Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

GIRL: Uma Linguagem de Modelagem e Verificação de Requisitos Invariantes

Marzina Vidal Negreiros Bezerra

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Ciência da Computação

Franklin de Souza Ramalho e Tiago Lima Massoni
(Orientadores)

Campina Grande, Paraíba, Brasil

B469g Bezerra, Marzina Vidal Negreiros.
Girl: uma linguagem de modelagem e verificação de requisitos invariantes**.** / Marzina Vidal Negreiros Bezerra. ¯ Campina Grande, 2018.
254 f. Il.

Dissertação (Mestrado em Ciência da Computação) ¯ Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2018.
"Orientação: Prof. Dr. Tiago Lima Massoni _.
"Orientação: Prof. Dr. Franklin de Souza Ramalho _.

Referências.

1. Engenharia de Requisitos. 2. DSL. 3. Métodos Formais. 4. Verificação de Requisitos. I. Massoni, Tiago Lima. II. Ramalho, Franklin de Souza. III. Título.

CDU 004.41 (043)

FICHA CATALOGRÁFICA ELABORADA PELO BIBLIOTECÁRIO JESIEL FERREIRA GOMES CRB-15/256

"GIRL: UMA LINGUAGEM DE MODELAGEM E VERIFICAÇÃO DE REQUISITOS INVARIANTES"

MARZINA VIDAL NEGREIROS BEZERRA

DISSERTAÇÃO APROVADA EM 14/12/2018

TIAGO LIMA MASSONI, Dr., UFCG
Orientador(a)

FRANKLIN DE SOUZA RAMALHO, Dr., UFCG
Orientador(a)

WILKERSON DE LUCENA ANDRADE, Dr., UFCG
Examinador(a)

JULIANO MANABU IYODA, PhD, UFPE
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Requisitos precisos e consistentes promovem maior qualidade no *software*, reduzindo seu custo final. Estudos apontam que o uso de métodos formais na especificação dos requisitos promovem a detecção de ambiguidades e a verificação automática da consistência dos requisitos. Entretanto, geralmente dispensa-se o formalismo por sua difícil aplicação, possivelmente por causa de barreiras culturais ou da falta de habilidades de profissionais de software. Propomos, nessa dissertação de mestrado, uma solução mais convergente ou conciliadora para viabilizar a verificação formal, sem perder o foco nas habilidades comuns desses profissionais. Para isso, implementamos GIRL (*Graphical InvaRiant Language*), uma linguagem gráfica para representação de requisitos, onde o formalismo está embutido na automação e abstração, o que permite a verificação automática da consistência dos requisitos através da verificação de consistência da linguagem Alloy. GIRL é uma linguagem gráfica específica de domínio (DSL) para representação de restrições invariantes, que são requisitos estáticos ou estruturais.

Para avaliar a solução, procedemos um estudo empírico que segue a estratégia de pesquisa do tipo Tarefa de Julgamento, onde profissionais desenvolvedores de software, com diferentes níveis de experiência em Engenharia de Requisitos, utilizaram e avaliaram GIRL. Observamos que os sujeitos representaram corretamente os requisitos propostos no estudo, considerando úteis, tanto a linguagem, como a verificação dos requisitos com o *feedback* gráfico. Além disso, identificamos facilidade no aprendizado e utilização das estruturas mais simples e com conceitos semelhantes em engenharia de software e na teoria de conjuntos; já em estruturas mais complexas, cujos conceitos estão mais próximos à lógica, o nível de dificuldade foi maior. Também observamos que uma notação gráfica é mais agradável de utilizar e, sendo bem projetada, é mais fácil de usar, alcançando mais facilmente os objetivos para que foi criada. Percebemos então, fortes indicativos de que, representações gráficas que ocultam um formalismo e permitem a verificação automática de requisitos, como na linguagem GIRL, são meios eficazes para elevar a qualidade em requisitos de software.

Abstract

Precise and consistent requirements promote higher quality in the software, reducing its final cost. Studies show that the use of formal methods in requirements specifications promotes the detection of ambiguities and the automatic verification of the consistency of requirements. However, formalism is often dispensed with because of its difficult application, possibly because of cultural barriers or the lack of skills of software professionals. We propose, in this master's dissertation, a more convergent or conciliatory solution to enable formal verification, without losing focus on the common skills of these professionals. For this, we implemented GIRL (Graphical InvaRiant Language), a graphical language for requirements representation, where formalism is embedded in automation and abstraction, which allows the automatic verification of consistency of requirements through the Alloy language consistency checking . GIRL is a graphical domain-specific language (DSL) for representing invariant constraints, which are either static or structural requirements.

To evaluate the solution, we performed an empirical study that follows the research strategy of the Judgment Task type, where professional software developers, with different levels of experience in Requirements Engineering, used and evaluated GIRL. We observed that the subjects correctly represented the requirements proposed in the study, considering the language and the verification of the requirements with the useful graphic feedback. In addition, we identified ease in learning and using the simplest structures and with similar concepts in software engineering and set theory; however, with more complex structures, whose concepts are closer to the logic, the level of difficulty was greater. We also noted that a graphical notation is more enjoyable to use and, being well designed, is easier to use, and more easily reaching the objectives for which it was created. We then see strong indications that graphical representations that hide a formalism and allow the automatic verification of requirements, as in the GIRL language, are effective means to raise the quality in software requirements.

Agradecimentos

Agradeço a Deus,

que me fez e me deu capacidade para sonhar, pensar, iniciar e finalizar esse projeto!

Agradeço a meus pais, por me incentivarem desde a tenra infância!

Agradeço a meus filhos, motivo de alegria e paz, dois já voaram para seus ninhos!

Agradeço a meu primeiro netinho, que sem saber, me ajudou a correr mais!

Agradeço a meus orientadores, Tiago e Franklin, vocês sonham mais alto que eu!

Agradeço a Dayvson, pela ajuda nas ferramentas!

Agradeço a meus amigos e irmãos pelo amor de Cristo, sem suas orações eu não conseguiria!

Agