papel da dona do produto. Os requisitos contém uma coluna de Tema para ajudar a equipe no entendimento das funcionalidades.

A cada iteração/sprint um objetivo tinha que ser alcançado, conforme mostrado nos objetivos abaixo:

Iteração/sprint 1 cujo objetivo é poder realizar as 10 questões do Nivelamento, responder e ter resposta de questões variadas utilizando texto, imagem, áudio ou vídeo, podendo rever as questões já respondidas, saber das respostas corretas quando estiverem erradas, poder parar e reiniciar o nivelamento e no final receber a orientação de qual nível o usuário deve seguir no jogo.

Iteração/sprint 2 cujo objetivo é poder realizar as 10 questões dos Níveis 1 ao 3 do jogo, e realizar as 10 questões do Reforço, caso seja necessário e de acordo com o *feedback*

dos níveis, caso não seja necessário o usuário deve receber a premiação. Responder e ter resposta de questões variadas utilizando texto, imagem, áudio ou vídeo, podendo rever as questões já respondidas, saber das respostas corretas quando estiverem erradas, poder parar e reiniciar os Níveis e o Reforço.

Iteração/sprint 3 cujo objetivo é poder realizar as 10 questões dos Níveis 4 ao 5 do jogo, e realizar as 10 questões do Reforço, caso seja necessário e de acordo com o *feedback* dos níveis, caso não seja necessário o usuário deve receber a premiação. Responder e ter resposta de questões variadas utilizando texto, imagem, áudio ou vídeo, podendo rever as questões já respondidas, saber das respostas corretas quando estiverem erradas, poder parar e reiniciar os Níveis e o Reforço. Após responder todos os níveis exibir o consultório pronto com os 5 equipamentos.

Os grupos eram livres para selecionar no *backlog* a quantidade de requisitos que quisessem desenvolver, contanto que correspondessem no mínimo ao objetivo proposto na iteração.

4.1.3 Teoria

A teoria para desenvolver a direção da pesquisa, foi baseada:

i) Nas descrições das metodologias, apresentadas no capítulo 1 deste trabalho, onde cada grupo recebeu o descritivo da metodologia que iria seguir para desenvolvimento;

ii) No histórico dos pesquisadores Ferreira (2015) e Novo (2018), que em suas dissertações fizeram seus quadros de referências entre as metodologias que estudaram, conforme anexo A e anexo B; a análise dos dados das pessoas que desenvolveram o jogo;

iii) No Quadro de comparação construído com as três metodologias ágeis orientadas a agentes, destacado na seção 3.1 da dissertação, onde a metodologia *Agile* PASSI é do quadro de referência de Ferreira (2015) e a metodologia Ingenias Scrum é do quadro de referência de Novo (2018) e a metodologia *Agile* O-MaSE, que foi criada neste trabalho.

4.1.4 Perfil dos Participantes da Pesquisa

Os alunos responderam a um questionário de caracterização dos participantes, Apêndice E, para identificação do nível de conhecimento nos assuntos agilidade, metodologia orientada a agentes e jogos, que seriam abordados no estudo de caso.

As expectativas dos alunos com relação ao trabalho que iriam iniciar, também foram requeridas e estão descritas no apêndice F. A partir do questionário de caracterização, foi criado o Quadro 15 e Quadro 16, onde foi possível realizar algumas comparações entre os grupos.

Sobre a faixa etária dos alunos, a maioria possuíam faixa etária abaixo dos 25 anos. No grupo A, 100% estava abaixo dos 25 anos; no grupo C, 75% estava abaixo dos 25 anos; No grupo E, 100% estava acima dos 25 anos e no grupo F, 50% estavam abaixo dos 25 anos.

Sobre ter experiência em sistemas multiagentes, a maioria dos alunos não tinha experiência.

Sobre o conhecimento da metodologia proposta, que cada grupo sorteou, a maioria não conhecia metodologia. É provável que os poucos que responderam ter algum conhecimento sobre a metodologia, possam ter se referido sobre a parte ágil do que a parte agente.

Sobre o conhecimento de metodologia ágil, a maioria que respondeu tinha conhecimento. O grupo A, respondeu que tinha 100% de conhecimento; O grupo C, 50% respondeu que tinha conhecimento; O grupo E, 65% respondeu que tinha conhecimento e no grupo F, 100% respondeu que tinha conhecimento.

Sobre ter participado em desenvolvimento de jogos, a maioria respondeu que nunca tinha participado. Sendo que, no grupo F, 50% do alunos havia participado.

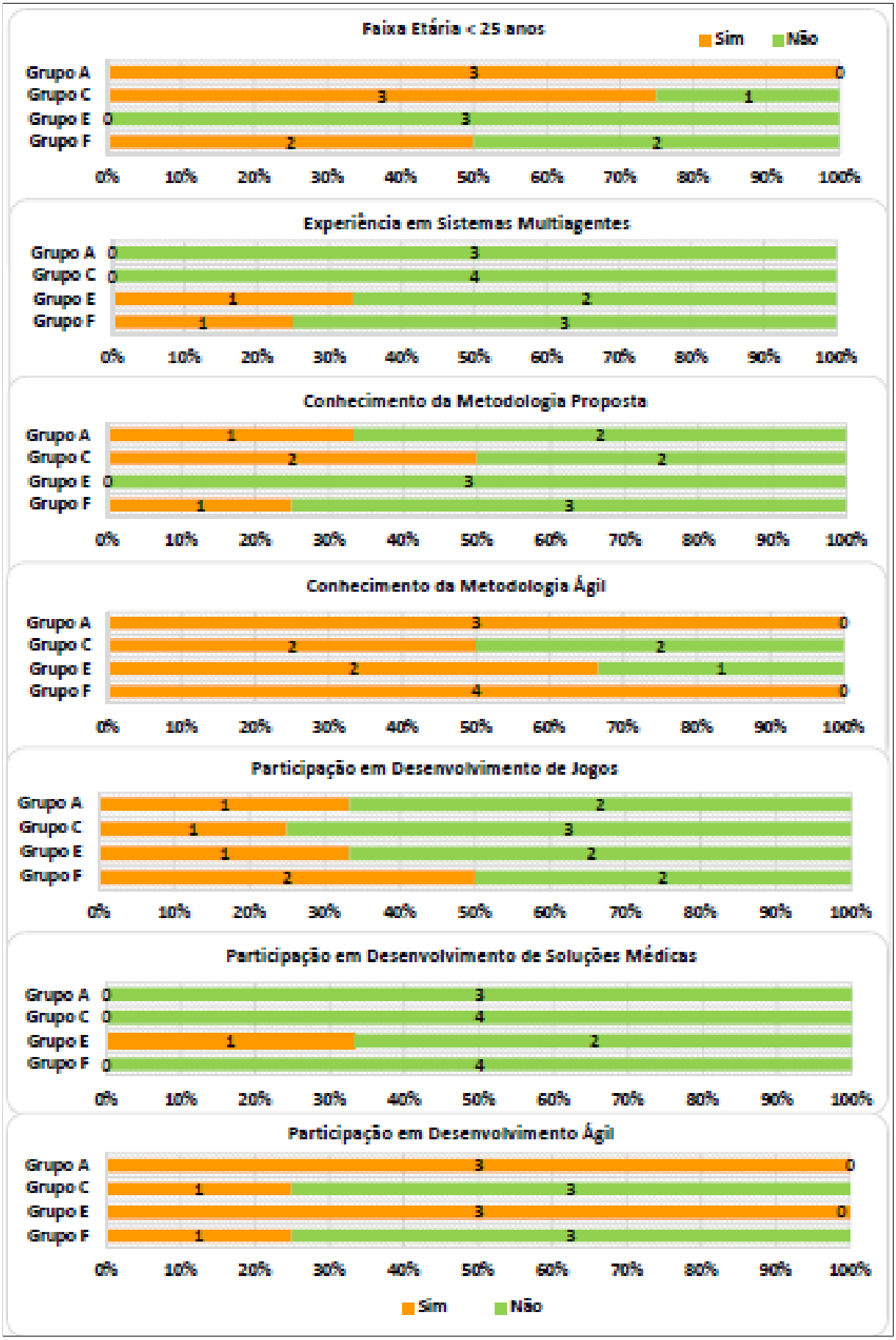
Sobre ter participado de desenvolvimento de soluções médicas, apenas 1 pessoa (30%) do grupo E afirmou que participou.

Sobre a participação em desenvolvimento ágil, os grupos A e E responderam 100% que haviam participado, já os grupos C e F apenas 1 pessoas (20%) havia participado de um projeto ágil.

Sobre o nível de conhecimento em agentes: o grupo A afirmou ter 100% de bom conhecimento; o grupo C, 50% tinha bom conhecimento, 25% algum conhecimento e 25% um conhecimento regular.

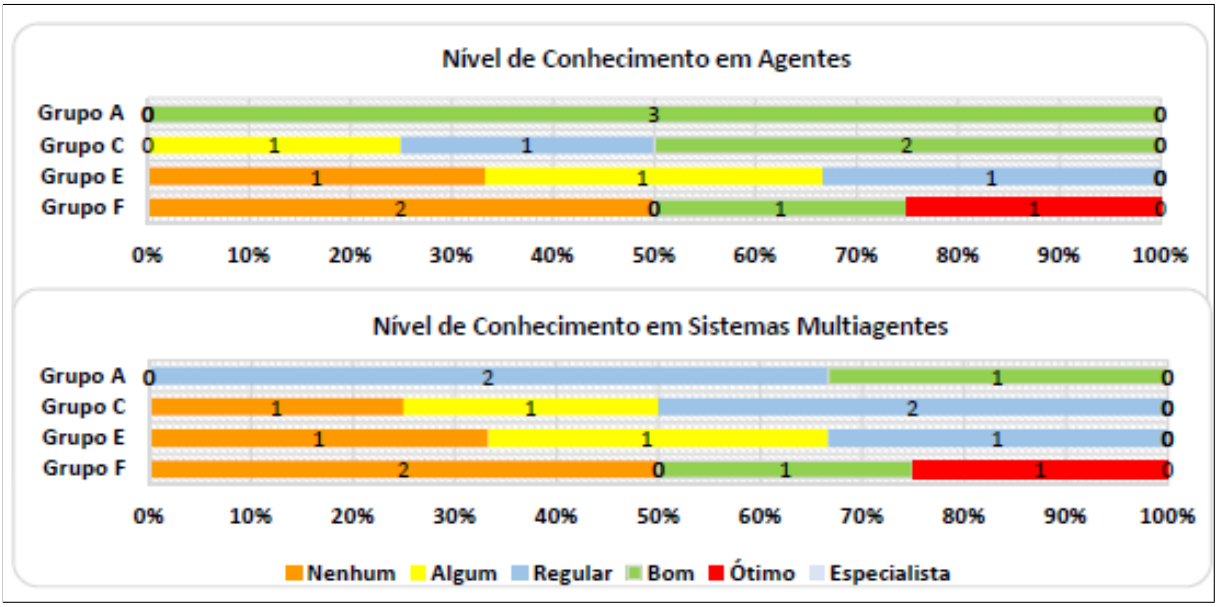
Sobre o nível de conhecimento em sistemas multiagentes: o grupo A tem 65% de regular conhecimento e 35% de bom conhecimento; o grupo C, tem 25% de nenhum conhecimento, 25% de algum conhecimento e 50% um conhecimento regular; o grupo E, tem 35% de nenhum conhecimento, 30% de algum conhecimento e 35% de regular conhecimento; o grupo F, tem 50% de nenhum conhecimento, 25% de bom conhecimento e 25% de ótimo conhecimento.

Quadro 15 - Comparação das respostas do questionário de caracterização



Fonte: A autora.

Quadro 16 - Comparação das respostas do questionário de caracterização



Fonte: A autora.

4.1.5 Método de Coleta de Dados

Para o estudo de caso foi utilizado o método de categoria independente, conforme definido por Lethbridge, Sim e Singer (2005) descrito na seção 1.3 desta dissertação, que utiliza a análise de documentação entregue pelos grupos de acordo com cada metodologia.

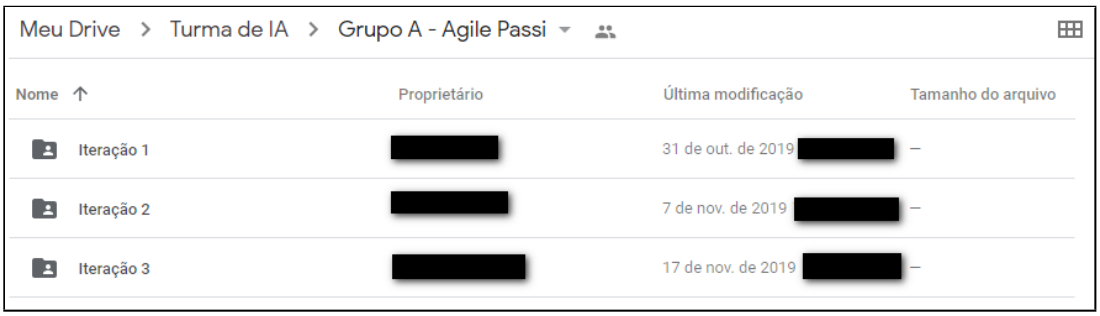
Os grupos faziam as suas entregas no último dia de cada iteração, e a coleta dos dados era realizada no dia seguinte após a entrega.

4.1.6 Estratégia de Seleção

Para cada grupo foi criado uma pasta no google drive, Figura 39 a Figura 42, para que no dia da entrega da iteração os alunos pudessem depositar os produtos que seriam entregues (documentação e o incremento de *software*).

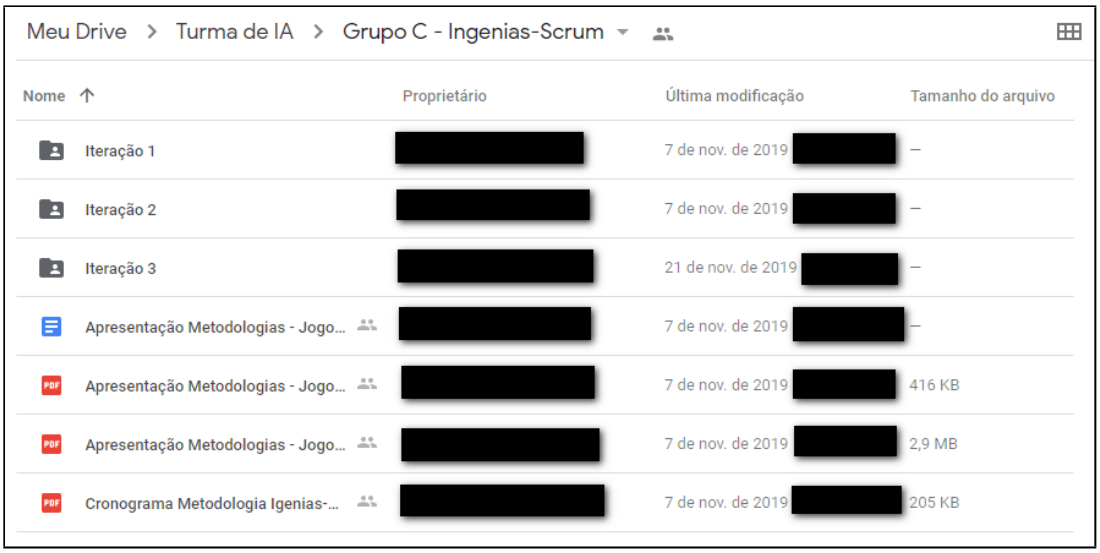
Como estratégia de seleção, os artefatos eram buscados em cada pasta do drive e conferidos de acordo com cada metodologia.

Figura 39 - Armazenamento das entregas dos projetos - Google drive - Grupo A



Fonte: A autora.

Figura 40 - Armazenamento das entregas dos projetos - Google drive - Grupo C



Fonte: A autora.

Figura 41 - Armazenamento das entregas dos projetos - Google drive - Grupo E

Fonte: A autora.

Figura 42 - Armazenamento das entregas dos projetos - Google drive - Grupo F

Fonte: A autora.

4.2 Etapa 2 – Preparação da Coleta de Dados

Para contemplar a etapa 2, preparação da coleta de dados, foi criado o documento protocolo para acompanhamento do estudo de caso, apêndice D.

A cada encontro o documento protocolo era alterado para coletar dados e informações a fim de registrar as ocorrências durante o estudo de caso, de forma particular. O mesmo era versionado por data de encontro.

4.3 Etapa 3 – Coleta de Dados

Em cada iteração os grupos realizavam as suas entregas, que eram conferidas de acordo com cada metodologia, conforme Figura 43, Figura 44, Figura 45 e Figura 46.

Figura 43 - Entregas do Grupo A

Grupo A – Agile Passi (3 alunos)			
Produtos	**Iteração 1 (De 17/10 a 31/10)**	**Iteração 2 (De 31/10 a 14/11)**	**Iteração 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Requisitos			
Preencher Plano de Iteração	Ok	Ok	Ok
Preencher Backlog do Produto	Ok	Ok	Ok
Preencher Backlog da Iteração	Ok	Ok	Ok
Descrição dos Requisitos de Subdomínios (diagrama UC + descrição UC)	OK	Ok	Ok
Fase de Sociedade de Agentes – Modelo de Sociedade de Agentes			
Identificação dos agentes (diagrama UC agrupados por agentes)	Não fez	Ok	Ok
Descrição da ontologia de domínio (diagrama de classes da UML)	Parcial	Parcial	Parcial
Fase de Plano de Testes			
Plano de Testes com Roteiro/Casos de Testes	Não fez	Ok	Ok
Fase de Codificação			
Modelo de Código	Não fez	Ok	Ok
Protótipo de acordo com o Backlog da Iteração	Ok	Ok	Ok

Fonte: A autora.

Figura 44 - Entregas do Grupo C

Grupo C – Ingenias-Scrum (4 alunos)				
Produtos	**Fase de Preparação (De 10/10 a 17/10)**	**Sprint 1 (De 17/10 a 31/10)**	**Sprint 2 (De 31/10 a 14/11)**	**Sprint 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Preparação				
Preencher Plano de Release	Uma semana a mais do que os outros grupos.	Ok	Ok	Ok
Preencher Backlog do Produto		Ok	Ok	Ok
Preencher Itens do Backlog do Produto (PBI)		Ok	Ok	Ok
Modelo de Organização (opcional)		Não fez	Ok	Ok
Modelo de Agente e Ambiente		Ok	Ok	Ok
Fase de Sprints				
Preencher Backlog da Sprint		Ok	Ok	Ok
Modelo de Metas e Tarefas		Não fez	Ok	OK
Modelo de Iteração		Não fez	Não fez	Ok
Modelo de Componente		Não fez	Não fez	Ok
Modelo de Desenvolvimento		Não fez	Não fez	Parcial
Protótipo de acordo com o Backlog da Sprint		Vídeo *	Vídeo	Vídeo

Fonte: A autora.

Figura 45 - Entregas do Grupo E

Grupo E – Agile O-MaSE (3 alunos)			
Produtos	**Iteração 1 (De 17/10 a 31/10)**	**Iteração 2 (De 31/10 a 14/11)**	**Iteração 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Análise de Requisitos			
Atividade Levantamento dos Requisitos: *Preencher* Backlog do Produto	Ok	Ok	Ok
Atividade Levantamento dos Requisitos: *Preencher* Backlog da Iteração	Ok	Ok	Ok
Atividade Análise do Problema: Modelar e Refinar Metas	Não fez	Ok	Ok
Atividade Análise do Problema: *Preencher* Plano de Release	Não fez	Ok	Ok
Atividade Análise da Solução: Modelo de Interface com a Organização	Não fez	Ok	Ok
Atividade Análise da Solução: Modelo de Papeis	Não fez	Ok	Ok
Fase de Qualidade			
Casos de Testes	Não fez	Não fez	Ok
Fase de Projeto e Arquitetura			
Modelar Classes de Agentes	Não fez	Não fez	Ok
Modelar Políticas	Não fez	Não fez	Ok
Modelar Protocolos	Não fez	Não fez	Ok
Fase de Codificação			
Incremento de acordo com o Backlog da Iteração (Jogo)	Não fez	Não fez	Disponibilizou APK *

Fonte: A autora.

Figura 46 - Entregas do Grupo F

Grupo F – Agile O-MaSE (4 alunos)			
Produtos	**Iteração 1 (De 17/10 a 31/10)**	**Iteração 2 (De 31/10 a 14/11)**	**Iteração 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Análise de Requisitos			
Atividade Levantamento dos Requisitos: *Preencher* Backlog do Produto	Ok	Ok	Ok
Atividade Levantamento dos Requisitos: *Preencher* Backlog da Iteração	Ok	Ok	Ok
Atividade Análise do Problema: Modelar e Refinar Metas	Ok	Ok	Ok
Atividade Análise do Problema: *Preencher* Plano de Release	Ok	Ok	Ok
Atividade Análise da Solução: Modelo de Interface com a Organização	Ok	Ok	Ok
Atividade Análise da Solução: Modelo de Papeis	Ok	Ok	Ok
Fase de Qualidade			
Casos de Testes	Parcial	Ok	Ok
Fase de Projeto e Arquitetura			
Modelar Classes de Agentes	Ok	Ok	Ok
Modelar Políticas	Não fez	Não fez	Ok
Modelar Protocolos	Ok	Ok	Ok
Fase de Codificação			
Incremento de acordo com o Backlog da Iteração (Jogo)	Ok	Ok	Ok

Fonte: A autora.

4.4 Etapa 4 – Análise dos Dados Coletados

Na etapa de análise dos dados coletados, foram criadas as seguintes pontuações para medição do valor das entregas:

- 0 ponto por não realizar a entrega;
- 1 ponto por entrega parcial;
- 3 pontos por entrega total;
- 5 pontos por incremento (desenvolvimento entregue de acordo com o objetivo).

Para realizar a medição, foram atribuídos os pontos de acordo com as entregas, depois feita a soma dos pontos por sprint/iteração e verificado o percentual dos pontos em relação aos pontos totais das entregas da Sprint (100%).

A pontuação das entregas ocorreram em cada iteração e de acordo com o objetivo proposta na iteração, caso contrário perderia o conceito ágil para ser tratado como um desenvolvimento cascata (tradicional), se fosse pontuada apenas na última iteração. Os dados que foram coletados no estudo de caso, produziram as pontuações alcançadas considerando as entregas por iteração/sprint.

- Grupo A (3 alunos) – *Agile* PASSI

  O Grupo A (Figura 47) utilizou a metodologia *Agile* PASSI, realizou 62% na 1a iteração, 93% na 2a iteração e 93% na 3a iteração.

  O grupo utilizou bem as 3 iterações/sprints. Entregaram todos os artefatos a partir da 2ª iteração, sendo 1 deles de forma parcial.

  O jogo final está completo, com login (identificação do usuário), nivelamento, níveis e reforço, habilitando os prêmios, que a cada nível é liberado quando o usuário passa no nível. O jogo não tem uma usabilidade tão adequada e uma aparência poluída com relação a fonte e cores, mas dentre os requisitos do *backlog*, não haviam requisitos não funcionais. Ao longo do projeto tiraram muitas dúvidas sobre a parte ágil e os requisitos.

Figura 47 - Pontuação do Grupo A

Grupo A – Agile Passi (3 alunos)			
Produtos	**Iteração 1 (De 17/10 a 31/10)**	**Iteração 2 (De 31/10 a 14/11)**	**Iteração 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Requisitos			
Preencher Plano de Iteração	3	3	3
Preencher Backlog do Produto	3	3	3
Preencher Backlog da Iteração	3	3	3
Descrição dos Requisitos de Subdomínios (diagrama UC + descrição UC)	3	3	3
Fase de Sociedade de Agentes – Modelo de Sociedade de Agentes			
Identificação dos agentes (diagrama UC agrupados por agentes)	0	3	3
Descrição da ontologia de domínio (diagrama de classes da UML)	1	1	1
Fase de Plano de Testes			
Plano de Testes com Roteiro/Casos de Testes	0	3	3
Fase de Codificação			
Modelo de Código	0	3	3
Protótipo de acordo com o Backlog da Iteração	5	5	5
% ENTREGA POR ITERAÇÃO	**62%**	**93%**	**93%**

Fonte: A autora.

- Grupo C (4 alunos) – Ingenias Scrum

  O Grupo C (Figura 48) utilizou a metodologia Ingenias-Scrum, realizou 43% na 1a iteração, 60% na 2a iteração e 80% na 3a iteração.

  Tiveram 1 semana a mais, por causa da metodologia. Entregaram todos os artefatos na 3ª iteração, porém 1 deles de forma parcial. Em todas as iterações disponibilizaram o jogo através de vídeo, não sendo possível jogar.

  O vídeo da primeira iteração mostrava apenas a tentativa de identificação do usuário, que não funcionava. Os demais vídeos também não atingiram os objetivos de cada iteração.

Figura 48 - Pontuação do Grupo C

Grupo C – Ingenias-Scrum (4 alunos)				
Produtos	**Fase de Preparação (De 10/10 a 17/10)**	**Sprint 1 (De 17/10 a 31/10)**	**Sprint 2 (De 31/10 a 14/11)**	**Sprint 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Preparação				
Preencher Plano de Release	Uma semana a mais do que os outros grupos.	3	3	3
Preencher Backlog do Produto		3	3	3
Preencher Itens do Backlog do Produto (PBI)		3	3	3
Modelo de Organização (opcional)		0	3	3
Modelo de Agente e Ambiente		3	3	3
Fase de Sprints				
Preencher Backlog da Sprint		3	3	3
Modelo de Metas e Tarefas		0	3	3
Modelo de Iteração		0	0	3
Modelo de Componente		0	0	3
Modelo de Desenvolvimento		0	0	1
Protótipo de acordo com o Backlog da Sprint		0	0	0
% ENTREGA POR ITERAÇÃO		**43%**	**60%**	**80%**

Fonte: A autora.

- Grupo E (3 alunos) – Agile O-MaSE

  O Grupo E logo no início ficou somente com 2 alunos, mas na 2a iteração, apareceu um novo aluno sem grupo se integrou ao grupo. Esse grupo (Figura 49) utilizou a metodologia *Agile* O-MaSE, realizou 17% na 1a iteração, 51% na 2a iteração e 80% na 3a iteração.

  Foram prejudicados pela desestruturação do grupo. Demoraram na organização das atividades e divisão das tarefas entre eles. Entregaram todos os artefatos na 3ª iteração, mas o jogo disponibilizado para celular, só tem praticamente as explicações iniciais. Não possui: identificação do usuário, o nivelamento está com erro (não consegui sair do nivelamento), o módulo de níveis não foi realizado (nem aparece o botão) e o botão do reforço não consegui testar, pois precisava dos desenvolvimento dos níveis.

Figura 49 - Pontuação do Grupo E

Grupo E – Agile O-MaSE (3 alunos)			
Produtos	**Iteração 1 (De 17/10 a 31/10)**	**Iteração 2 (De 31/10 a 14/11)**	**Iteração 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Análise de Requisitos			
Atividade Levantamento dos Requisitos: *Preencher* Backlog do Produto	3	3	3
Atividade Levantamento dos Requisitos: *Preencher* Backlog da Iteração	3	3	3
Atividade Análise do Problema: Modelar e Refinar Metas	0	3	3
Atividade Análise do Problema: *Preencher* Plano de Release	0	3	3
Atividade Análise da Solução: Modelo de Interface com a Organização	0	3	3
Atividade Análise da Solução: Modelo de Papeis	0	3	3
Fase de Qualidade			
Casos de Testes	0	0	3
Fase de Projeto e Arquitetura			
Modelar Classes de Agentes	0	0	3
Modelar Políticas	0	0	3
Modelar Protocolos	0	0	3
Fase de Codificação			
Incremento de acordo com o Backlog da Iteração (Jogo)	0	0	0
% ENTREGA POR ITERAÇÃO	**17%**	**51%**	**86%**

Fonte: A autora.

- Grupo F (4 alunos) – *Agile* O-MaSE

  O Grupo F (Figura 50) utilizou a metodologia *Agile* O-MaSE, realizou 91% na 1a iteração, 91% na 2a iteração e 100% na 3a iteração.

  Quase completam todos os artefatos na 1ª iteração (faltando 1). O grupo se mostrou muito interessado, tiraram dúvidas sobre a metodologia, os requisitos e o jogo. O jogo foi apresentado em cada iteração, com quase total aderência aos requisitos. O jogo está completo, porém ao chegar na parte de reforço a aplicação está dando erro (não sendo possível jogar).

  O grupo utilizou duas iterações para realizar os 22 requisitos funcionais, na 2a iteração finalizaram os requisitos que ainda tinha no *backlog*, porém seria necessário utilizar a 3a iteração para dar mais qualidade ao software, na tratativa dos erros e requisitos não funcionais como: usabilidade e interface externa, para deixar o jogo mais atraente para o jogador.

Figura 50 - Pontuação do Grupo F

Grupo F – Agile O-MaSE (4 alunos)			
Produtos	**Iteração 1 (De 17/10 a 31/10)**	**Iteração 2 (De 31/10 a 14/11)**	**Iteração 3 (De 14/11 a ~~28/11~~ 05/12)**
Fase de Análise de Requisitos			
Atividade Levantamento dos Requisitos: *Preencher* Backlog do Produto	3	3	3
Atividade Levantamento dos Requisitos: *Preencher* Backlog da Iteração	3	3	3
Atividade Análise do Problema: Modelar e Refinar Metas	3	3	3
Atividade Análise do Problema: *Preencher* Plano de Release	3	3	3
Atividade Análise da Solução: Modelo de Interface com a Organização	3	3	3
Atividade Análise da Solução: Modelo de Papeis	3	3	3
Fase de Qualidade			
Casos de Testes	1	3	3
Fase de Projeto e Arquitetura			
Modelar Classes de Agentes	3	3	3
Modelar Políticas	0	0	3
Modelar Protocolos	3	3	3
Fase de Codificação			
Incremento de acordo com o Backlog da Iteração (Jogo)	5	5	5
% ENTREGA POR ITERAÇÃO	**91%**	**91%**	**100%**

Fonte: A autora.

O grupo que foi sorteado com a metodologia *Agile* PASSI, tinha menos um integrante, mas conseguiu desenvolver o jogo mais completo e com menos bugs, provavelmente por ser uma metodologia que tem menos modelos para serem construídos, mas ainda sim uma documentação mínima necessária, para que a aplicação fosse desenvolvida conforme pedido.

O grupo que foi sorteado com a metodologia Ingenias Scrum, teve uma semana a mais para iniciar o trabalho. Mas provavelmente pela falta de experiência dos alunos em metodologia ágil ou afinidade entre eles ou até mesmo interesse na execução do trabalho, eles não tiveram um bom resultado.

Os grupos que pegaram a proposta: *Agile* O-MaSE tiveram que se esforçar mais, pois o tempo era o mesmo da metodologia *Agile* PASSI, porém com mais modelos para serem entregues, como a metodologia Ingenias Scrum. Em comparação com as metodologias do estudo de caso, a proposta deve ser readequada na documentação e para ficar mais leve a sua condução.

Como todo processo ágil, o empenho da equipe faz a diferença na completude do projeto com qualidade. O trabalho em todas as metodologias do estudo de caso mostrou que a 1a iteração é a que exige mais esforço, pois é o momento de criar todos os modelos das entregas e iniciar o software. Com o passar das iterações, esses produtos criados precisaram apenas de atualizações, até a entrega final de todos os requisitos do *backlog*.

É interessante a colocação de Schwaber e Sutherland (2017) na Seção 1.1.1 deste trabalho, que fala que o Scrum é difícil de dominar, que leva a entender que demanda muita disciplina para manter o *framework* durante o projeto e assim, obter um bom incremento de produto. Provavelmente por ter uma rotina de cerimônias durante a execução do *framework*, o ideal é que a documentação esteja adequada a proposta de um *framework* ágil.

O estudo também mostrou que os grupos que se destacaram, são compostos por

alunos que já iniciaram as suas experiências profissionais e todos possuem conhecimento em metodologia ágil. Os jogos apresentados foram mais completos e esses grupos possuíam maior habilidade para construir os modelos das metodologias.

É importante colocar que no estudo de caso, os requisitos já estavam definidos, separados por tema e priorizados, facilitando a execução das metodologias. Porém, vale ressaltar que a agilidade não é a "bala de prata", caso a equipe não tenha o controle das atividades de definição produto, cerimônias, qualidade do produto, correções dos *bugs* muito bem registradas e executadas.

Em relação aos agentes os grupos A, C e F identificaram 3 agentes enquanto o grupo E 4 agentes. Na modelagem dos agentes somente o grupo C identificou agentes que mais se assemelham a módulos, mas os papéis são similares aos identificados pelos outros grupos E e F. Entretanto não há muita diferença na quantidade de papeis sendo em torno de 4 e 5.

A metodologia *Agile* PASSI não identifica os papéis e somente as responsabilidades de cada agente, e o grupo A e F foram os que se destacaram nas modelagens dos agentes, refletindo num melhor produto final.

No estudo das metodologias O-MaSE, *Agile* PASSI e Scrum realizado por Ferreira (2015), o método O-MaSE teve um melhor resultado no atendimento dos requisitos e o processo SCRUM foi o que melhor entregou um produto final funcionando, mas com uma modelagem limitada.

Em Novo (2018) que estudou as metodologias Ingenias, *Agile* PASSI e Ingenias Scrum os grupos tiveram uma diversidade na avaliação nos que utilizaram o mesmo método. Em relação ao método *Agile* PASSI sua aplicação apresentou uma dependência maior nos fatores do grupo e o método ágil Ingenias Scrum se mostrou capaz de entregar um produto final que atendia às necessidades do solicitante, independente do grupo pelo apoio nos meta-modelos do método.

O estudo de caso mostrou que a proposta: *Agile* O-MaSE se destacou em comparação ao estudo de Ferreira (2015) com a metodologia O-MaSE. Na versão ágil tanto a modelagem quanto o jogo desenvolvido, foram bem sucedidos no estudo de caso.

4.5 Ameaças à Validade do Estudo

No decorrer do Estudo de Caso, algumas ameaças à validade do estudo foram identificadas, de acordo com Wohlin et al. (2012) podem ser:

- Validade Interna: - Os grupos não estavam distribuídos igualitariamente na quantidade de integrantes; - Alguns grupos demostraram mais interesse no desenvolvimento do estudo de caso; - Alguns alunos trabalhavam ou estagiavam, deixando de frequentar algumas aulas; - Tempo hábil para realizar as cerimônias do *framework*:

agile O-MaSE; - Atraso de alguns alunos no horário inicial da aula; - Responder errado o questionário de caracterização.

- Validade externa: - Os produtos não foram apresentados a um cliente da área da saúde, nem a especialistas em desenvolvimento de produtos de software para a área da saúde.

CONCLUSÕES E CONSIDERAÇÕES FINAIS

Este trabalho teve como principal objetivo uma proposta de adaptação de uma metodologia orientada a agente em um *framework* ágil, fundamentada no Manifesto Ágil e nas boas práticas de *frameworks* ágeis: Scrum e XP.

Realizada a experiência do estudo de caso, que utilizou metodologias ágeis e orientadas a agentes, verificou-se que o processo ágil adaptado foi bem utilizado nas entregas e no entendimento geral da agilidade, porém devido às ameaças a validade do estudo: - Alguns alunos trabalhavam ou estagiavam, deixando de frequentar algumas aulas; - Tempo hábil para realizar as cerimônias do *framework*: *agile* O-MaSE, não foi possível executar o *framework* na íntegra, faltando tempo hábil para realização da prática da reunião de retrospectiva da iteração (sprint *retrospective*), que seria realizada após as reuniões de entregas (sprint *review*) ao final de cada iteração/sprint, sendo assim considerado um fator limitante na utilização do *framework*.

Os fragmentos adicionados a metodologia O-MaSE: controle das iterações ágeis, modelo adaptativo, cerimônias de entrega e retrospectiva da iteração e a inclusão da fase de Qualidade, são elementos que possibilitam a apresentação de um *framework* ágil, que possa ser integrado a qualquer método orientado a agente.

Embora os fragmentos: reunião de revisão, retrospectiva da iteração, e a fase de Qualidade, não estejam inseridas em todos os *frameworks* ágeis existentes, esses são importantes para manter a transparência das entregas de produto ao cliente, a aproximação dos integrantes da equipe de projeto, a disciplina para obter um bom incremento do produto, inspecionando e adaptando o processo a qualidade das entregas.

O estudo permitiu identificar a importância da integração dos fragmentos retirados de propostas ágeis já existentes, que definem a sugestão do *framework Agile* O-MaSE. Outros estudos poderão ser realizados para aprimorar a proposta, assim como contribuir com mais evidências na construção de uma proposta genérica de *framework* ágil, que possa ser integrada a métodos orientados a agentes.

O trabalho teve como benefício a prática do *framework Agile* O-MaSE como metodologia ágil, respeitando a utilização de seus artefatos ágeis, criados para o controle e política de entregas das iterações do projeto.

Como trabalhos futuros, consideramos que a proposta *Agile* O-MaSE poderá aprimorada para ser aplicada em outros estudos, levando em consideração a seção 4.5 - Ameaças à Validação do Estudo, para comparação com o estudo realizado neste trabalho.

REFERÊNCIAS

AL-AZAWI, R.; AYESH, A.; OBAIDY, M. A. Towards agent-based agile approach for game development methodology. in: 2014 world congress on computer applications and information systems (wccais). ieee. In: . [S.l.: s.n.], 2014.

BABAR, M. et al. The evaluation of agile demand response: An applied methodology. ieee transactions on smart grid, v. 9, n. 6, p. 6118-6127. In: . [S.l.: s.n.], 2017.

BECK, K. Test driven development: By example. boston: Addison-wesley. In: . [S.l.: s.n.], 2002.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. Online. Disponível em: <https://agilemanifesto.org>. Acessado em 27/11/2019.

CHAVES, J. T. F.; FREITAS, S. A. A. de. Natvi-a framework for agile software development, service-oriented architecture and quality assurance. In: SPRINGER. *International Conference on Computational Science and Its Applications*. [S.l.], 2020. p. 442–458. Available at <https://doi.org/10.1007/978-3-030-58817-5_33>.

CHELLA, A.; COSSENTINO, M.; SABATUCCI, L. Designing jade systems with the support of case tools and patterns. In: EXP JOURNAL. [S.l.], 2003.

CHELLA, A. et al. Agile passi: An agile process for designing agents. in: Computer systems science and engineering. In: COMPUTER SYSTEMS SCIENCE AND ENGINEERING. [S.l.], 2006.

COHEN, D.; LINDVALL, M.; COSTA, P. Agile software development, fraunhofer center for experimental software engineering. In: . [S.l.: s.n.], 2004.

COSSENTINO, M. Design process documentation template. In: IEEE-FIPA. [S.l.], 2012. Available at: <http://fipa.org/specs/fipa00097/SC00097B.pdf>.

DALKEY, N.; HELMER, O. An experimental application of the delphi method to the use of experts. In: MANAG. SCI. [S.l.], 1963. p. 458–467.

DELOACH, A. S.; GARCIA-OJEDA, C. The o-mase methodology. in: Cossentino, m. et al. handbook on agent-oriented design processes. In: . [S.l.: s.n.], 2014.

DELOACH, S.; MILLER, M. A goal model for adaptive complex systems. int. j. comput. intell. theory pract. 5. In: . [S.l.: s.n.], 2010. p. 83–92.

FERREIRA, V. Comparação de desenvolvimento orientado a agentes para jogos educacionais: Um estudo de caso. In: INSTITUTO DE MATEMáTICA E ESTATíSTICA, UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO, RIO DE JANEIRO. BRAZIL. [S.l.]: Dissertação (Mestrado em Ciências Computacionais), 2015.

FERREIRA, V. et al. Developing an educational medical game using agilepassi multi-agent methodology. In: IEEE 28TH INTERNATIONAL SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS. [S.l.], 2015. p. 298–303. Available at: <https://ieeexplore.ieee.org/document/7167504>.

GONZáLEZ-MORENO, J. C. et al. Ingenias-scrum. in: Cossentino, m. et al. handbook on agent-oriented design processes. springer. In: . [S.l.: s.n.], 2014. p. 219–51.

GURUSAMY, K.; SRINIVASARAGHAVAN, N.; ADIKARI, S. An integrated framework for design thinking and agile methods for digital transformation. In: SPRINGER. *International Conference of Design, User Experience, and Usability*. [S.l.], 2016. p. 34–42. Available at <https://doi.org/10.1007/978-3-319-40409-7_4>.

HIGHSMITH, J. *History: The Agile Manifesto*. 2001. Online. Disponível em: <https://agilemanifesto.org/history.html>. Acessado em 27/11/2019.

JEFFRIES, R.; MELNIK, G. Tdd: The art of fearless programming. ieee software, v. 24. In: . [S.l.: s.n.], 2007. p. p. 24–30.

JENNINGS, N. R.; SYCARA, K.; WOOLDRIDGE, M. J. A roadmap of agent research and development. journal of autonomous agents and multi-agent systems, 1(1). In: . [S.l.: s.n.], 1998. p. 7–38.

KITCHENHAM, B. Guidelines for performing systematic literature reviews in software engineering", in: Ebse technical report ebse-2007-01. In: IEEE POWER AND ENERGY MAGAZINE. Online: IEEE, 2007.

KITCHENHAM, B.; PICKARD, L.; PFLEEGER, S. Case studies for method and tool evaluation. In: IEEE SOFTW. [S.l.]: ACM Sigsoft, 1995.

LETHBRIDGE, T. C.; SIM, S. E.; SINGER, J. Studying software engineers: data collection techniques for software field studies. empir. softw. eng. 10. In: . [S.l.: s.n.], 2005. p. 311–341.

MAES, P. Artificial life meets entertainment: Lifelike autonomous agents. communications of the acm, vol. 38, no. 11. In: . [S.l.: s.n.], 1995. p. p. 108–114.

MARQUES, J. C. *MACRE-SAR*: Um modelo Ágil para especificaÇÃo de requisitos de software em ambientes regulados. 204 p. Tese — Instituto Tecnológico da Aeronáutica (ITA), São José dos Campos, SP, 2016.

MASCARDI, V.; WEYNS, D. Engineering multi-agent systems anno 2025. In: . [S.l.: s.n.], 2019.

NOORI, F.; KAZEMIFARD, M. Simulation of pair programming using multi-agent and mbti personality model.in: 2015 sixth international conference of cognitive science (iccs). ieee. In: . [S.l.: s.n.], 2015. p. p. 29–36.

NOVO, F. Aplicação de métodos orientados a agentes para jogos educacionais: um estudo de caso. In: INSTITUTO DE MATEMáTICA E ESTATíSTICA, UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO, RIO DE JANEIRO. BRASIL. [S.l.]: Dissertação (Mestrado em Ciências Computacionais), 2018.

NOVO, F. et al. Modelando requisitos de um jogo educacional médico usando a metodologia ingenias scrum. In: WER-WORKSHOP OF REQUIREMENTS ENGINEERING. Online, 2018. Available at: <http://wer.inf.puc-rio.br/~wer/WERpapers/artigos/artigos_WER18/WER_2018_paper_22.pdf>.

OKOYE, C.; ADETOKUNBO, M. J.; OJIEABU, E. C. Multi-agent based software engineering models: A review. *International Journal of Software Engineering & Applications*, v. 8, n. 2, p. 139–146, 2017. Available at <https://aircconline.com/ijsea/V8N2/8217ijsea09.pdf>.

OWEN, S.; BRERETON, P.; BUDGEN, D. Protocol analysis: a neglected practice. In: MANAG. SCI. [S.l.], 2006. p. 458–467.

PFLEEGER, S. Experimental design and analysis in software engineering. In: . [S.l.]: ACM Sigsoft, 1994.

PRESSMAN, R. *Engenharia de Software: a practitioner's approach*. 7. ed. New York: MCGrawHill, 2010.

QURESHI, M. R. J.; KASHIF, M. Adaptive framework to manage multiple teams using agile methodologies. *International Journal of Modern Education and Computer Science*, Modern Education and Computer Science Press, v. 9, n. 1, p. 52, 2017. Available at <https://doi.org/10.5815/ijmecs.2017.01.06>.

ROBSON, C. Real world research: A resource for social scientists and practitioners-researchers. In: BLACKWELL. [S.l.]: Oxford/Madden, 2002.

RUNESON, P. et al. Case study research in software engineering. guidelines and examples. In: WILEY. [S.l.], 2012.

RUNESON, P.; SKOGLUND, M. Reference-based search strategies in systematic reviews. In: PROCEEDINGS OF THE 13TH INTERNATIONAL CONFERENCE ON EMPIRICAL ASSESSMENT AND EVALUATION IN SOFTWARE ENGINEERING. ELECTRONIC WORKSHOPS IN COMPUTING (EWIC). BCS, DURHAM UNIVERSITY, UK. Fez, 2009.

RUSSELL, S. J.; NORVING, P. Artificial intelligence: A modern approach. In: . Upper Saddle River: Prentice Hall, 2002.

SCHWABER, K. Scrum development process. In: *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*. [S.l.: s.n.], 1995. p. 117–134.

SCHWABER, K.; SUTHERLAND, J. *Guia do Scrum*: Um guia definitivo para scrum: As regras do jogo. [S.l.: s.n.], 2017. Online. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf>. Acessado em 27/11/2019.

SHEHORY, O.; STURM, A. Agent-oriented software engineering: Reflections on architectures, methodologies, languages, and frameworks. In: APPLIED ENERGY. [S.l.]: Springer-Verlag Berlin Heidelberg, 2014.

SILVA, C. H. S. da. *Gerenciamento ágil de projetos*: Formação agile professional. [S.l.]: ProjectLab, 2021. Online.

SOMMERVILLE, I. *Engenharia de Software*. 10. ed. São Paulo: Pearson, 2019.

STAKE, R. The art of case study research. In: THOUSAND OAKS. [S.l.]: SAGE, 1995.

STEGHOFER, J.-P. et al. Posomas: An extensible, modular se process for open self-organising systems. In: DAM, H. K. et al. (Ed.). *PRIMA 2014: Principles and Practice of Multi-Agent Systems*. Cham: Springer International Publishing, 2014. p. 1–17. ISBN 978-3-319-13191-7.

TAKEUCHI, H.; NONAKA, I. The new new product development game. *Harvard Business Review magazine*, jan 1986.

TENSO, T. et al. Enhancing requirements engineering in agile methodologies by agent-oriented goal models: Two empirical case studies. In: IEEE 25TH INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE WORKSHOPS (REW). [S.l.], 2017.

VERNER, J. et al. Guidelines for industrially-based multiple case studies in software engineering. In: THIRD INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE. Fez, 2009.

WANG, Z. Teamworking strategies of scrum team: a multi-agent based simulation. in: Proceedings of the 2018 2nd international conference on computer science and artificial intelligence. In: . [S.l.: s.n.], 2018. p. p. 404–408.

WOHLIN, C. et al. Experimentation in software engineering. In: . [S.l.: s.n.], 2012.

WOOLDRIDGE, M.; JENNINGS, N. Intelligent agents: Theory and practice, the knowledge engineering review 10 (2). In: . [S.l.: s.n.], 1995. p. 115–152.

YIN, R. Case study research design and methods. In: . Beverly Hills: SAGE, 2009.

YU, H. et al. A multi-agent game for studying human decision-making. in: Proceedings of the 2014 international conference on autonomous agents and multi-agent systems. In: . [S.l.: s.n.], 2014. p. p. 1661–1662.

APÊNDICE A – Backlog do produto

BACKLOG DO PRODUTO							
ITEM	**DESCRIÇÃO**	**TEMA**	**PRIORIDADE**	**TIPO**	**ESTIMATIVA DE COMPLEXIDADE (Fibonacci - 1 a 13)**	**ITERAÇÃO**	**SITUAÇÃO**
1	Ao entrar no jogo pela primeira vez, usuário é submetido a questões para definir seu nível e as demais funcionalidades estão indisponíveis.	Nivelamento	10	Requisito			Em aberto
2	A qualquer momento o usuário pode verificar módulos disponíveis para acesso.	Nivelamento, Reforço e Níveis	130	Requisito			Em aberto
3	O usuário visualiza como habilitado apenas módulos disponíveis conforme sua evolução.	Nivelamento, Reforço e Níveis	20	Requisito			Em aberto
4	Ao selecionar o nível disponível na página principal, usuário é direcionado para a primeira questão aleatória do Nível em questão.	Níveis	160	Requisito			Em aberto
5	Os níveis mais altos têm questões com maior complexidade. 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.	Níveis	180	Requisito			Em aberto
6	Feedback ao usuário do resultado ao final do módulo (positivo ou negativo).	Níveis	200	Requisito			Em aberto
7	As questões do nivelamento, níveis e reforço são mescladas nos formatos: áudio, vídeo, imagens e texto.	Nivelamento, Reforço e Níveis	70	Requisito			Em aberto

8	Cada questão do nivelamento, de nível e reforço é composta por 3 opções de respostas (A, B e C), onde haverá uma opção correta e duas incorretas.	Nivelamento, Reforço e Níveis	30	Requisito			Em aberto
9	Ao selecionar uma opção incorreta em uma questão, usuário é informado da opção correta.	Nivelamento, Reforço e Níveis	80	Requisito			Em aberto
10	Ao selecionar uma opção incorreta em uma questão, usuário recebe um retorno com uma justificativa relatando o porquê da opção escolhida estar incorreta. Este retorno pode ser em formato de áudio, vídeo, imagens e texto.	Nivelamento, Reforço e Níveis	90	Requisito			Em aberto
11	Usuário para cada pergunta pode rever a questão, a resposta e o retorno, entretanto o usuário não consegue alterar a opção selecionada.	Nivelamento, Reforço e Níveis	100	Requisito			Em aberto
12	O nivelamento, corresponde a 10 questões referentes aos cinco níveis disponíveis. Para cada nível, existem duas questões distribuídas aleatoriamente.	Nivelamento	40	Requisito			Em aberto
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, o usuário vê um indicativo vermelho.	Nivelamento, Reforço e Níveis	50	Requisito			Em aberto
14	Ao final do nivelamento, respondendo todas as questões, usuário é informado sobre o nível alcançado: Nível 5 (acertar todas as questões do nível 1 ao 4), Nível 4 (acertar todas as questões do nível 1 ao 3), Nível 3 (acertar todas as questões do nível 1 e 2), Nível 2 (acertar todas as	Nivelamento	60	Requisito			Em aberto
15	Usuário pode interromper o nivelamento e deve iniciar o processo novamente e as respostas anteriores são desconsideradas.	Nivelamento	110	Requisito			Em aberto
16	Usuário deve realizar o Nível 5, mesmo que acerte todas as questões do nivelamento.	Nivelamento	120	Requisito			Em aberto

17	Após finalizar o nivelamento, usuário é direcionado para a página principal do jogo e poderá iniciar o nível alcançado.	Nivelamento, Reforço e Níveis	140	Requisito			Em aberto
18	Ao entrar no jogo após realização do nivelamento, usuário permanecerá no último nível alcançado.	Níveis	150	Requisito			Em aberto
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.	Níveis	170	Requisito			Em aberto
20	O usuário que errar mais de 50% em qualquer um dos níveis deverá fazer o reforço.	Níveis e Reforço	210	Requisito			Em aberto
21	Ao final de cada nível, usuário que obter 70% de acertos ou mais, está apto a avançar para o próximo nível.	Níveis	190	Requisito			Em aberto
22	Usuário obtém ao final de cada nível um equipamento para o seu consultório.	Níveis	220	Requisito			Em aberto

APÊNDICE B – Backlog da iteração

BACKLOG DA ITERAÇÃO						
META DA ITERAÇÃO 1	ITEM	HISTÓRIA DE USUÁRIO	TAREFAS DA HISTÓRIA	DURAÇÃO DAS TAREFAS	DISTRIBUIÇÃO DAS TAREFAS	SITUAÇÃO DA TAREFA

APÊNDICE C – Plano de release

Plano de Release da metodologia *Agile O-MaSE* – Grupos E e F
Projeto de Desenvolvimento do Jogo Médico Educacional

Histórico de Revisões

Data	Versão	Descrição	Autor
17/10/2019	1.0	Criação do documento e preenchimento das informações: 1, 2, 3 e 4.	Fabiana Gominho

1. **Definição do Critério de Sucesso da *Release***

   Este projeto é orientado por data, e deve ser finalizado no dia 05/12/19. Consequentemente, as entregas das Iterações são do tipo *End-Date Driven,* ou seja, orientada para data de termino, cujas as entregas devem estar disponíveis antes da data limite.

   Os critérios para considerar uma release terminado com sucesso são:

   - A entrega de todas as histórias do *backlog da Iteração;*
   - A qualidade da entrega: requisitos corretos e sem bugs e
   - A apresentação dos modelos e artefatos que a metodologia requer.

   O projeto possui um Comitê de Qualidade, composto por: Fabiana Gominho, Vera Werneck e Rosa Costa, que fará a aceitação das entregas de cada iteração.

2. **Estimativa dos Itens de *Backlog* do Produto**

   A estimativa da complexidade de desenvolvimento deve ser analisada pelo Time de desenvolvimento, item por item do *Backlog* e deve ser iniciado pelo item de maior prioridade. A técnica utilizada na estimativa dos itens do *backlog*, deve ser a sequência de número *Fibonacci*: 1, 2, 3, 5, 8 e 13. Quanto maior for o número, maior será a complexidade de desenvolvimento.

3. **Definição do Tamanho e Quantidade de Iterações**

   O tamanho da Iteração está definido limitado a data final do projeto, num *time-box* de 2 semanas, dando um total de 3 iterações.

4. **Estimativa da Velocidade**

   A estimativa da velocidade será realizada de forma manual, observando a primeira iteração e ajustando as demais iterações.

5. **Associação dos Itens de *Backlog* do Produto as Iterações**

Iteração	Backlog do Produto

Iteração	Backlog do Produto

APÊNDICE D – Acompanhamento do estudo de caso

Aplicação de Métodos Ágeis Orientados a Agentes:
Agile Passi, Ingenias-Scrum* e *Agile O-MaSE
Jogo Médico Educacional

Orientadora: Vera Werneck
Coorientadora: Rosa Costa
Mestranda: Fabiana Gominho – fgominho@gmail.com

Outubro/2019

Roteiro da Apresentação

- Objetivo do projeto
- Nosso projeto
- Aulas
- Agentes
- Manifesto Ágil
- Modelo Iterativo e Incremental
- Conceitos de agilidade
- Práticas de agilidade
- Metodologias: Agile Passi, Ingenias Scrum e Agile O-MaSE
- Cronogramas
- O jogo médico educacional – MedEduc

Objetivo do Projeto

O objetivo do nosso projeto é desenvolver o mesmo jogo médico educacional, utilizando 3 diferentes metodologias ágeis orientadas a agentes: ***Agile Passi, Ingenias-Scrum e Agile O-MaSE.***

Nosso Projeto – Aula 10/10/19

Divisão dos Grupos:

- 1 representante por grupo
- ~~5 grupos de 3 alunos~~ → 2 grupos de 3 alunos
- ~~1 grupo de 4 alunos~~ → ~~1 grupo de 2 alunos~~ → 2 grupos de 4 alunos

Sorteio dos grupos:

- Grupos **A ~~e B~~** → Metodologia ***Agile Passi***
- Grupos **C ~~e D~~** → Metodologia ***Ingenias-Scrum***
- Grupos **E e F** → Metodologia ***Agile O-MaSE***

Nosso Projeto – Aula 10/10/19

Grupo A – *Agile Passi*:

- VE (Representante)
- RS
- PL → ausente na aula.

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS

Nosso Projeto – Aula 10/10/19

Atividades:

- Preenchimento do formulário de termo de consentimento;
- Responder o questionário de caracterização do participante;
- Nossos encontros serão às 5as feiras;
- Seguir o cronograma.

Teremos 3 marcos de entregas:

- **31/10** – 1ª entrega**, 14/11** – 2ª entrega e **28/11** – 3ª e última entrega;
- Disponibilizar as entregas no Goolge Drive;
- As entregas contarão para avaliação com peso 1;
- Entregar o software funcionando + Modelos/Artefatos da metodologia equivale a **70% da nota da P2**.

Aula 17/10/19

Grupo A – *Agile Passi*:

- VE (Representante) → Presente na aula.
- RS → Presente na aula.
- PL → Presente na aula.

Grupo C – *Ingenias-Scrum*:

- LC (Representante) → Presente na aula.
- AS → Presente na aula.
- LR → Presente na aula.
- ES → Presente na aula.

Grupo E – *Agile O-MaSE*:

- DC (Representante) → Presente na aula.
- WS → Presente na aula.
- PL → Presente na aula.

Grupo F – *Agile O-MaSE*:

- DJ (Representante) → Presente na aula.
- VA → Ausente na aula.

Aula 17/10/19

1ª Iteração/Sprint: 17/10/19 a 31/10/19 (apresentação e entrega):

- **Reapresentação da aula anterior;**
- **Meta/Objetivo da 1ª Iteração** → Poder realizar as 10 questões do Nivelamento, responder e ter resposta de questões variadas utilizando texto, imagem, áudio ou vídeo, podendo rever as questões já respondidas, saber das respostas corretas quando estiverem erradas, poder parar e reiniciar o nivelamento e no final receber a orientação de qual nível o usuário deve seguir no jogo.
- **Requisitos envolvidos apenas com o nivelamento → *Em amarelo*;**
- **Requisitos envolvidos com o nivelamento, reforço e níveis → *Em verde*;**
- **Requisitos em branco não correspondem a esta Iteração;**
- **Verificar os requisitos acima, pela coluna TEMA nos artefatos enviados.**

Aula 17/10/19

Item	Requisitos	Prioridade
1	Ao entrar no jogo pela primeira vez, usuário é submetido a questões para definir seu nível e as demais funcionalidades estão indisponíveis.	10
2	A qualquer momento o usuário pode verificar módulos disponíveis para acesso.	130
3	O usuário visualiza como habilitado apenas módulos disponíveis conforme sua evolução.	20
4	Ao selecionar o nível disponível na página principal, usuário é direcionado para a primeira questão aleatória do Nível em questão.	160
5	Os níveis mais altos têm questões com maior complexidade. 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.	180
6	Feedback ao usuário do resultado ao final do módulo (positivo ou negativo).	200
7	As questões do nivelamento, níveis e reforço são mescladas nos formatos: áudio, vídeo, imagens e texto.	70
8	Cada questão do nivelamento, de nível e reforço é composta por 3 opções de respostas (A, B e C), onde haverá uma opção correta e duas incorretas.	30
9	Ao selecionar uma opção incorreta em uma questão, usuário é informado da opção correta.	80
10	Ao selecionar uma opção incorreta em uma questão, usuário recebe um retorno com uma justificativa relatando o porquê da opção escolhida estar incorreta. Este retorno pode ser em formato de áudio, vídeo, imagens e texto.	90

Aula 17/10/19

Item	Requisitos	Prioridade
11	Usuário para cada pergunta pode rever a questão, a resposta e o retorno, entretanto o usuário não consegue alterar a opção selecionada.	100
12	O nivelamento, corresponde a 10 questões referentes aos cinco níveis disponíveis. Para cada nível, existem duas questões distribuídas aleatoriamente.	40
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, o usuário vê um indicativo vermelho.	50
14	Ao final do nivelamento, respondendo todas as questões, usuário é informado sobre o nível alcançado: Nível 5 (acertar todas as questões do nível 1 ao 4), Nível 4 (acertar todas as questões do nível 1 ao 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).	60
15	Usuário pode interromper o nivelamento e deve iniciar o processo novamente e as respostas anteriores são desconsideradas.	110
16	Usuário deve realizar o Nível 5, mesmo que acerte todas as questões do nivelamento.	120
17	Após finalizar o nivelamento, usuário é direcionado para a página principal do jogo e poderá iniciar o nível alcançado.	140
18	Ao entrar no jogo após realização do nivelamento, usuário permanecerá no último nível alcançado.	150

Aula 17/10/19

Item	Requisitos	Prioridade
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.	170
20	O usuário que errar mais de 50% em qualquer um dos níveis deverá fazer o reforço.	210
21	Ao final de cada nível, usuário que obter 70% de acertos ou mais, está apto a avançar para o próximo nível.	190
22	Usuário obtém ao final de cada nível um equipamento para o seu consultório.	220

Aula 24/10/19

Grupo A – *Agile Passi*:

- VE (Representante)
- RS
- PL

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR → Ausente na aula.
- ES

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS → Ausente na aula.
- PL

Grupo F – *Agile O-MaSE*:

- DJ (Representante)
- VA
- GE
- GS

Aula 24/10/19

1ª Iteração/Sprint: 17/10/19 a 31/10/19 (apresentação e entrega)

- **Os grupos foram criados no Goolge Drive** – Depositar as entregas da Iteração:

- Teremos um encontro extra, na próxima aula **dia 29/10 (terça-feira)**, para tirar as dúvidas;
- **Sobre a apresentação no dia 31/10:** Apresentar o planejamento realizado seguindo o cronograma (artefatos, modelos diagramas...) e no final, o protótipo desenvolvido;
- **Duvidas?**

Aula 29/10/19

Grupo A – *Agile Passi*:

- VE (Representante)
- RS
- PL → Ausente na aula.

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR → Ausente na aula.
- ES

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS → Ausente na aula.
- PL

Grupo F – *Agile O-MaSE*:

- DJ (Representante) → Ausente na aula.
- VA → Ausente na aula.
- GE
- GS → Ausente na aula.

Aula 29/10/19

1ª Iteração/Sprint: 17/10/19 a 31/10/19 (apresentação e entrega)

- **Aula para tirar as dúvidas:**

- **Grupo A → Tirou dúvidas** 
- **Grupo C**
- **Grupo E**
- **Grupo F → Tirou dúvidas** 

Aula 31/10/19 – Entrega da Iteração/Sprint 1

Grupo A – *Agile Passi*:

- VE (Representante)
- RS
- PL

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR → Ausente na apresentação.
- ES → Ausente na apresentação (Justificado).

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS
- PL

Grupo F – *Agile O-MaSE*:

- DJ (Representante)
- VA
- GE
- GS

Aula 31/10/19 – Entrega da Iteração/Sprint 1

- Apresentação e entrega da Iteração/Sprint: 1;
- **2ª Iteração/Sprint: 31/10/19 a 14/11/19 (apresentação e entrega):**
- **Meta/Objetivo →** Poder realizar as 10 questões dos **Níveis 1 ao 3** do jogo, e realizar as 10 questões do **Reforço**, caso seja necessário e de acordo com o feedback dos níveis, caso não seja necessário o usuário deve receber a premiação. Responder e ter resposta de questões variadas utilizando texto, imagem, áudio ou vídeo, podendo rever as questões já respondidas, saber das respostas corretas quando estiverem erradas, poder parar e reiniciar os Níveis e o Reforço.

Aula 07/11/19

Grupo A – *Agile Passi*:
- VE (Representante)
- RS
- PL → Ausente na aula

Grupo C – *Ingenias-Scrum*:
- LC (Representante)
- AS
- LR
- ES

Grupo E – *Agile O-MaSE*:
- DC (Representante)
- WS → Ausente na aula
- PL
- LL → Iniciou no grupo

Grupo F – *Agile O-MaSE*: → Ausente na aula
- DJ (Representante)
- VA
- GE
- GS

Aula 07/11/19

2ª Iteração/Sprint: 31/10/19 a 14/11/19 (apresentação e entrega)

- **Aula para tirar as dúvidas:**

- **Grupo A → Tirou dúvidas** 
- **Grupo C → Tirou dúvidas** 
- **Grupo E → Tirou dúvidas** 
- **Grupo F**

Aula 14/11/19 – Entrega da Iteração/Sprint 2

- **Apresentação e entrega da Iteração/Sprint: 2;**

- **3ª Iteração/Sprint: 14/11/19 a 28/11/19 (apresentação e entrega):**

- **Meta/Objetivo →** Poder realizar as 10 questões dos **Níveis 4 ao 5** do jogo, e realizar as 10 questões do **Reforço**, caso seja necessário e de acordo com o feedback dos níveis, caso não seja necessário o usuário deve receber a premiação. Responder e ter resposta de questões variadas utilizando texto, imagem, áudio ou vídeo, podendo rever as questões já respondidas, saber das respostas corretas quando estiverem erradas, poder parar e reiniciar os Níveis e o Reforço. Após responder todos os níveis exibir o consultório pronto com os 5 equipamentos.

Aula 14/11/19 – Entrega da Iteração/Sprint 2

Grupo A – *Agile Passi*:

- VE (Representante)
- RS
- PL

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR
- ES

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS
- PL
- LL

Grupo F – *Agile O-MaSE*:

- DJ (Representante)
- VA → Ausente na apresentação.
- GE
- GS

Aula 21/11/19

Grupo A – *Agile Passi*:

- VE (Representante)
- RS → Ausente na aula.
- PL

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR → Ausente na aula.
- ES → Ausente na aula

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS → Ausente na aula.
- PL
- LL

Grupo F – *Agile O-MaSE*: → Ausente na aula.

- DJ (Representante)
- VA
- GE
- GS

Aula 21/11/19

3ª Iteração/Sprint: 17/10/19 a 31/10/19 (apresentação e entrega)

- **Aula para tirar as dúvidas:**

- **Grupo A → Tirou dúvidas** 
- **Grupo C**
- **Grupo E → Tirou dúvidas** 
- **Grupo F**

Aula 28/11/19 – Entrega da Iteração/Sprint 3

- A apresentação da 3ª Iteração/Sprint seria hoje, mas foi adiada para **próxima quinta-feira 05/12**.

Aula 28/11/19

Grupo A – *Agile Passi*:

- VE (Representante)
- RS
- PL

Grupo C – *Ingenias-Scrum*:

- LC (Representante)
- AS
- LR
- ES

Grupo E – *Agile O-MaSE*:

- DC (Representante)
- WS → Ausente na aula
- PL
- LL

Grupo F – *Agile O-MaSE*:

- DJ (Representante)
- VA → Ausente na aula
- GE
- GS → Ausente na aula

Aula 28/11/19

3ª Iteração/Sprint: 31/10/19 a ~~*28/11/19*~~ 05/12/19 (apresentação e entrega)

- **Aula para tirar as dúvidas:**

- **Grupo A**
- **Grupo C → Tirou dúvidas**
- **Grupo E → Tirou dúvidas**
- **Grupo F**

APÊNDICE E – Questionário de Caracterização

Questionário de Caracterização

Avaliação do Jogo MEDEDUC	
Nome participante:	
e-mail:	

Prezado (a) participante,
Este formulário contém algumas perguntas sobre sua experiência acadêmica e profissional.

1. **Faixa etária:**
   ( ) < 25 anos
   ( ) >=25 anos

2. **Atuação no Mercado**
   ( ) Trabalha ou faz estágio na área de computação
   ( ) Trabalha ou faz estágio, mas não na área de computação
   ( ) Trabalhou ou fez estágio na área de computação
   ( ) Trabalhou ou fez estágio, mas não na área de computação
   ( ) Não trabalha

3. **Experiência em sistemas multiagentes?** ( )SIM ( ) NÃO

4. **Nível de conhecimento sobre agentes** _____
   0 (nenhum conhecimento) até 5 (especialista no assunto)

5. **Nível de conhecimento sobre sistemas multiagentes** _____
   0 (nenhum conhecimento) até 5 (especialista no assunto)

6. **Conhecimento da metodologia proposta?** ( )SIM ( ) NÃO

7. **Conhecimento de metodologia ágil?** ( )SIM ( ) NÃO

   Qual/Quais? _____

8. **Participação em desenvolvimento de jogos?** ( )SIM ( ) NÃO

9. **Participação em desenvolvimento de soluções médicas?** ( )SIM ( ) NÃO

10. **Expectativa do trabalho proposto**

_____

_____

_____

_____

_____

APÊNDICE F – Expectativas dos alunos

"Espero contribuir com o meu conhecimento e adquirir novos conhecimentos."

"Expectativa de ampliar meu conhecimento sobre o tema proposto."

"Desenvolver o jogo usando a metodologia Agile O-MaSE."

"Aprender a utilização de agentes em jogos."

"Aprender melhor a aplicar os conhecimentos sobre agentes na metodologia ágil."

"Expectativa de grande aprendizado."

"Expectativa bem positiva, poder aplicar meus conhecimentos com colegas de turma usando uma nova metodologia."

"Adquirir experiência na modelagem de agentes e em metodologia de desenvolvimento ágil."

"Espero aprender mais sobre o framework que será utilizado e ter a experiência de utilizar o ágil com o pessoal da faculdade."

"Incrementar o aprendizado sobre agentes, aprendizado da metodologia ágil proposta e entregar algo legal."

"Não sei exatamente o que esperar devido ao baixo conhecimento sobre o assunto, o que talvez indique uma grande possibilidade de aprendizado."

"Aprender sobre metodologia ágil."

"Obter conhecimentos base para desenvolvimento ágil que possa ser aplicado em ambiente corporativo."

"Adquirir conhecimentos valiosos que eu possa aplicar no meu ambiente de trabalho e contribuir com a minha vida acadêmica e profissional."

ANEXO A – Comparação entre O-MaSE, Agile PASSI e Scrum

	O-MaSE	AgilePASSI	Scrum
Abordagem	Tradicional	Agil	Agil
Foco	Plano, processo, documentação e ferramenta	Grupo, jogo, colaboração do cliente e resposta à mudança	Grupo, jogo, colaboração do cliente e resposta à mudança
Orientação do método	Orientado a processos que precisam ser seguidos para garantia da qualidade	Orientado a pessoas e foco no desenvolvimento de agentes	Orientado a pessoas com foco na motivação e comprometimento dos envolvidos
Orientado a agentes	Sim	Sim	Não
Identificação de agentes	Baixa Dificuldade	Baixa Dificuldade	Alta Dificuldade
Fases	Concepção, elaboração e construção	Não há	Não há
Necessidade de iteração	Sim	Sim	Sim
Iterações	6 iterações	Quantas forem necessárias	Quantas forem necessárias
Modelos desenvolvidos	Modelo de metas, modelo de metas refinado, modelo de domínio, modelo de interfaces da organização, modelo de papéis, modelos de definição de papéis, modelos de classes de agentes, modelos de protocolo, modelos de políticas, modelos de planos, modelos de capacidade e modelos de ações	Modelo de requisitos do sistema, modelo de sociedade, modelo de código e modelo de teste	-
Artefatos	Diagrama de metas, diagrama de metas refinado, diagrama de domínio, diagrama de interfaces da organização, diagrama de papéis, diagrama de definição de papéis, diagrama de classes de agentes, diagrama de protocolo, diagrama de políticas, diagrama de planos, diagrama de capacidade, diagrama de ações e código	Estórias de usuários, diagramas de caso de uso UML, diagramas de classes de agentes, plano de testes e código	*Product backlog*, *sprint backlog*, *BurnDown Chart*, estórias de usuários e código
Flexibilidade	Não há	Média	Alta
Tipo de reunião	-	-	*Sprint planning*, *daily meeting*, *sprint review*, *sprint retrospective*, *FUP meeting* e *reestimate meetings*
Papéis	-	-	*Scrum Master*, *Product Owner*, e *team*
Tamanho da equipe	-	-	No máximo 15 participantes

ANEXO B – Comparação entre Ingenias, Agile PASSI e Ingenias Scrum

	INGENIAS	AgilePASSI	INGENIAS Scrum
Abordagem	Tradicional	Ágil	Ágil
Origem	MESSAGE/UML	PASSI	UML/RUP
Orientação do método	Orientado a projetos e a produção de produtos.	Orientado a pessoas e foco no desenvolvimento de agentes.	Orientado a pessoas com foco na motivação e comprometimento dos envolvidos.
Fases	Concepção, elaboração e construção.	Definição de requisitos, modelo de sociedade de agentes, plano de teste, codificação e testes.	Fase de preparação e fase de *sprints*.
Modelos desenvolvidos	Modelo de organização, modelo de agentes, modelo de metas e tarefas, modelo de ambiente e modelo de interação.	Modelo de requisitos do sistema, modelo de sociedade, modelo de código e modelo de teste.	Modelo de organização, modelo de agentes, modelo de metas e tarefas, modelo de ambiente e modelo de interação.
Artefatos	Diagrama de casos de uso, diagrama de organização, diagrama de agentes, diagrama de metas e tarefas, diagrama de ambiente, diagrama de interação e modelo de códigos.	Estórias de usuários, diagramas de caso de uso UML, diagramas de classes de agentes, plano de testes e código.	Diagrama de casos de uso, diagrama de organização, diagrama de agentes, diagrama de metas e tarefas, diagrama de ambiente, diagrama de interação, *product backlog*, *Sprint backlog*, *BurnDown Chart*, estórias de usuários e código.
Tipo de reunião	Não possui	Não possui	*Sprint planning, daily meeting, sprint review, sprint retrospective, FUP meeting* e *reestimate meetings*
Equipe	Não define	Não define	*Scrum Master, Product Owner,* e *Team Scrum.*
Tamanho da equipe	Não possui	Não possui	Máximo 15 participantes.

ANEXO C – Grupo A (Agile Passi) - Modelo de Requisitos e Sociedade de Agente

Sumário

Introdução

Este documento tem como objetivo descrever e especificar os casos de uso referentea ao software "Jogo Médico Educacional", descrevendo os atores e os casos de uso associados a estes.

Atores do Sistema

Ator	Descrição
Jogador	Um usuário humano
Perguntador	Agente responsável por selecionar as questões de acordo com o status de progresso do Jogador
Validador	Agente responsável por receber a resposta do Jogador de cada pergunta e verificar se está certa.
Processador de Resutado	Agente responsável por análisar o desempenho do Jogador na série de questões e atualizar seu status de progresso

Casos de Uso

UC1	Fazer Login
Descrição	Jogador se autentica no sistema.
Requisitos associados	[**RF02**][**RF03**]
Atores	**Jogador** (principal)
Pré-condições	Não há.
Pós-condições	Jogador terá acesso à tela inicial do jogo.
FP: Fluxo principal	1. Jogador acessa o sistema. 2. Sistema solicita seu nome de usuário e senha. 3. Jogador informa nome de usuário e senha. 4. Jogador seleciona a opção de fazer login. [**FA1**] 5. Sistema valida nome de usuário e senha. 6. Caso o nome de usuário ou senha estejam incorretos, voltar ao passo **2**. Caso contrário, avançar para o passo **7**; 7. Sistema concede acessso à tela inicial. 8. O caso de uso é encerrado.
FA1: Cadastrar Usuário	1. No passo **4** do fluxo [**FP**], o Jogador decide selecionar a opção de cadastrar usuário. 2. Caso o nome e a senha já exista, o controle é repassado ao passo **2** do fluxo [**FP**]. 3. O controle e repassado ao passo **7** do fluxo [**FP**].

UC2	Fazer Nível
Descrição	Jogador responde a uma lista de dez questões referentes ao seu nível atual.
Requisitos associados	[**RF04**][**RF05**][**RF06**][**RF07**][**RF08**][**RF09**][**RF10**][**RF11**] [**RF13**][**RF18**][**RF19**][**RF20**][**RF21**]
Atores	**Jogador** (principal), **Perguntador**,**Validador**, **Processador de Resultado**(secundário)
Pré-condições	Jogador deve ter feito login no sistema e ter concluído o nivelamento.
Pós-condições	Caso acerte no mínimo 70% das questões, estará apto a realizar o nível seguinte.
FP: Fluxo principal	1. Jogador seleciona a opção de fazer o nível atual. 2. Perguntador carrega uma lista de dez questões aleatórias referentes ao nível atual. 3. Perguntador escolhe uma questão aleatória da lista e exibe para o Jogador. [**UC5**] 4. Jogador escolhe uma alternativa e confirma a escolha.[**FA1**][**FA2**] 5. Validador processa a resposta e informa ao Jogador se acertou (exibindo a resposta certa em caso de erro). [**UC6**] 6. Se for a décima questão, ir para o passo **7**. Caso contrário, voltar ao passo **3**. 7. Processador de Resultado faz a análise do desempenho do jogador.[**UC7**] 8. O caso de uso é encerrado.
FA1: Jogador Reinicia Série	1. No passo **4** do fluxo [**FP**], o Jogador decide reiniciar a série de questões. 2. Todo o progresso do Jogador na série atual é desconsiderado. 3. O controle e repassado ao passo **1** do fluxo [**FP**].
FA2: Jogador Desiste de Responder as Questões	1. No passo **4** do fluxo [**FP**], o Jogador desiste de reponder as questões e decide encerrar o jogo. 2. Todo o progresso do Jogador na lista de questões atual é

	desconsiderado. 3. O caso de uso é encerrado.

UC3	Fazer Nivelamento
Descrição	Jogador responde a uma lista de dez questões referentes aos cinco níveis para determinar em qual nível será inserido.
Requisitos associados	[**RF01**][**RF02**][**RF03**][**RF11**][**RF12**][**RF13**][**RF14**][**RF15**][**RF17**]
Atores	**Jogador** (principal), **Perguntador**,**Validador**, **Processador de Resultado**(secundário)
Pré-condições	Jogador deve estar jogando pela primeira vez e ter feito login no sistema.
Pós-condições	Jogador estará apto a responder às questões do nível a ser determinado no término do nivelamento.
FP: Fluxo principal	1. O caso de uso inicia sua execução a partir da execução do fluxo principal do [**UC2 - Fazer Nível**] , do passo **1** ao passo **2** (inclusive). 2. Jogador seleciona a opção de fazer nivelamento. 3. Perguntador carrega uma lista de dez questões, contendo duas questões aleatórias de cada nível. 4. O caso de uso é encerrado.

UC4	Fazer Reforço
Descrição	Jogador responde a uma lista de dez questões referentes ao seu nível atual após não ter sido aprovado.
Requisitos associados	[**RF07**][**RF08**][**RF09**][**RF10**][**RF11**][**RF13**]

UC4	Fazer Reforço
Atores	**Jogador** (principal), **Perguntador**, **Validador**(secundário)
Pré-condições	Jogador precisa ter acertado menos de 70% das questões do seu nível atual.
Pós-condições	Jogador realizará novamente o seu nível atual.
FP: Fluxo principal	1. O caso de uso inicia sua execução a partir da execução do fluxo principal do [**UC2 - Fazer Nível**] , substituindo o passo **7**. 2. O resultado não será analisado pelo Processador de Resultado. 3. O caso de uso é encerrado.

UC5	Disponibilizar Questão
Descrição	Perguntador escolhe uma questão aleatória da sua lista.
Requisitos associados	[**RF07**][**RF08**][**RF12**]
Atores	**Perguntador** (principal), **Validador**(secundário)
Pré-condições	Perguntador deve ter um lista de questões carregadas.
Pós-condições	Uma questão deve ser exibida ao Jogador e enviada ao Validador.
FP: Fluxo principal	1. Perguntador seleciona uma questão aleatória da sua lista. 2. Perguntador envia a questão selecionada para o Validador, informando a alternativa correta e removendo a questão de sua lista. 3. Validador armazena a questão e sua alternativa correta. 4. Perguntador exibe o enunciado da questão e as alternativas para o Jogador. 5. O caso de uso é encerrado.

UC6	Processar Resposta
Descrição	Validador verifica se o Jogador acertou ou não questão.
Requisitos associados	[**RF09**][**RF10**]
Atores	**Validador** (principal), **Processador de Resultado** (secundário)
Pré-condições	Jogador deve ter respondido a questão.
Pós-condições	Resultado da verificação da resposta deve ser enviado ao processador de resultado e exibido ao Jogador.
FP: Fluxo principal	1. Validador recebe a resposta do jogador. 2. Validador verifica se o jogador acertou ou errou. 3. Validador envia a questão e o resultado para o Processador de Resultados. 4. Processador de Resultado armazena o resultado da questão. 5. Validador exibe a resposta certa para o Jogador e a mensagem de acerto ou erro. 6. O caso de uso é encerrado.

UC7	Processar Resultado
Descrição	Processador de Resultado analisa as respostas e informa ao jogador se estará apto a realizar o nível seguinte.
Requisitos associados	[**RF20**][**RF21**]

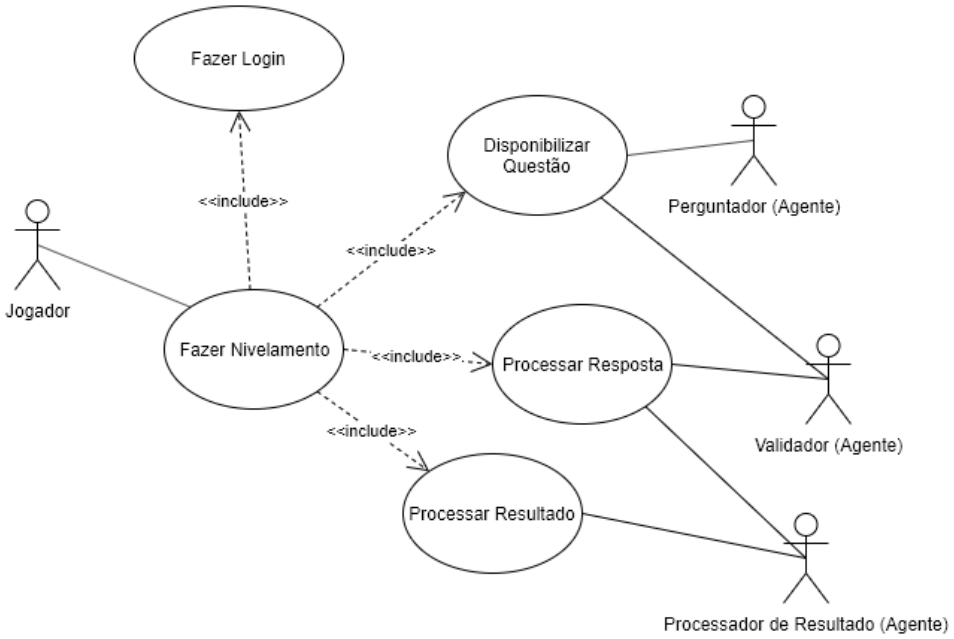
UC7	Processar Resultado
Atores	**Processador Resultado** (principal), **Jogador** (secundário)
Pré-condições	Jogador deve ter respondido a todas as questões.
Pós-condições	Será determinado se o jogador passará de nível ou não.
FP: Fluxo principal	1. Processador de Resultado verifica quantas questões o jogador acertou. 2. Caso o percentual de acertos seja igual ou superior a 70%, o Jogador passará para o nível seguinte. 3. Processador de Resultado informa ao Jogador o resultado. 4. Jogador visualiza a mensagem. 5. O caso de uso é encerrado.

UC8	Processar Resultado do Nivelamento
Descrição	Processador de Resultado analisa as respostas do Jogador e define o nível no qual ele será inserido.
Requisitos associados	[**RF14**][**RF16**][**RF17**]
Atores	**Processador Resultado** (principal), **Jogador** (secundário)
Pré-condições	Jogador deve ter respondido a todas as questões.
Pós-condições	O nível no qual o Jogador será inserido estará determinado.
FP: Fluxo principal	1. O caso de uso inicia sua execução a partir da execução do fluxo principal do [**UC7 - Processar Resultado**] , do passo **1** ao passo **2** (inclusive). 2. Processador de Resultado analisa os acertos e erros do Jogador. 3. Processador de Resultado determina o nível em que o Jogador será inserido. 4. O caso de uso é encerrado.

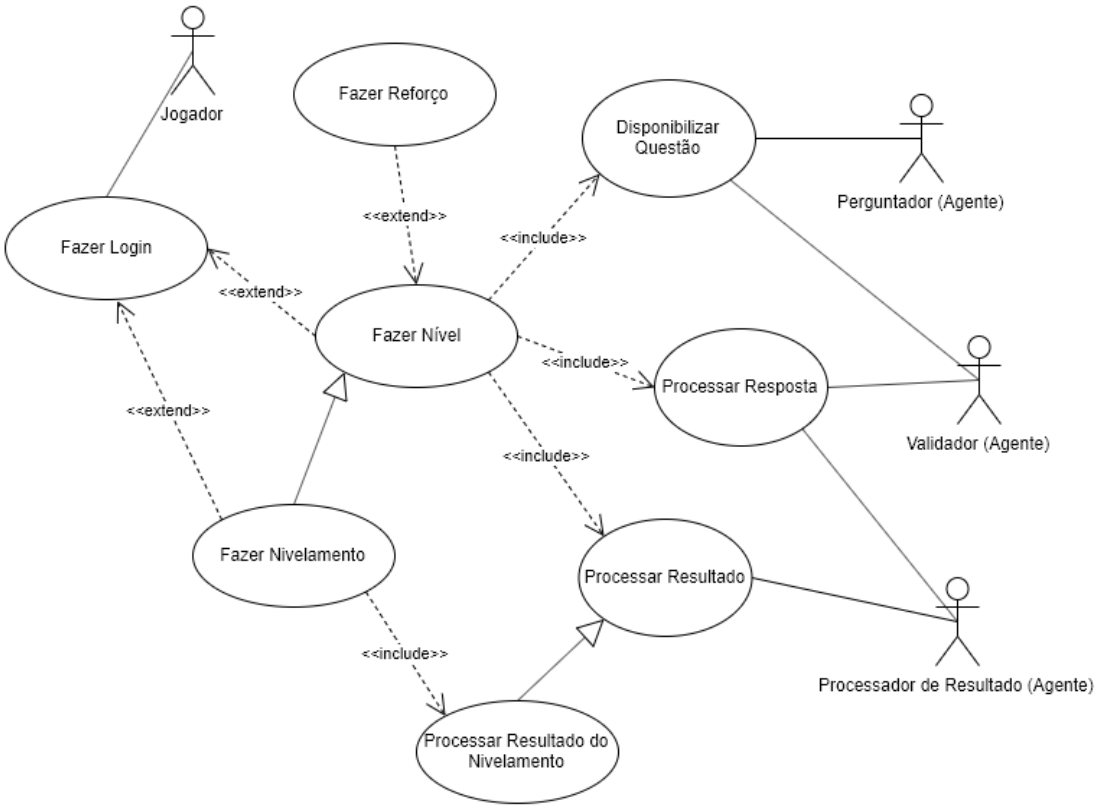
UC9	Visualizar Consultório
Descrição	Jogador visualiza o estado do seu consultório, bem como itens bloqueados e desbloqueados.
Requisitos associados	[**RF22**]
Atores	**Jogador** (principal)
Pré-condições	Jogador deve ter feito login no sistema.
Pós-condições	Não há.
FP: Fluxo principal	1. Jogador seleciona a opção de visualizar o consultório. 2. Sistema exibe o estado do consultório, mostrando os itens bloqueados e desbloqueados. 3. Jogador visualiza as itens e o consultório. 4. O caso de uso é encerrado.

Diagrama de casos de uso (UML)

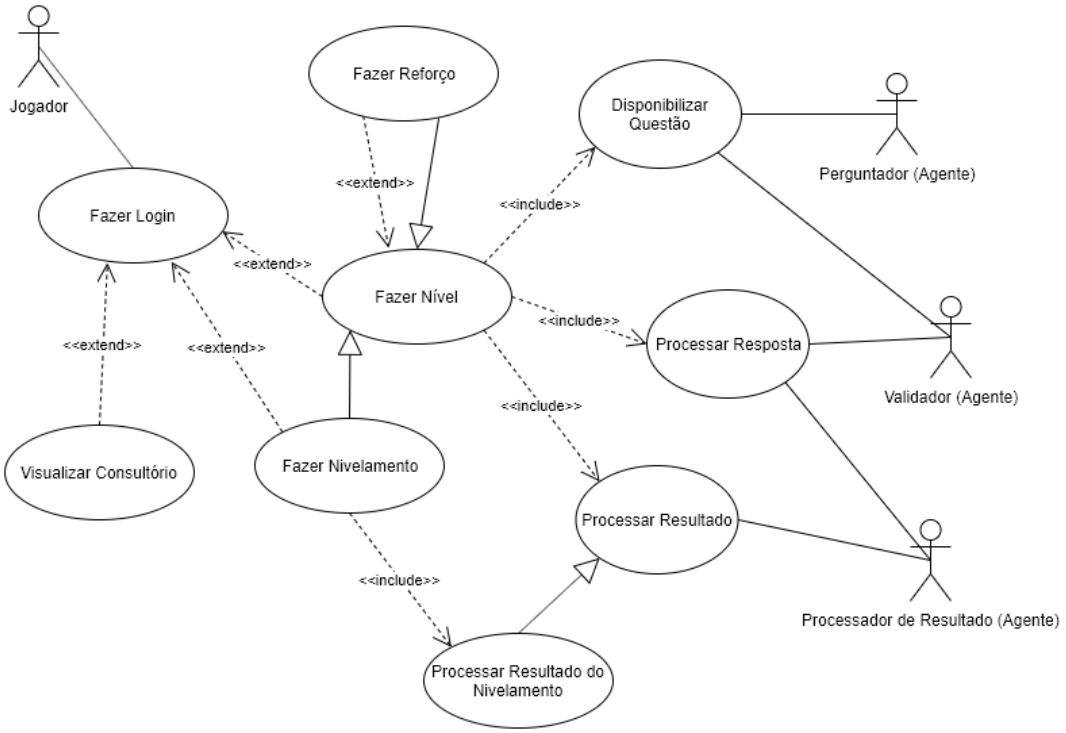
- 1ª Iteração:



- 2ª Iteração:

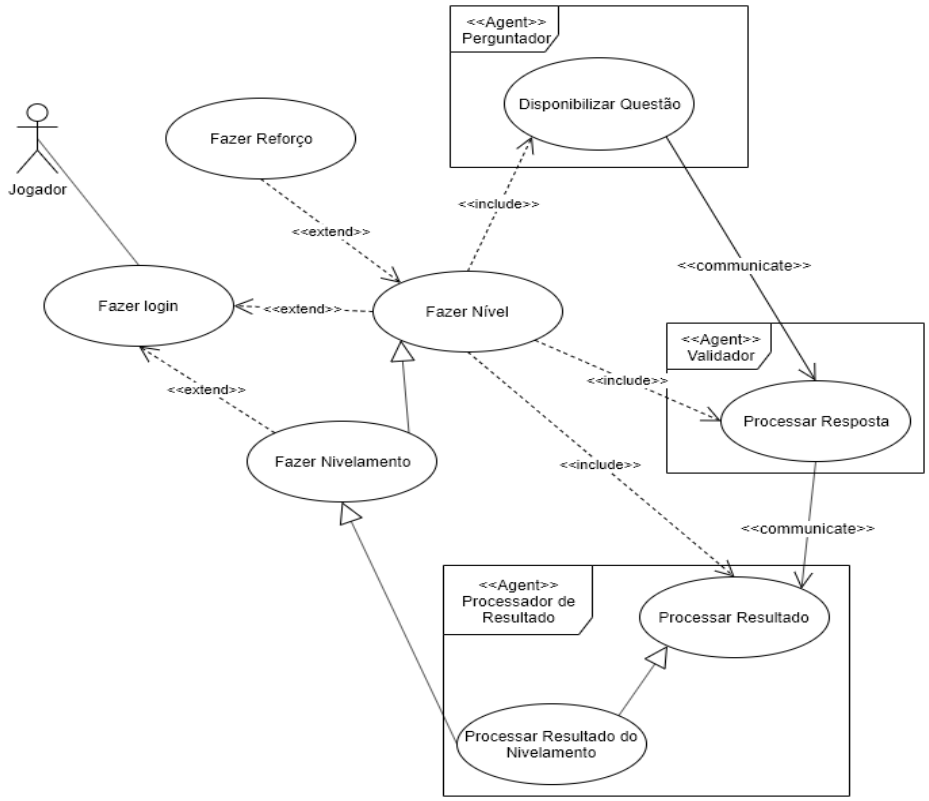


- 3ª iteração:



Identificação dos Agentes

- 2ª iteração:



- 3ª iteração:

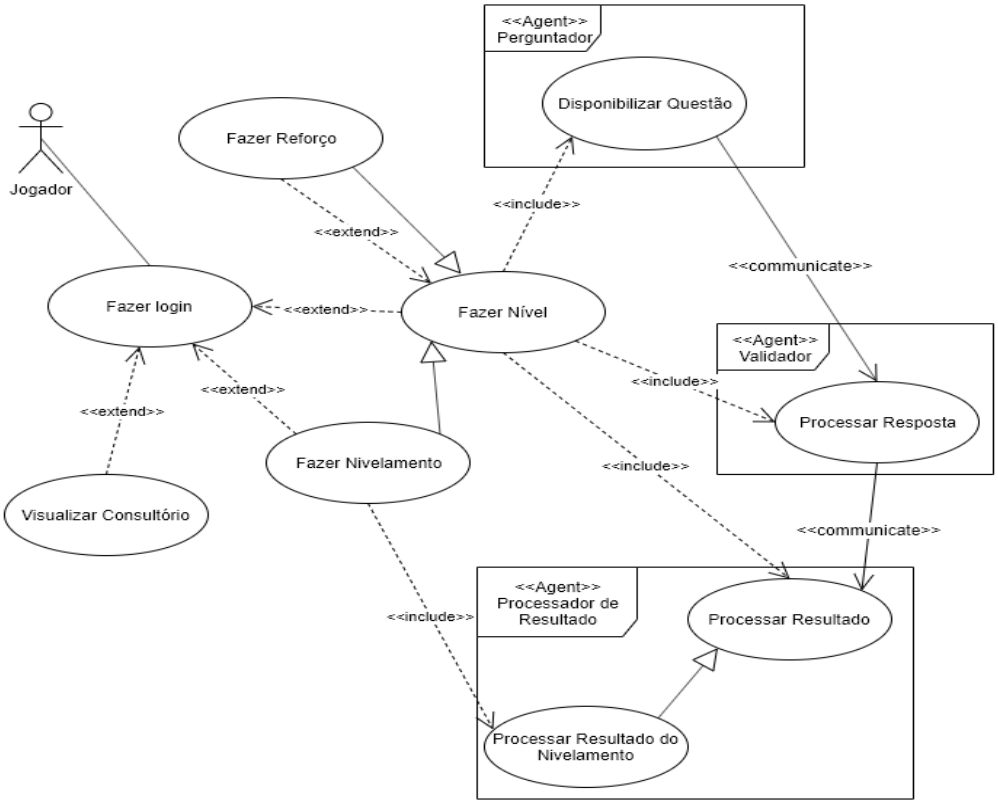
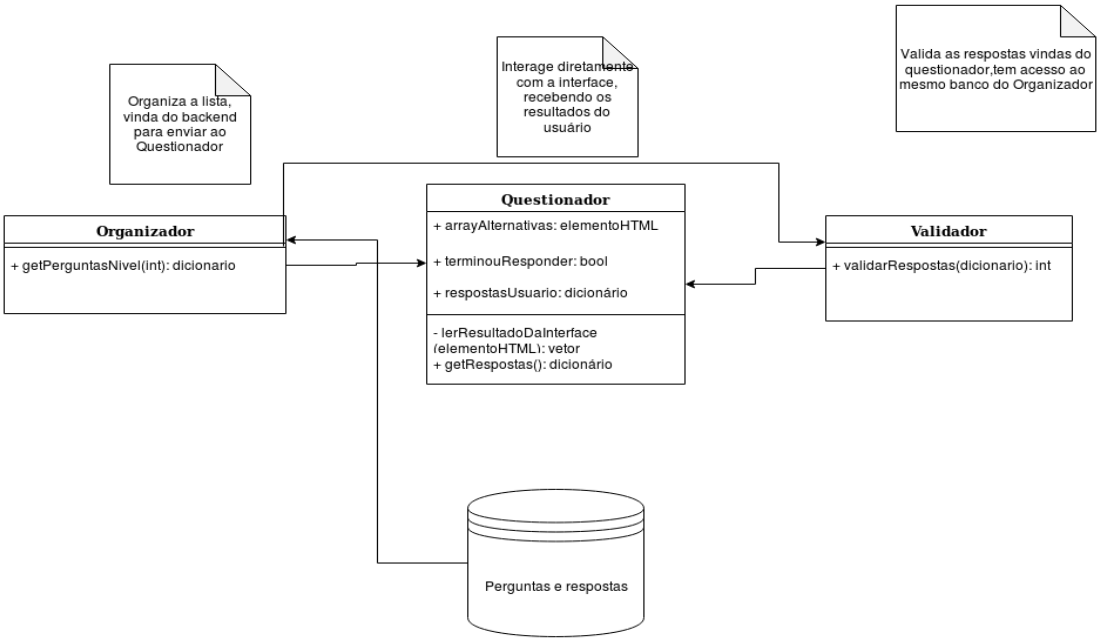
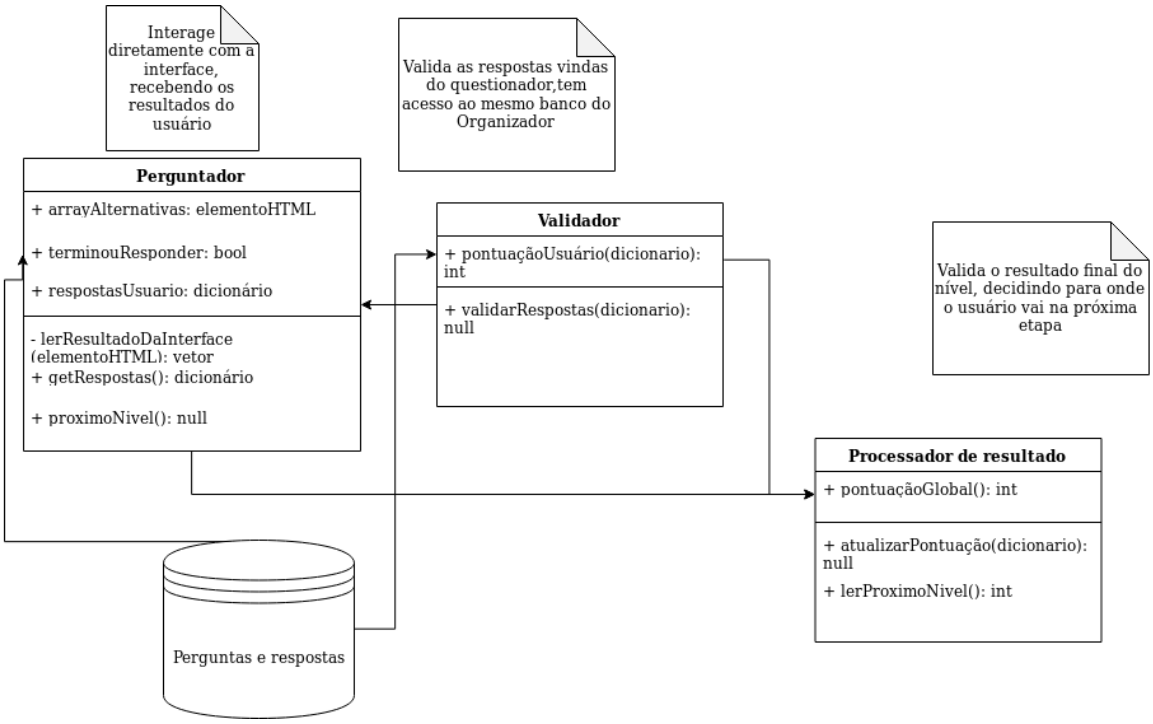


Diagrama de classes (UML)

- 1ª iteração:



- 2ª iteração:


Interage diretamente com a interface, recebendo os resultados do usuário
Valida as respostas vindas do questionador,tem acesso ao mesmo banco do Organizador
Perguntador
+ arrayAlternativas: elementoHTML
+ terminouResponder: bool
+ respostasUsuario: dicionário
- lerResultadoDaInterface (elementoHTML): vetor
+ getRespostas(): dicionário
+ proximoNivel(): null
Validador
+ pontuaçãoUsuário(dicionario): int
+ validarRespostas(dicionario): null
Valida o resultado final do nível, decidindo para onde o usuário vai na próxima etapa
Processador de resultado
+ pontuaçãoGlobal(): int
+ atualizarPontuação(dicionario): null
+ lerProximoNivel(): int
Perguntas e respostas

ANEXO D – Grupo A (Agile Passi) - Modelo de Código

Modelo de código - Typescript

Um Guia de Estilo TypeScript não oficial

Tópicos:

Variável e Função

- Use `camelCase` para nomes de variáveis e funções

Motivo: JavaScript convencional

```
var FooVar;
function BarFunc () {}
```

Boa

```
var fooVar;
function barFunc () {}
```

Class

- Use `PascalCase` para nomes de classe.

Razão: Na verdade, isso é bastante convencional no JavaScript padrão.

Ruim

```
class foo {}
```

Boa

```
class Foo {}
```

- Uso `camelCase` de alunos e métodos de classe

Razão: Naturalmente segue da convenção de nomenclatura de variáveis e funções.

Ruim

```
class Foo {
    Bar : número ;
    Baz () {}
}
```

Boa

```
class Foo {
    bar : número ;
    baz () {}
}
```

Interface

- Use `PascalCase` para o nome.

Razão: semelhante à classe

- Não use prefixo `I`

Razão: Não convencional. `lib.d.ts` define interfaces importantes sem um `I`(por exemplo, Janela, Documento etc).

Ruim

```
interface IFoo {
}
```

Boa

```
interface Foo {
}
```

Tipo

- Use `PascalCase` para o nome.

Razão: semelhante à classe

- Use `camelCase` para membros.

Razão: semelhante à classe

Namespace

- Use `PascalCase` para nomes

Motivo: Convenção seguida pela equipe TypeScript. Os espaços para nome são efetivamente apenas uma classe com membros estáticos. Os nomes de classe são `PascalCase` => os nomes de namespace são `PascalCase`

```
namespace foo {
}
```

Boa

```
namespace Foo {
}
```

Enum

- Use `PascalCase` para nomes de enumeração

Razão: semelhante à classe. É um tipo.

Ruim

```
enum cor {
}
```

Boa

```
enum Cor {
}
```

- Use `PascalCase` para membro enum

Razão: Convenção seguida pela equipe TypeScript, ou seja, pelos criadores da linguagem, por exemplo `SyntaxKind.StringLiteral`. Também ajuda na tradução (geração de código) de outros idiomas para o TypeScript.

Ruim

```
enum Cor {
    vermelho
}
```

Boa

```
enum Cor {
    Vermelho
}
```

- Prefira não usar por indisponibilidade explícita

Razão: esses valores são comumente usados para manter uma estrutura consistente entre os valores. No TypeScript, você usa *tipos* para denotar a estrutura

Ruim

```
let foo = {x: 123 , y: undefined };
```

Boa

```
let foo : {x : number , y ? : number } = {x: 123 };
```

- Use `undefined` em geral (considere retornar um objeto como em `{valid:boolean,value?:Foo}`vez disso)

Ruim

```
return null ;
```

Boa

```
return undefined ;
```

- Use `null` onde faz parte da API ou convencional

Motivo: é convencional no Node.js, por exemplo, `error` é `null` para retornos de chamada no estilo NodeBack.

Ruim

```
cb ( undefined )
```

Boa

```
cb ( null )
```

- Use verificação de *verdade* para objetos sendo `null` ou `undefined`

```
if ( erro === null )
```

Boa

```
if ( erro )
```

- Use `== undefined`/ `!= undefined` (não `===`/ `!==`) para verificar `null`/ `undefined` em primitivas como ele funciona tanto para `null`/ `undefined`, mas não outros valores Falsas (como `''`, `0`, `false`) eg

Ruim

```
if ( erro ! == null )
```

Boa

```
if ( erro ! = undefined )
```

Formatação

O compilador TypeScript é fornecido com um ótimo serviço de linguagem de formatação. Qualquer saída que ela produz por padrão é boa o suficiente para reduzir a sobrecarga cognitiva na equipe.

Use `tsfmt` para formatar automaticamente seu código na linha de comando. Além disso, seu IDE (atom / vscode / vs / sublime) já possui suporte de formatação embutido.

Exemplos:

```
// Espaço antes do tipo ie foo: <space> string
const foo : string = " hello " ;
```

Citações

- Prefira aspas simples (`'`), a menos que escape.

Motivo: mais equipes de JavaScript fazem isso (por exemplo , airbnb, standard, npm, node, google/angular, facebook/react). É mais fácil digitar (não é necessário turno na maioria dos teclados). A equipe mais bonita recomenda aspas simples também

- Quando você não puder usar aspas duplas, tente usar as marcações de retorno (`).

Razão: geralmente representam a intenção de seqüências complexas o suficiente.

Espaços

- Use `2` espaços. Não abas.

Razão: Mais equipes de JavaScript fazem isso (por exemplo , airbnb , idiomático , padrão , npm , nó , google / angular , facebook / react). As equipes TypeScript / VSCode usam 4 espaços, mas são definitivamente a exceção no ecossistema.

Ponto e vírgula

- Use ponto e vírgula.

Razões: ponto e vírgula explícito ajuda as ferramentas de formatação de idioma a fornecer resultados consistentes. A falta de ASI (inserção automática de ponto e vírgula) pode desarmar novos desenvolvedores, por exemplo `foo() \n (function(){})`, será uma única instrução (não duas). TC39 alerta sobre isso também . Exemplos de equipes: airbnb , idiomático , google / angular , facebook / react , Microsoft / TypeScript .

Array

- Anote matrizes como em `foos:Foo[]` vez de `foos:Array<Foo>`.

Razões: É mais fácil de ler. É usado pela equipe TypeScript. Torna mais fácil saber que algo é uma matriz que a mente é treinada para detectar `[]`.

Nomeie arquivos com `camelCase`. Por exemplo `accordian.tsx`, `myControl.tsx`, `utils.ts`, `map.ts` etc.

Razão: Convencional em muitas equipes de JS.

type vs. interface

- Use `type` quando você *pode* precisar de uma união ou cruzamento:

```
type Foo = number | { someProperty: number }
```

- Use `interface` quando quiser `extends` ou `implements` por exemplo

```
interface Foo {
  foo: string;
}
interface FooBar extends Foo {
  bar: string;
}
class X implements FooBar {
  foo: string;
  bar: string;
}
```

- Caso contrário, use o que te faz feliz naquele dia.

ANEXO E – Grupo A (Agile Passi) - Casos de Testes

Plano de testes da metodologia *Agile Passi* – Grupo A
Projeto de Desenvolvimento do Jogo MedEduc

Histórico de Revisão

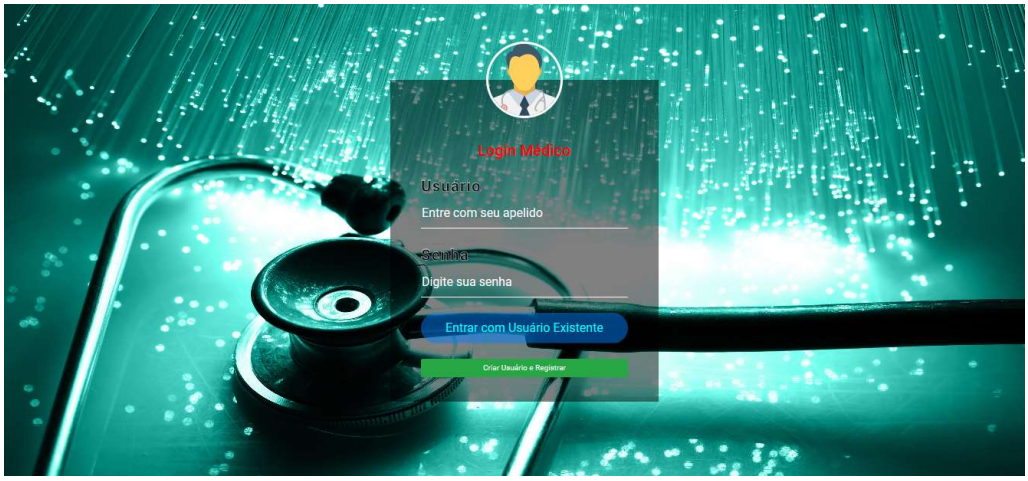
Data	Versão	Descrição
07/11/2019	1.0	Descrição e Resultado dos Testes
15/11/2019	2.0	Classificação e Tipo dos Testes

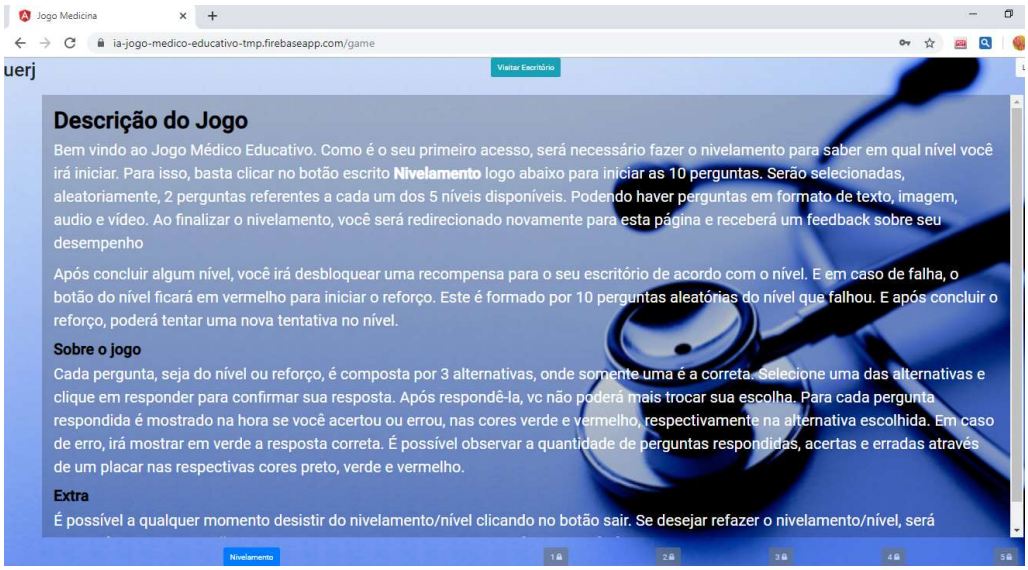
Classificação do teste	Tipos de teste
● Teste unitário ● Teste de interação com a interface	● Da interface com o usuário

Nº teste	Descrição	Resultado Esperado	Resultado Obtido Dentro do Esperado
1	Usuário entrar pela primeira vez no jogo	Deve exibir nivelamento desbloqueado e níveis bloqueados	Sim
2	Usuário responde pergunta corretamente	Deve receber feedback mostrando a opção marcada em verde	Sim
3	Usuário responde pergunta erradamente	Deve receber feedback mostrando a opção correta em verde e a escolhida em vermelho	Sim
4	Usuário responde pergunta	Deve aparecer justificava e ele não pode alterar a resposta selecionada	Sim
5	Usuário respondeu as 10 perguntas do nivelamento	Deve sair da tela de nivelamento, ser redirecionado para tela inicial e saber em qual nível foi nivelado	Sim
6	Usuário erra as 10 perguntas	Deve exibir os níveis bloqueados e o nivelamento disponível	Sim
7	Usuário acerta 1 questão do nível 1	Deve habilitar o nível 1	Sim
8	Usuário acerta todas as questões	Deve fazer o nível 5	Sim
9	Usuário erra mais de 30%	Deve aparecer o nível com a cor vermelha para ele fazer o	Sim

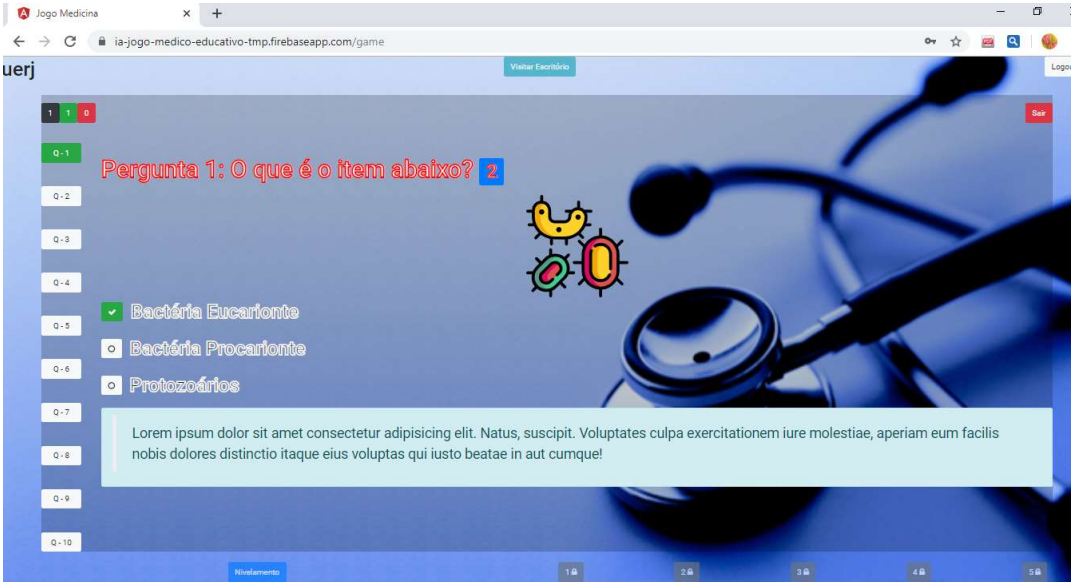
10	Usuário faz reforço	Deve aparecer o nível desbloqueado para ele fazer	Sim
11	Usuário acerta mais de 70%	Deve desbloquear o próximo nível e mostrar recompensa	Sim
12	Usuário inicia nivelamento	Deve conter 2 questões de cada nível	Sim
13	Usuário inicia nível	Deve contar 10 questões do nível	Sim
14	Usuário sai do jogo e volta	Deve ter as informações salvas	Sim
15	Usuário sai do reforço	Deve continuar com botão do nível em vermelho para fazer o reforço	Sim
16	Usuário sai do nível	Deve continuar com o botão do nível em azul para fazer o nível novamente	Sim
17	Usuário reinicia reforço	Deve desconsiderar todas as resposta respondidas anteriormente	Sim
18	Usuário reinicia nível	Deve desconsiderar todas as respostas respondidas anteriormente	Sim
19	Usuário tenta iniciar nível já concluído	Não deve permitir que o nível inicie	Sim
20	Usuário respondeu às 10 pergunta do nível	Deve processar o resultado e dá feedback	Sim

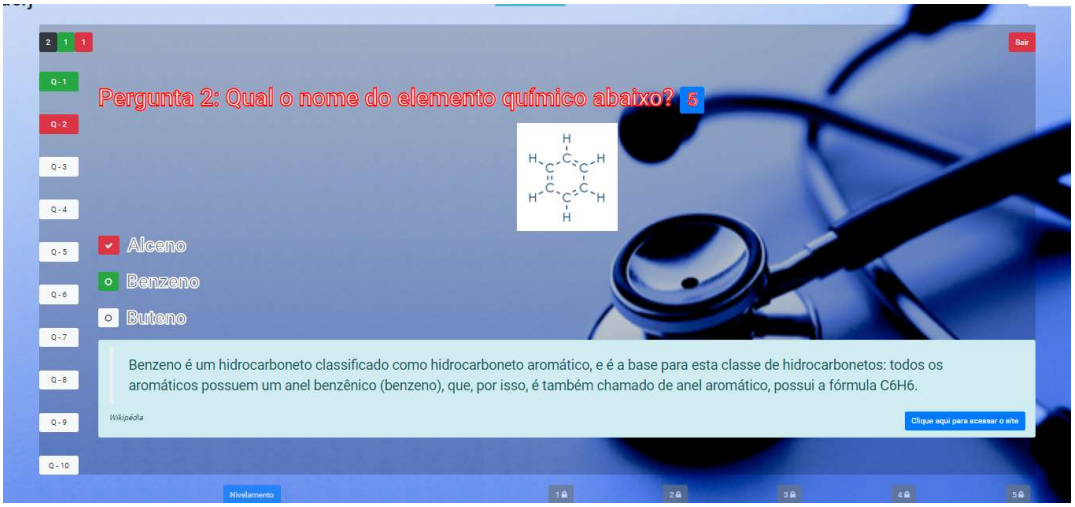
ANEXO F – Grupo A (Agile Passi) - Jogo Desenvolvido

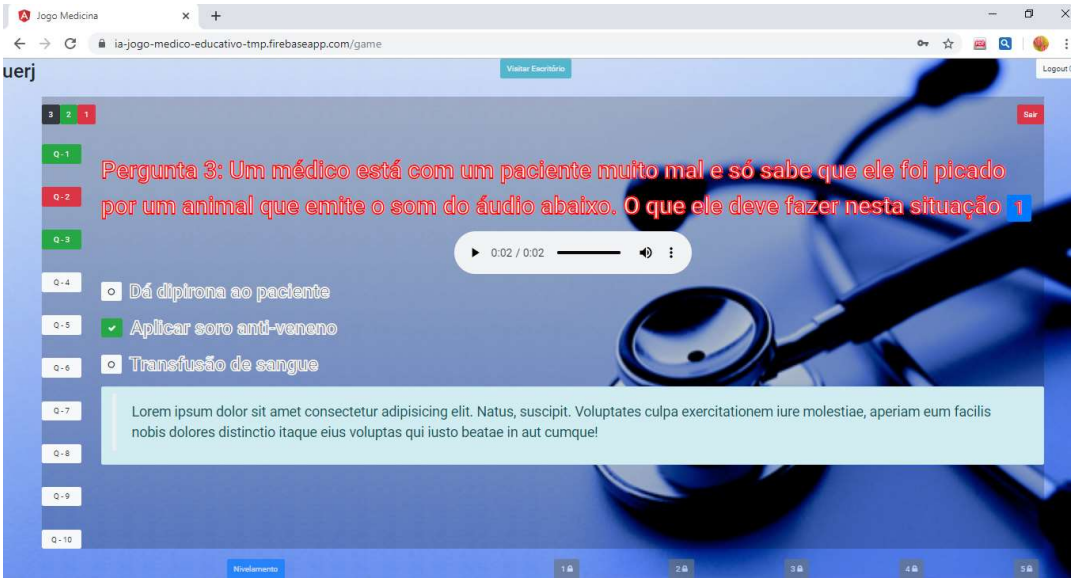


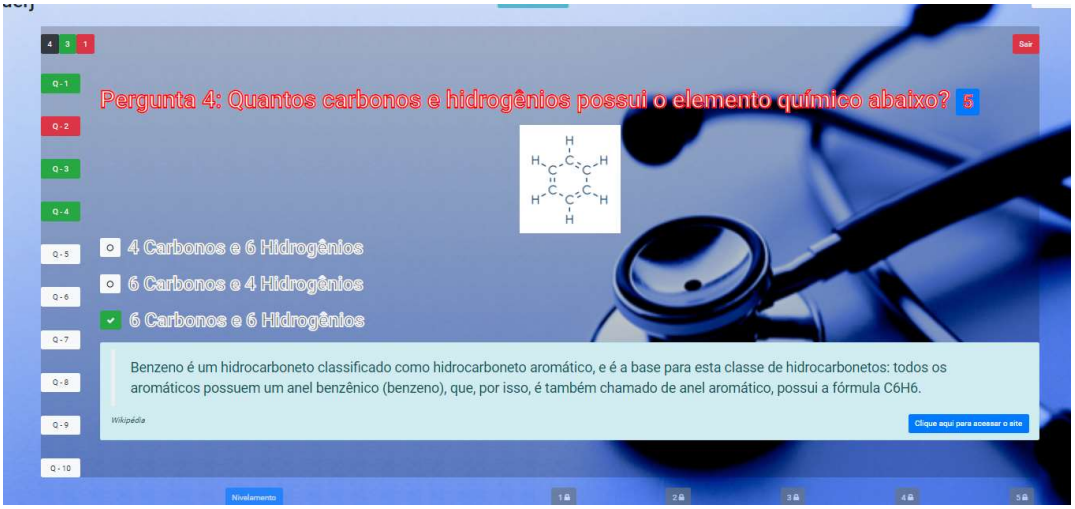



Termometro
Recompensa Nível 1
Bloqueado
Remedios
Recompensa Nível 2
Bloqueado
Cama Hospital
Recompensa Nível 3
Bloqueado
Ambulancia
Recompensa Nível 4
Enfermeira
Recompensa Nível 5
Nivelamento

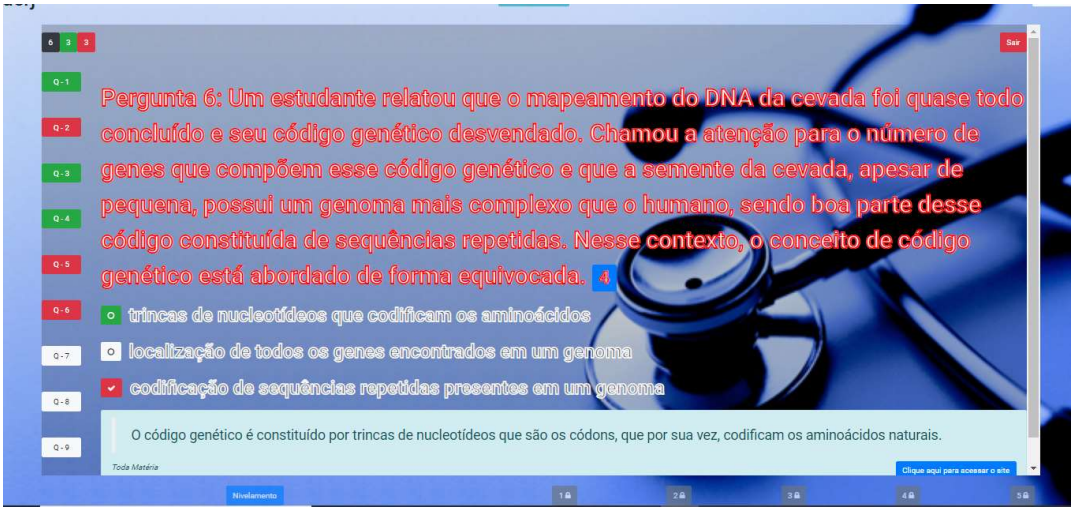

Jogo Medicina
ia-jogo-medico-educativo-tmp.firebaseapp.com/game
uerj
Visitar Escritório
Sair
Pergunta 1: O que é o item abaixo? 2
Bactéria Eucarionte
Bactéria Procarionte
Protozoários
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus, suscipit. Voluptates culpa exercitationem iure molestiae, aperiam eum facilis nobis dolores distinctio itaque eius voluptas qui iusto beatae in aut cumque!
Nivelamento

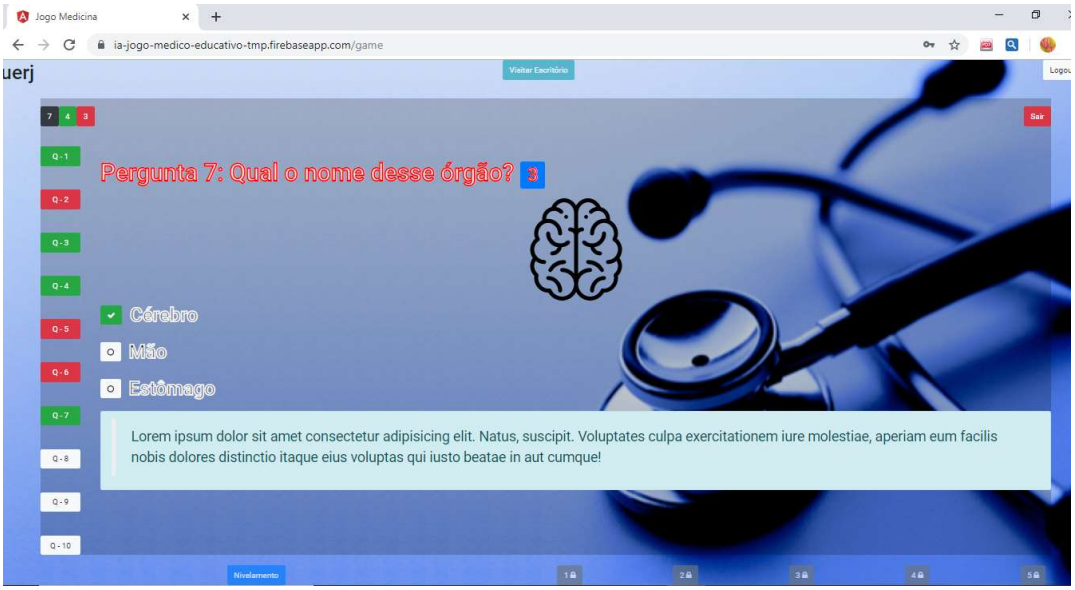

Sair
Pergunta 2: Qual o nome do elemento químico abaixo? 5
Alceno
Benzeno
Buteno
Benzeno é um hidrocarboneto classificado como hidrocarboneto aromático, e é a base para esta classe de hidrocarbonetos: todos os aromáticos possuem um anel benzênico (benzeno), que, por isso, é também chamado de anel aromático, possui a fórmula C6H6.
Wikipédia
Clique aqui para acessar o site
Nivelamento


Jogo Medicina
ia-jogo-medico-educativo-tmp.firebaseapp.com/game
uerj
Visitar Escritório
Logout
Sair
Pergunta 3: Um médico está com um paciente muito mal e só sabe que ele foi picado por um animal que emite o som do áudio abaixo. O que ele deve fazer nesta situação 1
0:02 / 0:02
Dá dipirona ao paciente
Aplicar soro anti-veneno
Transfusão de sangue
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus, suscipit. Voluptates culpa exercitationem iure molestiae, aperiam eum facilis nobis dolores distinctio itaque eius voluptas qui iusto beatae in aut cumque!
Nivelamento


4 3 1
Sair
Q-1
Q-2
Q-3
Q-4
Q-5
Q-6
Q-7
Q-8
Q-9
Q-10
Pergunta 4: Quantos carbonos e hidrogênios possui o elemento químico abaixo? 5
4 Carbonos e 6 Hidrogênios
6 Carbonos e 4 Hidrogênios
6 Carbonos e 6 Hidrogênios
Benzeno é um hidrocarboneto classificado como hidrocarboneto aromático, e é a base para esta classe de hidrocarbonetos: todos os aromáticos possuem um anel benzênico (benzeno), que, por isso, é também chamado de anel aromático, possui a fórmula C6H6.
Wikipédia
Clique aqui para acessar o site
Nivelamento
1
2
3
4
5

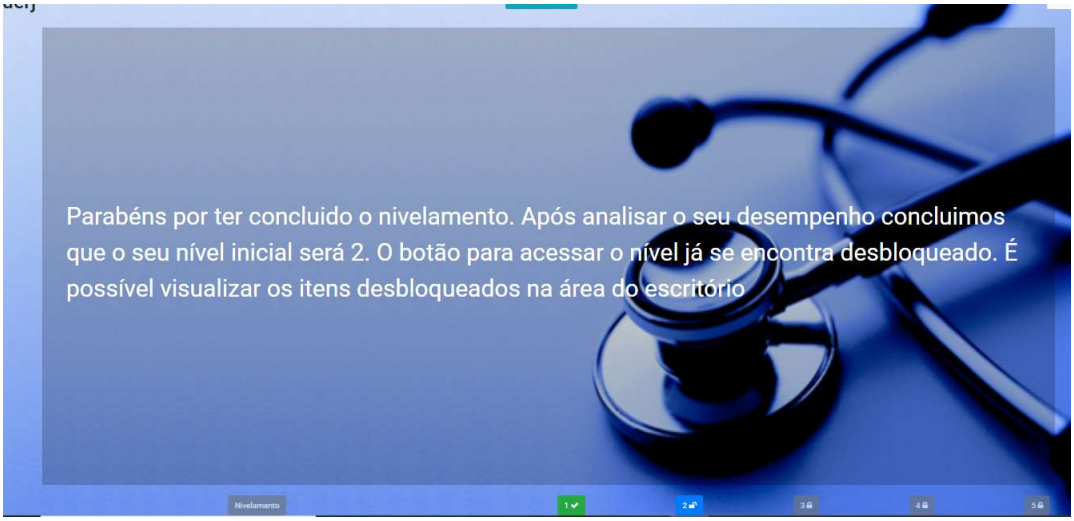

Jogo Medicina
ia-jogo-medico-educativo-tmp.firebaseapp.com/game
uerj
Visitar Escritório
Logout
5 3 2
Sair
Q-1
Q-2
Q-3
Q-4
Q-5
Q-6
Q-7
Q-8
Q-9
Q-10
Pergunta 5: Quantos carbonos e hidrogênios possui o elemento químico abaixo? 3
1 Carbono e 2 Hidrogênios
2 Carbonos e 4 Hidrogênios
1 Carbono e 4 Hidrogênios
Alcano possui 1 carbono e 4 Hidrogênios
Wikipédia
Clique aqui para acessar o site
Nivelamento
1
2
3
4
5


Sair
Pergunta 6: Um estudante relatou que o mapeamento do DNA da cevada foi quase todo concluído e seu código genético desvendado. Chamou a atenção para o número de genes que compõem esse código genético e que a semente da cevada, apesar de pequena, possui um genoma mais complexo que o humano, sendo boa parte desse código constituída de sequências repetidas. Nesse contexto, o conceito de código genético está abordado de forma equivocada.
trincas de nucleotídeos que codificam os aminoácidos
localização de todos os genes encontrados em um genoma
codificação de sequências repetidas presentes em um genoma
O código genético é constituído por trincas de nucleotídeos que são os códons, que por sua vez, codificam os aminoácidos naturais.
Nivelamento


Jogo Medicina
ia-jogo-medico-educativo-tmp.firebaseapp.com/game
Sair
Pergunta 7: Qual o nome desse órgão?
Cérebro
Mão
Estômago
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus, suscipit. Voluptates culpa exercitationem iure molestiae, aperiam eum facilis nobis dolores distinctio itaque eius voluptas qui iusto beatae in aut cumque!
Nivelamento


Sair
Pergunta 8: Qual profissional médico é responsável por cuidar de ... ? 2
Dentista
Clínico Geral
Veterinário
Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus, suscipit. Voluptates culpa exercitationem iure molestiae, aperiam eum facilis nobis dolores distinctio itaque eius voluptas qui iusto beatae in aut cumque!
Nivelamento


Jogo Medicina
ia-jogo-medico-educativo-tmp.firebaseapp.com/game
Visitar Escritório
Sair
Pergunta 9: Qual a especialidade médica estudada para se cuidar do ouvido, do nariz, seios paranasais, faringe, laringe, cabeça e pescoço ? 1
Cardiologista
Giniciologista
Otorrinolaringologista
A otorrinolaringologia (ORL) é uma especialidade médica com características clínicas e cirúrgicas. Seu campo de atuação envolve as doenças do ouvido, do nariz e seios paranasais, faringe, laringe, cabeça e pescoço.
Wikipedia
Clique aqui para acessar o site
Nivelamento


Parabéns por ter concluido o nivelamento. Após analisar o seu desempenho concluimos que o seu nível inicial será 2. O botão para acessar o nível já se encontra desbloqueado. É possível visualizar os itens desbloqueados na área do escritório
Nivelamento
1
2
3
4
5

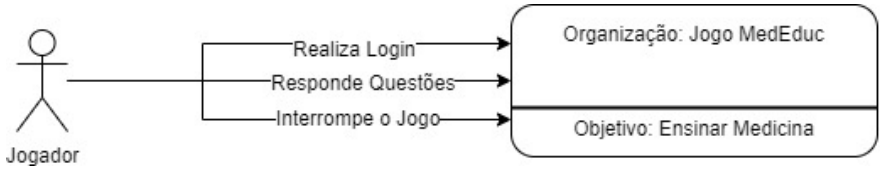

Jogo Medicina
ia-jogo-medico-educativo-tmp.firebaseapp.com/game
uerj
Visitar Escritório
Logo
Sair
Q - 1
Q - 2
Q - 3
Q - 4
Q - 5
Q - 6
Q - 7
Q - 8
Q - 9
Pergunta 1: Qual é o nome da especialidade que faz diagnostico por imagem? (RAIO-X)
2
Radiologia
Mamografia
Ginecologia
A ultrassonografia ou ecografia é um método diagnóstico muito recorrente na medicina moderna que utiliza o eco gerado através de ondas ultrassônicas de alta frequência para visualizar, em tempo real, as estruturas internas do organismo. Por meio de uma ultrassonografia com doppler, o médico é capaz de ver o fluxo sangüíneo nos principais vasos.
Wikipedia
Clique aqui para acessar o site
Nivelamento
1
2
3
4
5

ANEXO G – Grupo C (Ingenias-Scrum) - Modelo de Organização

Modelo de Organização

Data	Versão	Descrição	Autor
13/11/2019	1.0	Criação modelo de organização	Grupo C

Diagrama de Organização



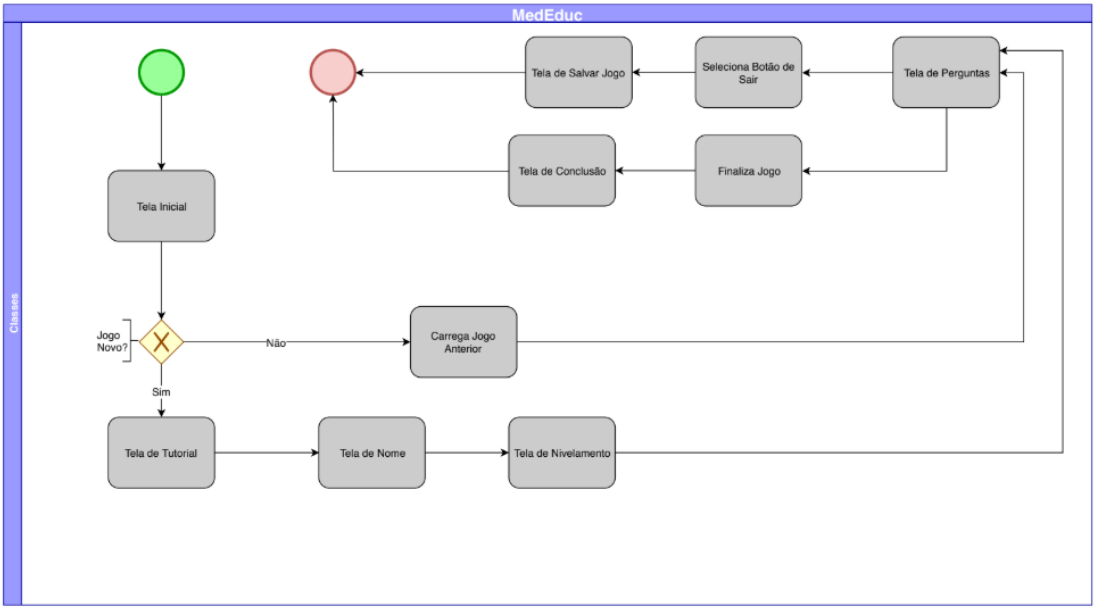
ANEXO H – Grupo C (Ingenias-Scrum) - Modelo de Agente e Ambiente

Diagrama de Agentes – Grupos C

Histórico de Revisões

Data	Versão	Descrição	Autor
31/10/2019	1.0		Grupo C

Versão 1.0



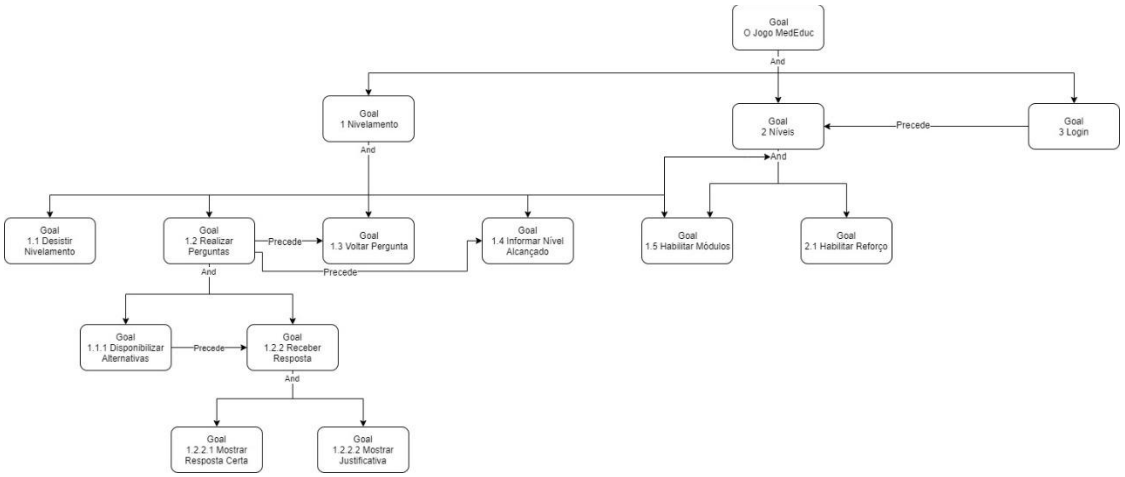
Versão 2.0

ANEXO I – Grupo C (Ingenias-Scrum) - Modelo de Metas e Tarefas

Modelo de Metas e Tarefas

Data	Versão	Descrição	Autor
13/11/2019	1.0	Criação Modelo de Metas e Tarefas	Grupo C

Diagrama de Metas e Tarefas



ANEXO J – Grupo C (Ingenias-Scrum) - Modelo de Desenvolvimento

Modelo de Desenvolvimento

Histórico de Revisões

Data	Versão	Descrição	Autor
25/11/2019	1.0	Criação do Documento	Grupo C

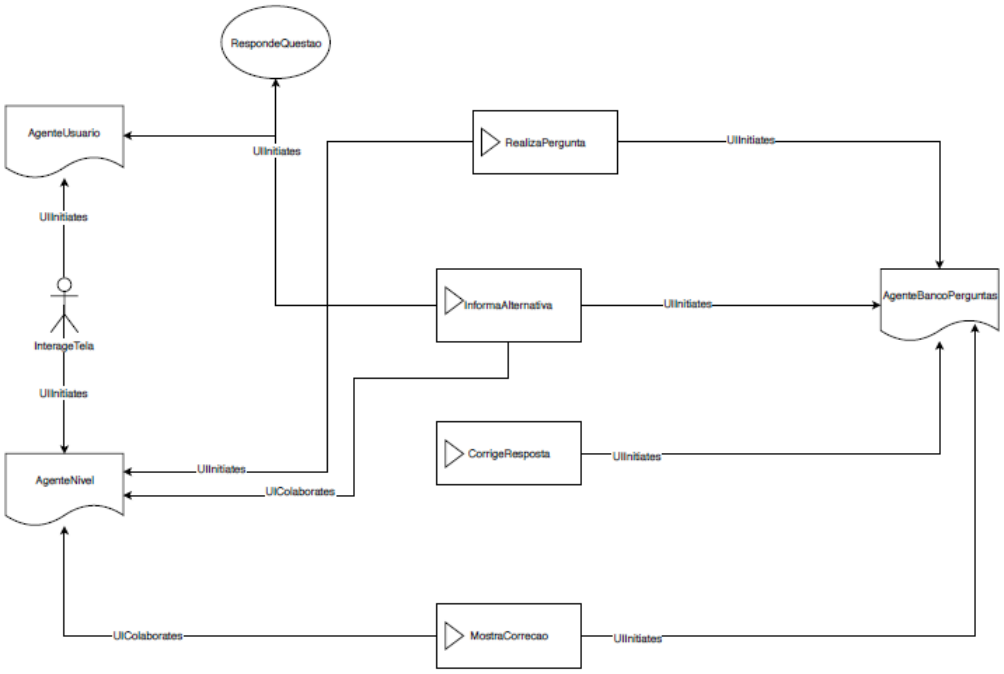
1. Todas as Classes devem seguir o padrão Nome_Da_Classe com cada primeira letra da palavra utilizando maiúsculas e _ entre elas.
2. Todos os agentes devem seguir o padrão Agente_Nome_Do_Agente.
3. Todas as variáveis devem ser escritas em letras minúsculas.
4. Todos os botões devem ter o nome btnNomebotao
5. Todas as imagens devem ser escritas com letra minúscula

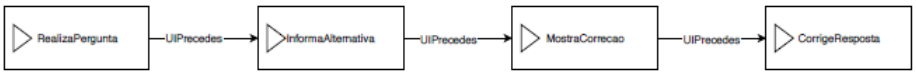
ANEXO K – Grupo C (Ingenias-Scrum) - Modelo de Interação

Modelo de Interação – Grupos C

Histórico de Revisões

Data	Versão	Descrição	Autor
25/11/2019	1.0	Criação do Documento	Grupo C



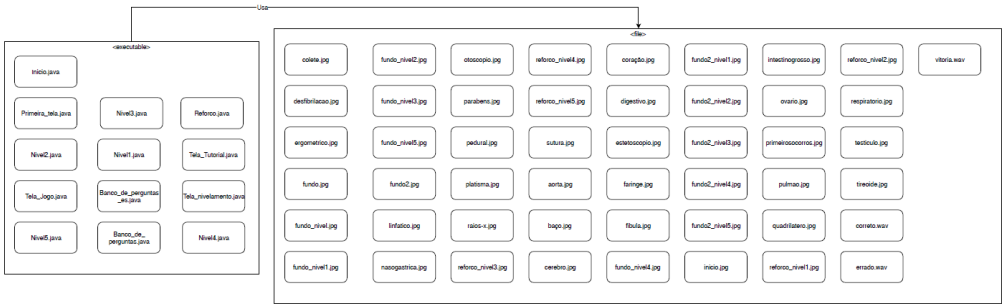


ANEXO L – Grupo C (Ingenias-Scrum) - Modelo de Componentes

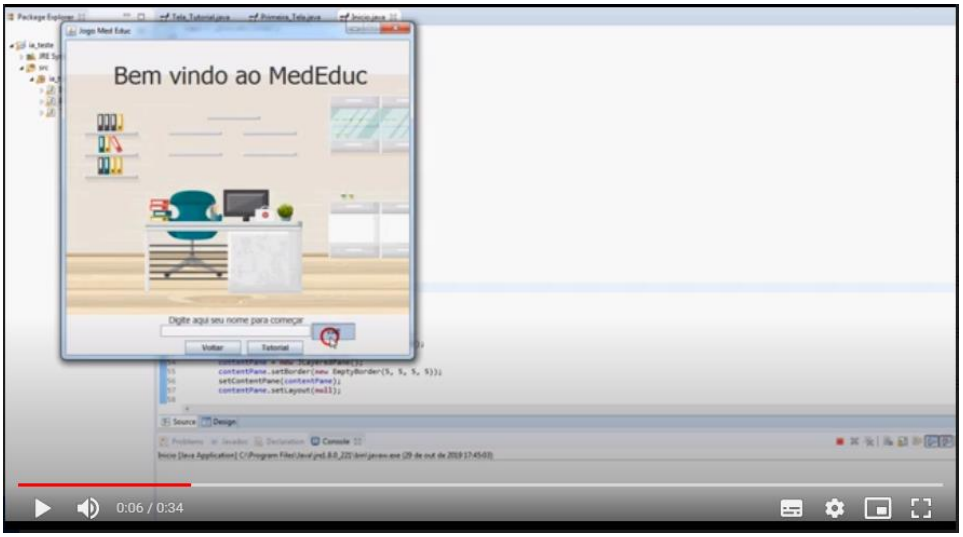
Modelo de Componentes – Grupos C

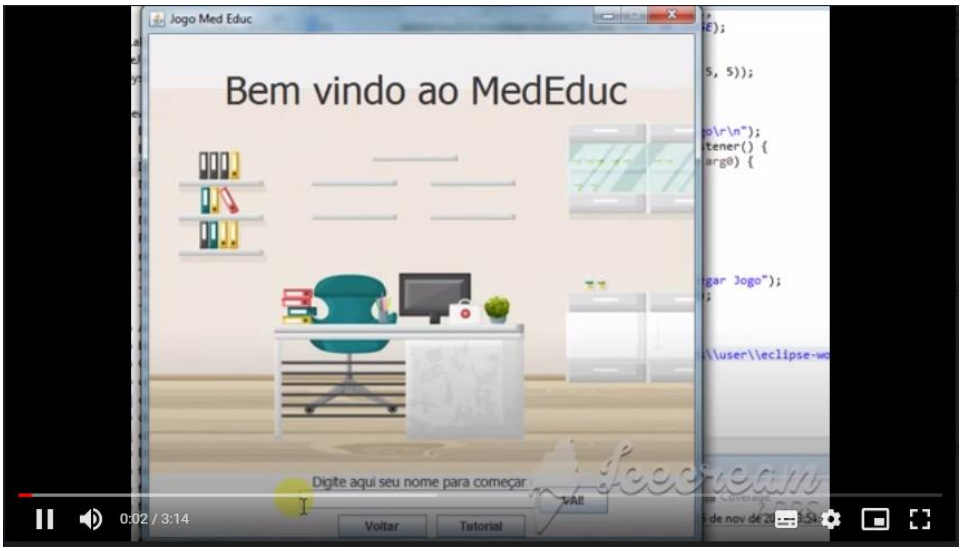
Histórico de Revisões

Data	Versão	Descrição	Autor
25/11/2019	1.0	Criação do Documento	Grupo C

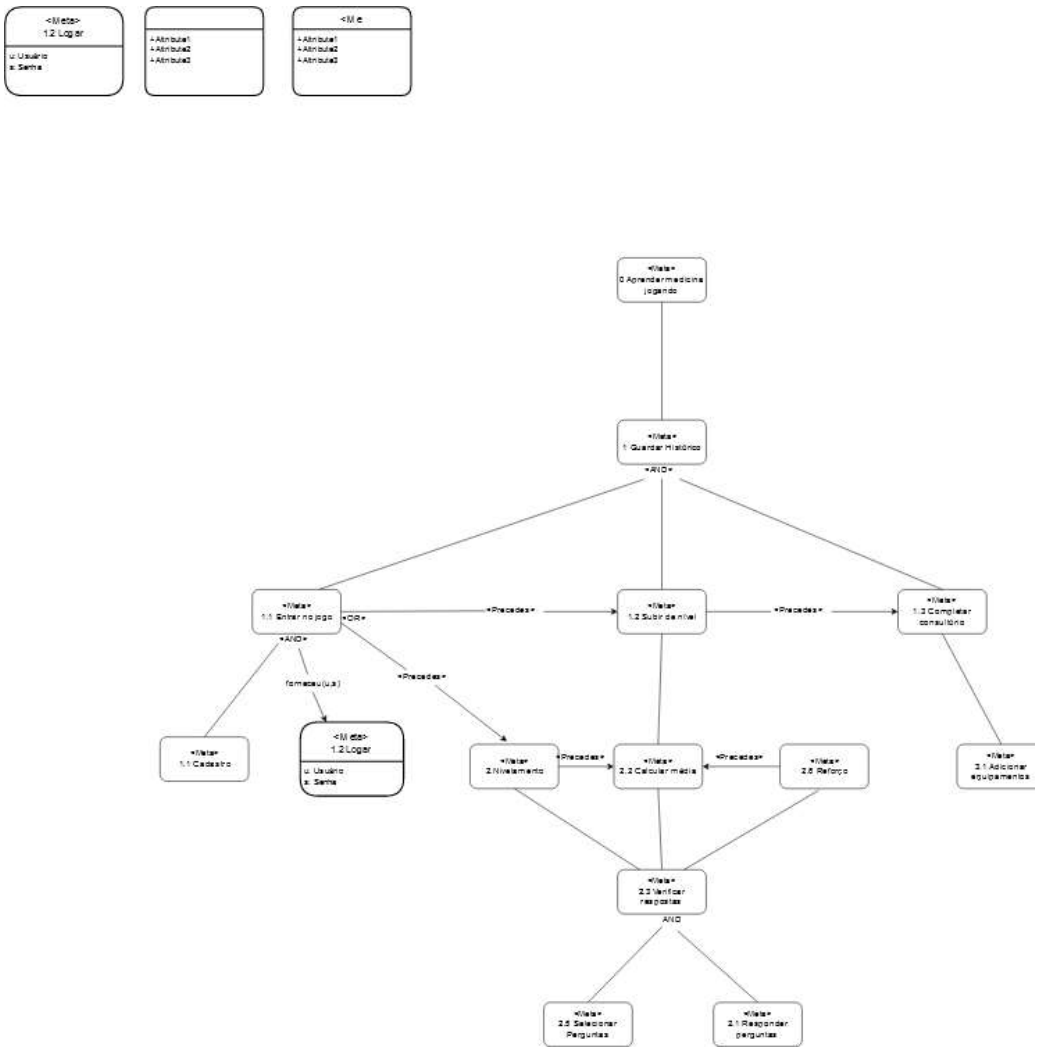


ANEXO M – Grupo C (Ingenias-Scrum) - Jogo Desenvolvido

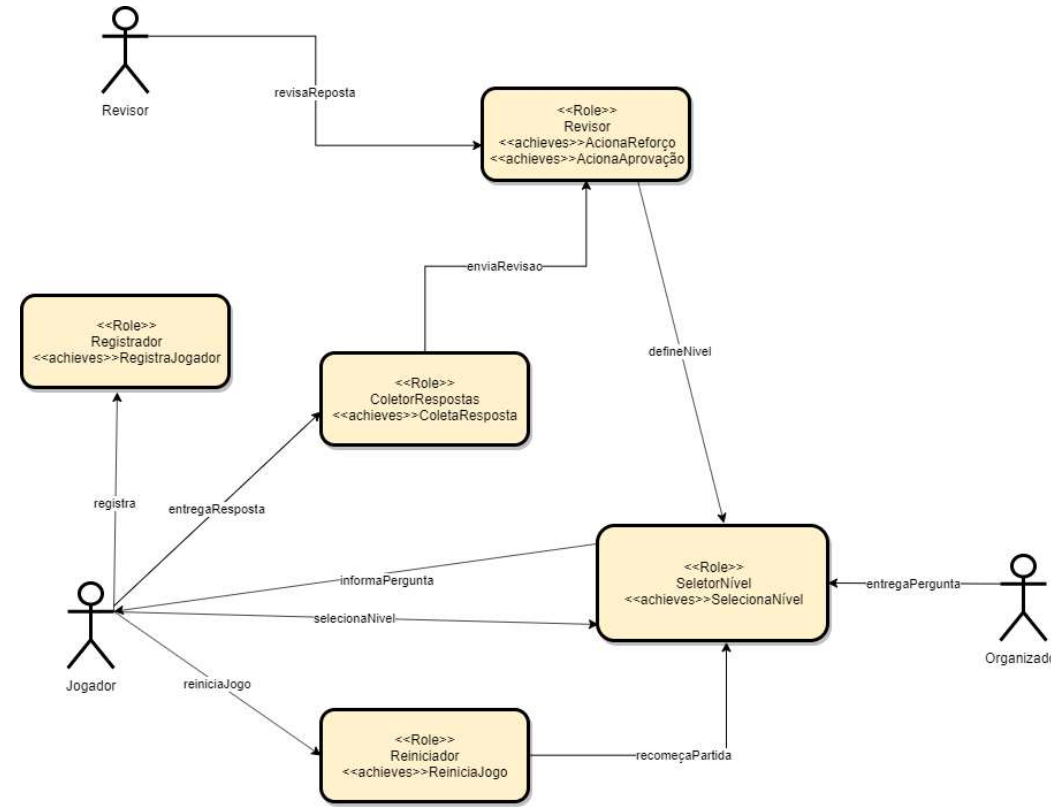




ANEXO N – Grupo E (Agile O-MaSE) - Modelo de Metas



ANEXO O – Grupo E (Agile O-MaSE) - Modelo de Papéis



ANEXO P – Grupo E (Agile O-MaSE) - Casos de Testes

Casos de Teste	
Suíte de Testes: Acesso ao Jogo	
Caso de Teste: CT-1	
Objetivo: Verificar o fluxo de acesso ao jogo de usuários que irão se cadastrar na plataforma	
Pré-condições: Usuário ter adquirido o aplicativo em uma loja virtual e não ter cadastro na plataforma	
Ações	**Resultados Esperados**
Acessar o aplicativo instalado previamente	- Abrir o aplicativo mostrando a splash screen - Exibir o formulário de login, com três botões de interação, login, esqueci senha e cadastro
Clicar no botão de cadastro	Deverá exibir formulário de cadastro com os seguintes campos: - Nome Completo - E-mail - Senha - Botão de envio
Prencher informações e clicar no botão de envio	Deverá exibir retorno de validação do formulário, informando em caso de sucesso: - Exibir modal com mensagem de sucesso e botão para prosseguimento para tela do jogo Em caso de falha: - Exibir modal com mensagem informando o motivo de falha e botão para fechar o modal e retornar ao fomulário
Clicar no botão do modal	Deverá enviar para a tela de nivelamento do jogo
Caso de Teste: CT-2	
Objetivo: Verificar o fluxo de acesso ao jogo de usuários que possuem cadastro e irão acessar a plataforma	
Pré-condições: Usuário ter adquirido o aplicativo em uma loja virtual e ter cadastro na plataforma	
Ações	**Resultados Esperados**
Acessar o aplicativo instalado previamente	- Abrir o aplicativo mostrando a splash screen - Exibir o formulário de login, com três botões de interação, login, esqueci senha e cadastro
Prencher informações e clicar no botão de login	Deverá exibir modal de validação do acesso, em caso de sucesso com: - Mensagem de sucesso - Redirecionando para a tela onde o jogador salvou pela última vez Em caso de insucesso: - Exibir mensagem de insucesso - Exibir botão para fechar modal

ANEXO Q – Grupo E (Agile O-MaSE) - Modelo de Políticas

Modelo de Políticas
Metodologia *Agile O-MaSE* – Grupos E
Projeto de Desenvolvimento do Jogo Médico Educacional

Histórico de Revisões

Data	Versão	Descrição
04/12/2019	1.0	Criação do documento e preenchimento das políticas: 1, 2 e 3

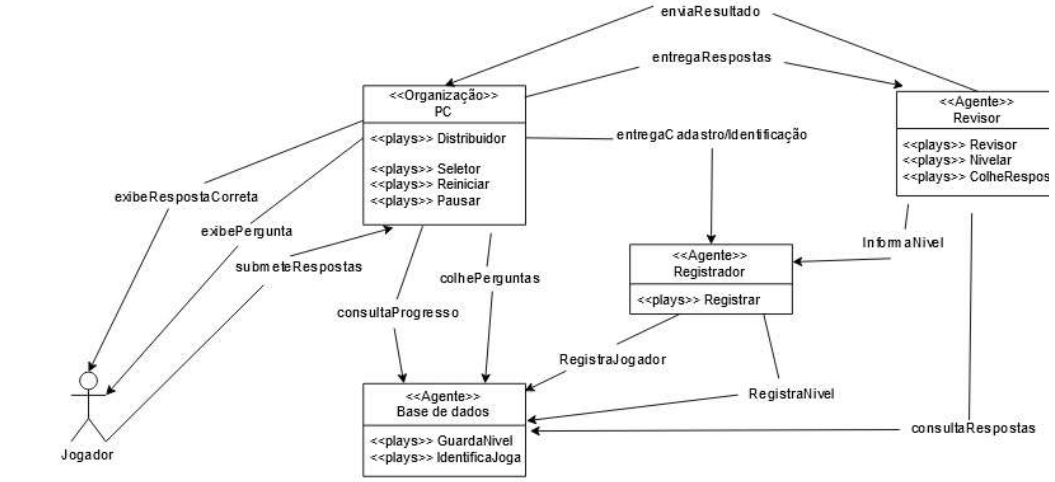
Descrição das Políticas

1. **O Acesso a plataforma só poderá ser feito através de uma instância de Jogador**
   Toda as atividades devem ser executadas levando em conta uma instância de Jogador, de modo que todos os outros agentes e seus papéis esperam uma identificação do Jogador no protocolo que estará solicitando a execução da atividade. A instância de Jogador é atribuída por usuário ao acessar a plataforma após cadastro através do agente Registrador que concederá um Jogador por e-mail.

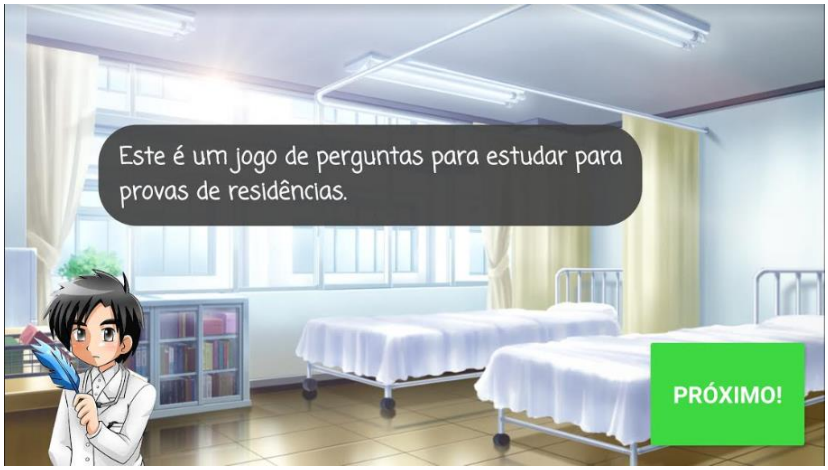
2. **Limite de nível alcançado no Nivelamento**
   No Nivelamento, o agente Revisor deve direcionar o Jogador ao nível 5, quando o mesmo acertar todas as respostas ou todas as respostas até o nível 4. Aos demais níveis, o Jogador será redirecionado, quando acertar todas as respostas até um determinado nível, ao nível posterior.

3. **Restrição de Acesso do Jogador aos níveis**
   O agente Revisor só deve permitir acesso ao Jogador ao nível desejado, se e somente se ele tiver sido habilitado através do Nivelamento ou através da passagem níveis concedida também pelo Revisor através da avaliação das respostas.

ANEXO R – Grupo E (Agile O-MaSE) - Modelo de Classes de Agentes



ANEXO S – Grupo E (Agile O-MaSE) - Jogo Desenvolvido



ANEXO T – Grupo F (Agile O-MaSE) - Modelo de Metas

Plano de Release da metodologia *Agile O-MaSE* – Grupos F
Projeto de Desenvolvimento do Jogo Médico Educacional

Histórico de Revisões

Data	Versão	Descrição	Autor
17/10/2019	1.0	Criação do documento e preenchimento das informações: 1, 2, 3 e 4.	[redacted]
30/10/2019	2.0	Associação dos Itens de Backlog do Produto a Iteração 1 e Modelo de Metas. - Iteração 1	[redacted]
14/11/2019	3.0	Associação dos Itens de Backlog do Produto a Iteração 2 e Modelo de Metas atualizado. - Iteração 2	[redacted]
05/12/2019	4.0	Modelo de Metas atualizado. - Iteração 3	[redacted]

1. Definição do Critério de Sucesso da *Release*

Este projeto é orientado por data, e deve ser finalizado no dia 05/12/19. Consequentemente, as entregas das Iterações são do tipo *End-Date Driven,* ou seja, orientada para data de termino, cujas as entregas devem estar disponíveis antes da data limite.

Os critérios para considerar uma release terminado com sucesso são:

- A entrega de todas as histórias do *backlog da Iteração;*
- A qualidade da entrega: requisitos corretos e sem bugs e
- A apresentação dos modelos e artefatos que a metodologia requer.

O projeto possui um Comitê de Qualidade, composto por: Fabiana Gominho, Vera Werneck e Rosa Costa, que fará a aceitação das entregas de cada iteração.

2. Estimativa dos Itens de *Backlog* do Produto

A estimativa da complexidade de desenvolvimento deve ser analisada pelo Time de desenvolvimento, item por item do *Backlog* e deve ser iniciado pelo item de maior prioridade. A técnica utilizada na estimativa dos itens do *backlog*, deve ser a sequência de número *Fibonacci*: 1, 2, 3, 5, 8 e 13. Quanto maior for o número, maior será a complexidade de desenvolvimento.

3. Definição do Tamanho e Quantidade de Iterações

O tamanho da Iteração está definido limitado a data final do projeto, num *time-box* de 2 semanas, dando um total de 3 iterações.

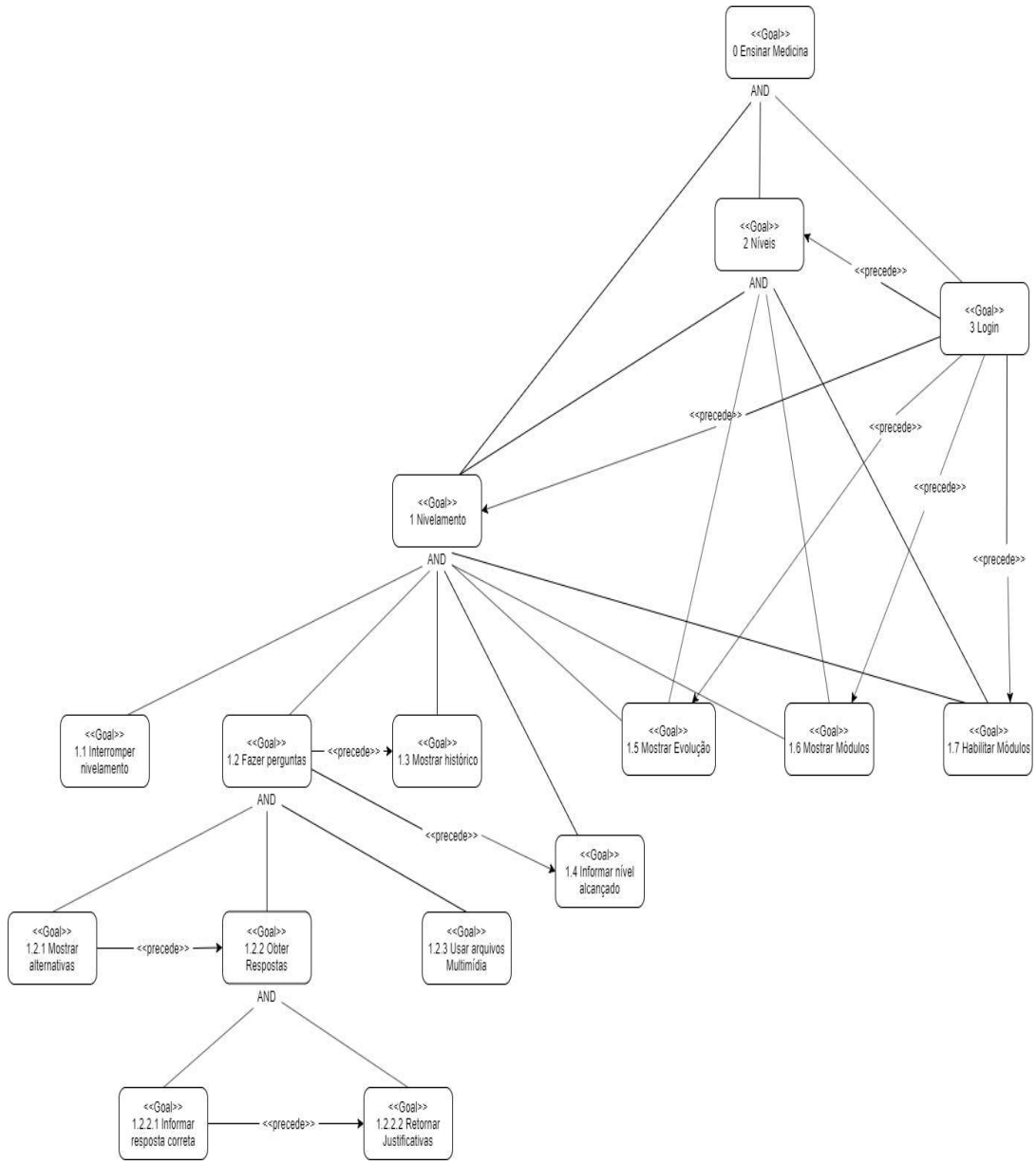
4. Estimativa da Velocidade

A estimativa da velocidade será realizada de forma manual, observando a primeira iteração e ajustando as demais iterações.

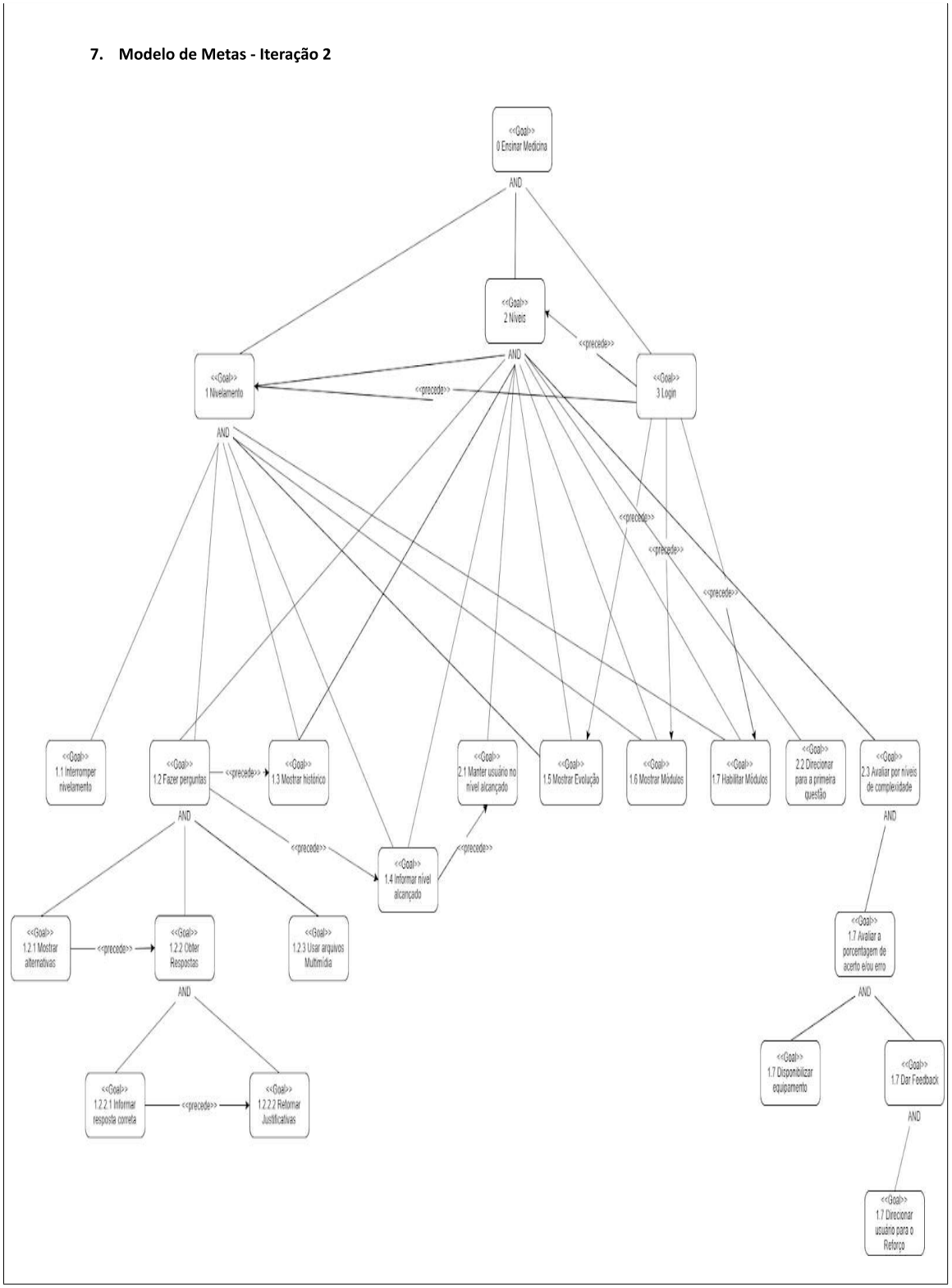
5. Associação dos Itens de *Backlog* do Produto as Iterações

Iteração	Backlog do Produto
1	1, 2, 3, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
2	4, 5, 6, 18, 19, 20, 21, 22

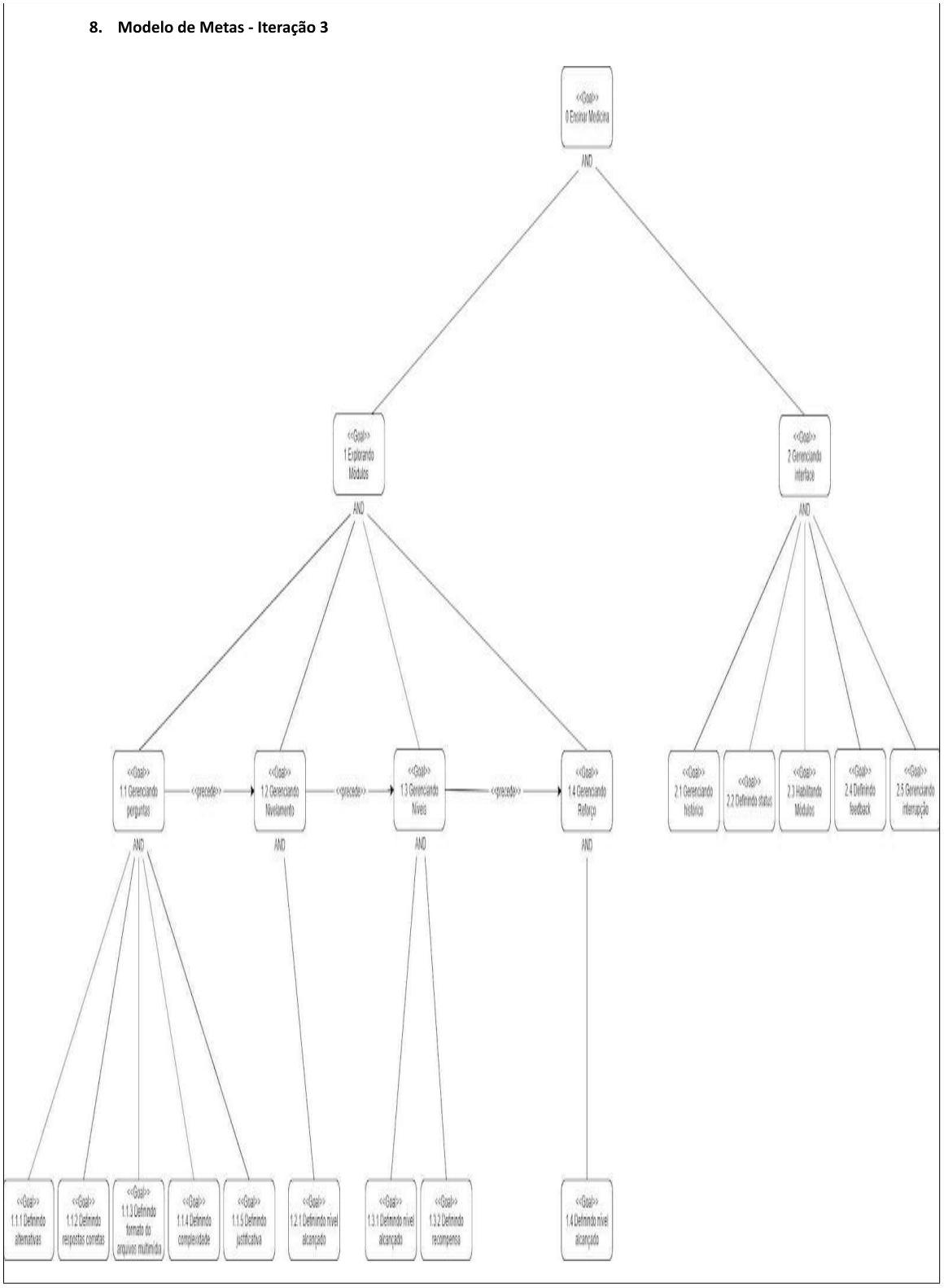
6. Modelo de Metas - Iteração 1



7. Modelo de Metas - Iteração 2



8. Modelo de Metas - Iteração 3



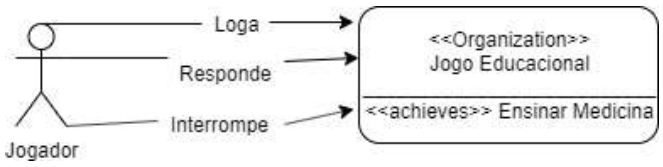
ANEXO U – Grupo F (Agile O-MaSE) - Modelo de Interface com a Organização e Modelo de Papéis

Atividade Análise da Solução da metodologia *Agile O-MaSE* – Grupos F
Projeto de Desenvolvimento do Jogo Médico Educacional

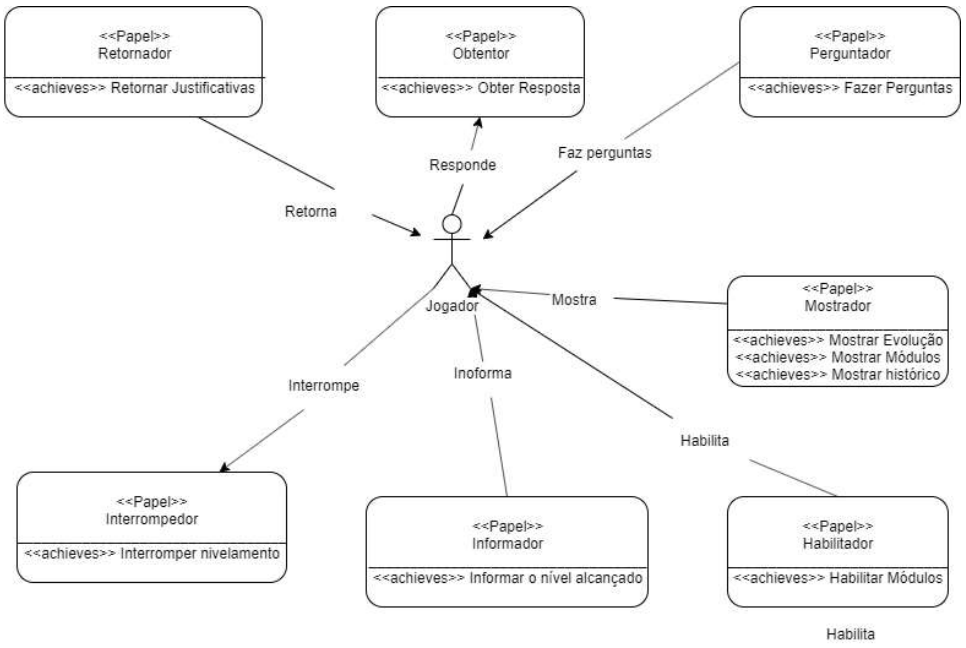
Histórico de Revisões

Data	Versão	Descrição	Autor
30/10/2019	1.0	Criação do documento, do Modelo de Interface com a Organização e do Modelo de papéis para iteração 1.	[redacted]
14/11/2019	2.0	Atualização do Modele de Interface com a Organização e do Modelo de papéis para iteração 2.	[redacted]
05/12/2019	3.0	Atualização do Modele de Interface com a Organização e do Modelo de papéis para iteração 3.	[redacted]

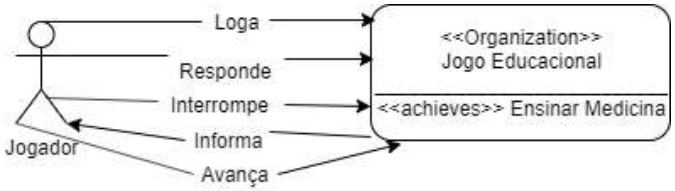
1. Modelo de Interface com a Organização - Iteração 1



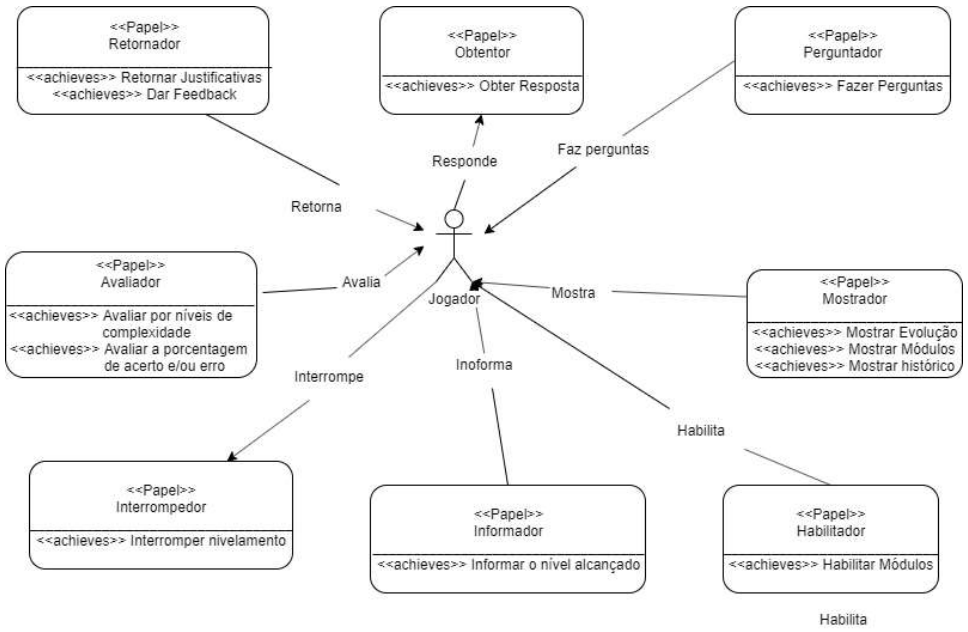
2. Modelo de Papeis - Iteração 1



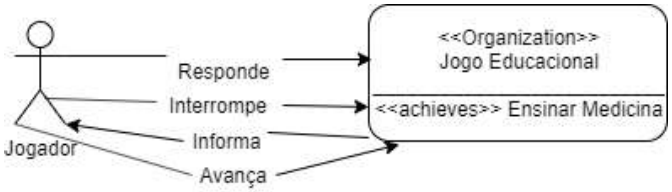
3. Modelo de Interface com a Organização - Iteração 2



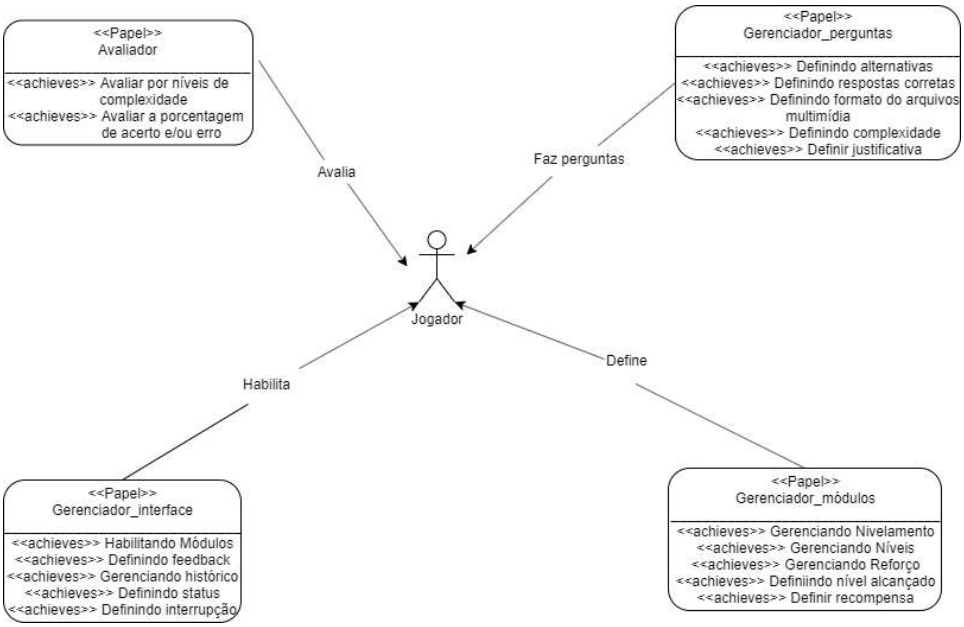
4. Modelo de Papeis - Iteração 2



5. Modelo de Interface com a Organização - Iteração 3



6. Modelo de Papeis - Iteração 3



ANEXO V – Grupo F (Agile O-MaSE) - Casos de Testes

Projeto de Arquitetura da metodologia ***Agile O-MaSE* – Grupos F**
Projeto de Desenvolvimento do Jogo Médico Educacional

Histórico de Revisões

Data	Versão	Descrição	Autor
30/10/2019	1.0	Criação do documento e dos Casos de Teste - Iteração 1.	[redacted]
05/12/2019	2.0	Criação dos Casos de Teste - Iteração 3.	[redacted]

1. **Casos de Teste - Iteração 1**

#	Ação	Resultado Esperado
1	Logar no jogo	Sistema deve apresentar página inicial.
2	Disponibilizar módulo	Sistema deve tornar acessível um módulo
3	Contabilizar acertos	Sistema deve direcionar o jogador para algum nível . Nível 5 (acertar todas as questões do nível 1 ao 4), Nível 4 (acertar todas as questões do nível 1 ao 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).
4	Justificar resposta	Sistema deve apresentar a resposta correta com uma explicação em caso de resposta errada.
5	Voltar em questões anteriores	Sistema deve exibir as perguntas já respondidas.
6	Preencher barra de status	Sistema deve acrescentar um quadrado verde para acertos e um vermelho para erros.
7	Interromper	Sistema deve voltar para a página inicial.

2. **Casos de Teste - Iteração 2**

#	Ação	Resultado Esperado
1	Acessar um nível	Sistema deve apresentar a primeira pergunta do nível.
2	Dar feedback	Sistema deve mostrar uma mensagem dando parabéns em caso do jogador passar de nível ou uma mensagem indicando reforço em caso de não conseguir passar de nível.
3	Gravar nível alcançado	Sistema deve manter o jogador no nível alcançado quando o mesmo sair e entrar no jogo.
4	Contabilizar erros e acertos	Sistema deve direcionar o jogador para o reforço, se o jogador errar mais de 50% da perguntas, e para o próximo nível, se o jogador obter 70% de acertos.
5	Recompensar	Sistema deve dar um equipamento de consultório médico para o jogador por nível alcançado.

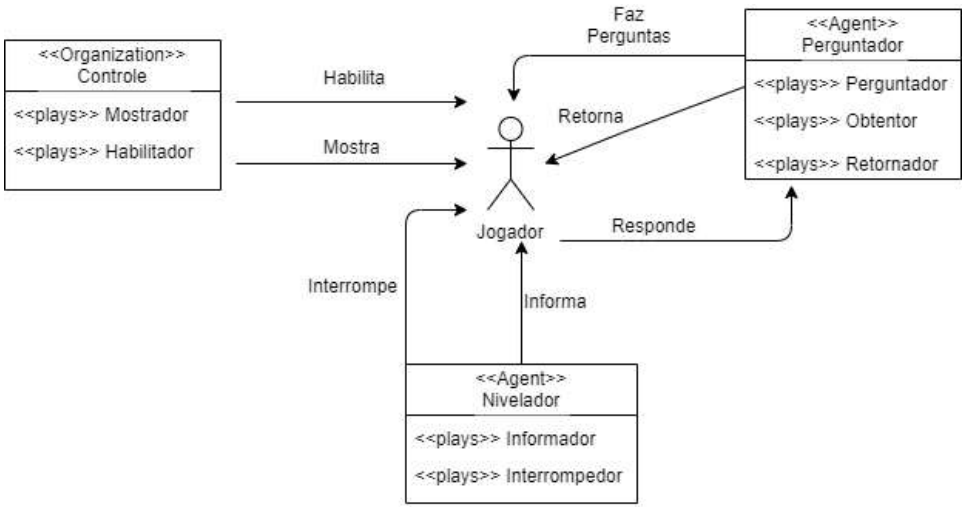
ANEXO W – Grupo F (Agile O-MaSE) - Modelo de Classes de Agentes e Modelo de Políticas

Projeto de Arquitetura da metodologia *Agile O-MaSE* – Grupos F
Projeto de Desenvolvimento do Jogo Médico Educacional

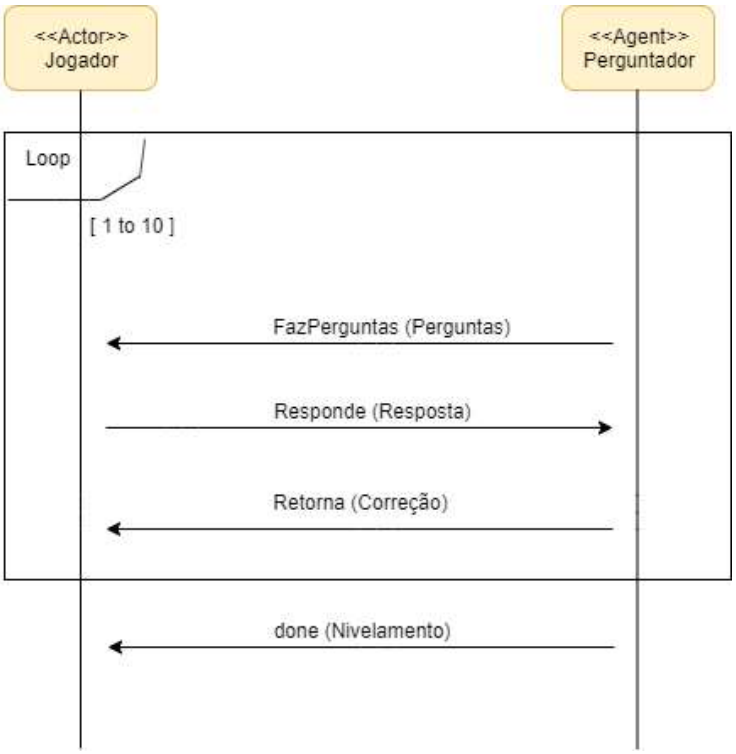
Histórico de Revisões

Data	Versão	Descrição	Autor
30/10/2019	1.0	Criação do documento, do Modelo de Classes de Agentes e do Modelo de Protocolos - Iteração 1.	[redacted]
05/12/2019	2.0	Atualização dos Modelo de Classes de Agentes e Modelo de Protocolos - Iteração 3.	[redacted]

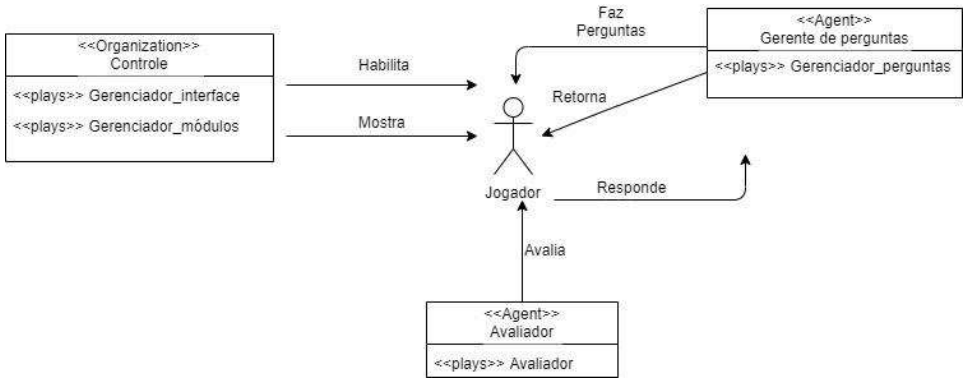
1. Modelo Classes de Agentes - Iteração 1



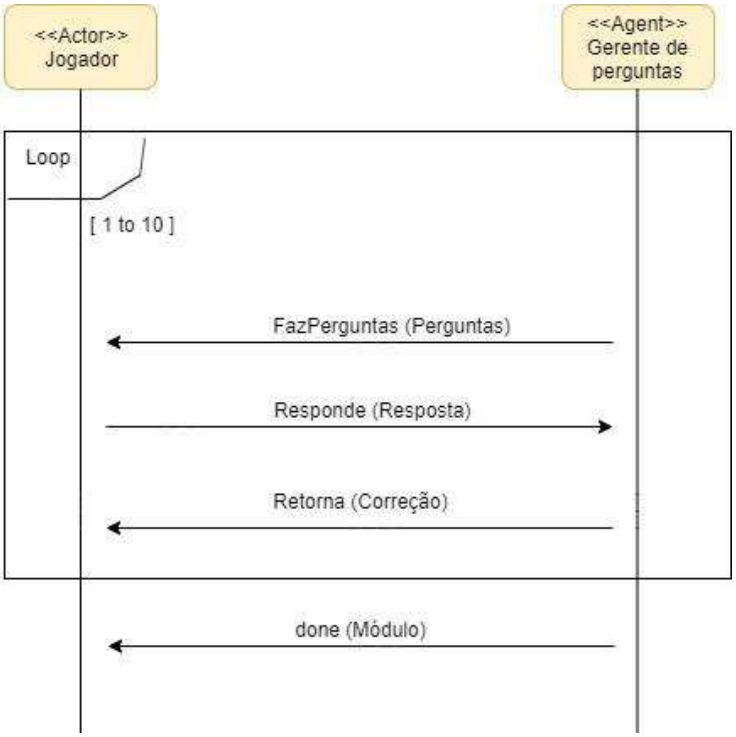
2. Modelo de Protocolos - Iteração 1



3. Modelo Classes de Agentes - Iteração 3



4. Modelo de Protocolos - Iteração 3



5. Políticas - Iteração 3

O jogador passa por um nivelamento antes de ter acesso aos níveis do sistema.

O sistema oferece o melhor nível para o jogador após o nivelamento.

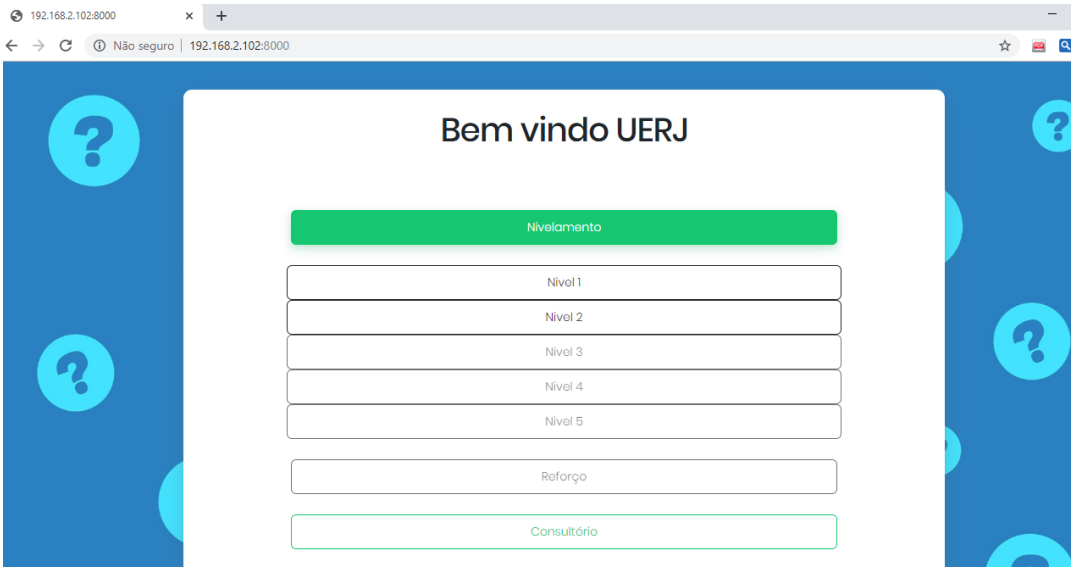
O jogador volta para o nivelamento caso suas respostas não atinjam a passagem de nível.

O jogador avança de nível caso suas respostas atinjam a passagem de nível.

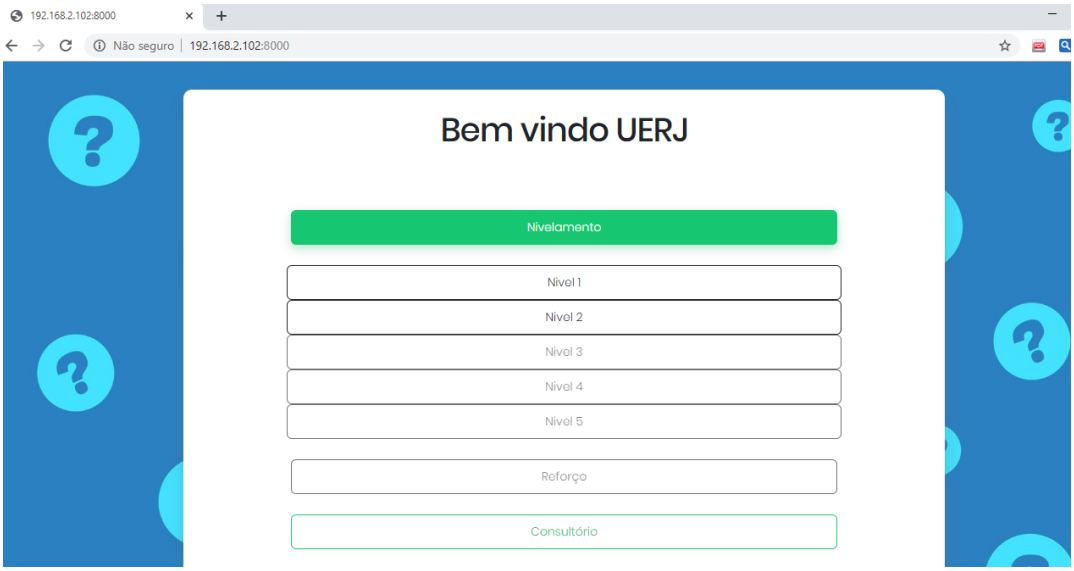
O sistema apresenta a resposta correta e sua respectiva explicação para o jogador após a tentativa de resposta de cada questão.

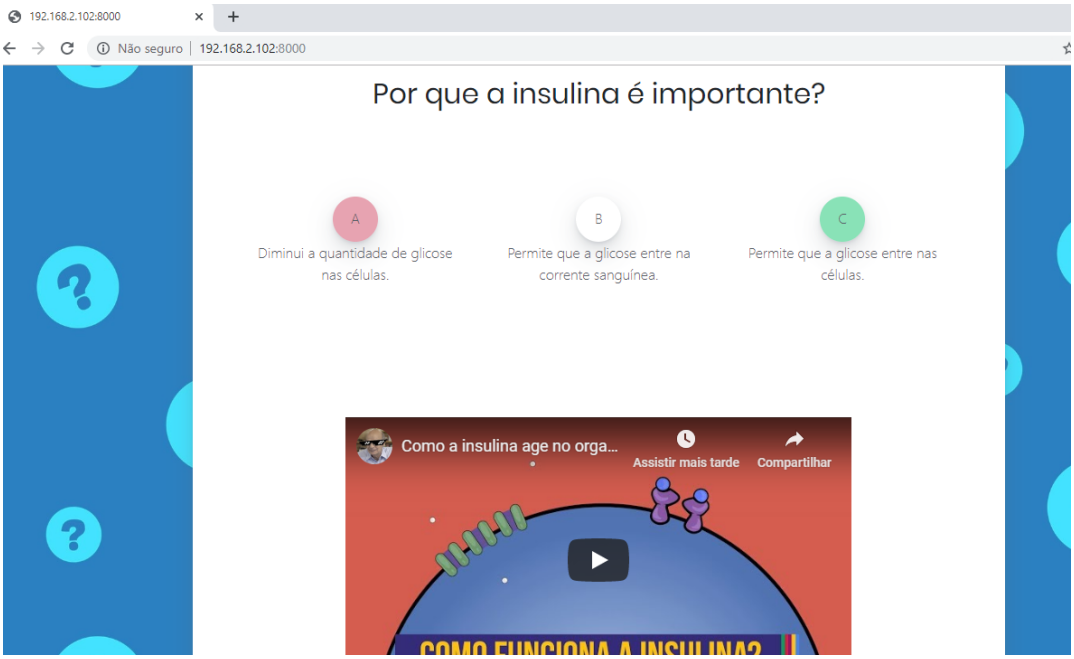
O jogador é premiado pelas respostas corretas com móveis para o consultório.

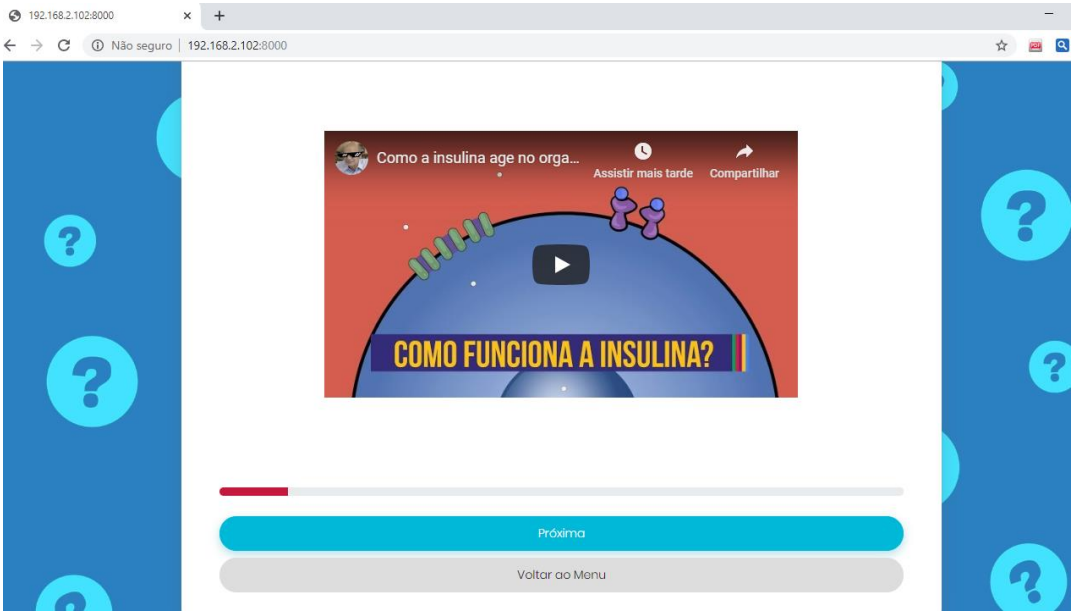
ANEXO X – Grupo F (Agile O-MaSE) - Jogo Desenvolvido

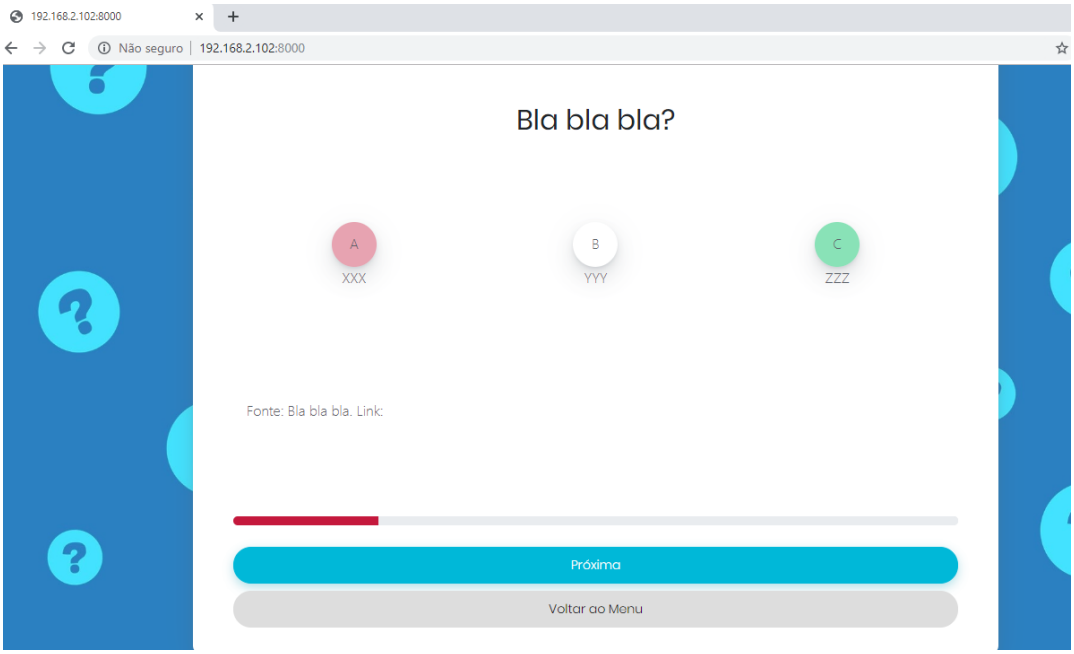



192.168.2.102:8000
Não seguro | 192.168.2.102:8000
Bem vindo ao MedEdu
uerj
Entrar
MedEdu - 2019


192.168.2.102:8000
Não seguro | 192.168.2.102:8000
Bem vindo UERJ
Nivelamento
Nivel 1
Nivel 2
Nivel 3
Nivel 4
Nivel 5
Reforço
Consultório


192.168.2.102:8000
Não seguro | 192.168.2.102:8000
Por que a insulina é importante?
A
Diminui a quantidade de glicose nas células.
B
Permite que a glicose entre na corrente sanguínea.
C
Permite que a glicose entre nas células.
Como a insulina age no orga...
Assistir mais tarde
Compartilhar


192.168.2.102:8000
Não seguro | 192.168.2.102:8000
Como a insulina age no orga...
Assistir mais tarde
Compartilhar
COMO FUNCIONA A INSULINA?
Próxima
Voltar ao Menu



Como controlar a vontade de comer doce?

A

Comer doces diet.

B

Equilibrar a alimentação e optar por doces mais naturais. Desde que controlado da para inserir doces na dieta.

C

Pode comer doces a vontade.

Próxima

Voltar ao Menu

Quais são as causas da embolia pulmonar?

A — Cirurgias extensas, câncer, gravidez, obesidade e traumas.

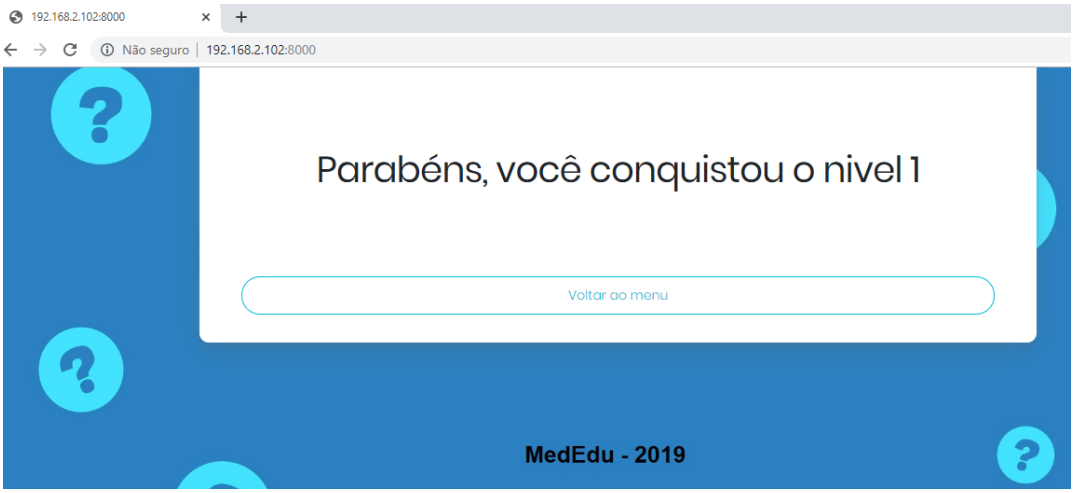
B — Pneumonia, doenças virais, cirurgias extensas.

C — Genética e pneumonia.

Embolia pulmonar é causada pela obstrução das artérias dos pulmões por coágulos. Fatores de risco são imobilidade prolongada, cirurgias, câncer, tabagismo, anticoncepcionais com estrógeno e reposição hormonal. Fonte: https://drauziovarella.uol.com.br/doencas-e-sintomas/embolia-pulmonar/

Finalizar


A
Cirurgias extensas, câncer, gravidez, obesidade e traumas.
B
Pneumonia, doenças virais, cirurgias extensas.
C
Genética e pneumonia.
Embolia pulmonar é causada pela obstrução das artérias dos pulmões por coágulos. Fatores de risco são imobilidade prolongada, cirurgias, câncer, tabagismo, anticoncepcionais com estrógeno e reposição hormonal.
Fonte: https://drauziovarella.uol.com.br/doencas-e-sintomas/embolia-pulmonar/
Finalizar
Voltar ao Menu


192.168.2.102:8000
Não seguro | 192.168.2.102:8000
Parabéns, você conquistou o nivel 1
Voltar ao menu
MedEdu - 2019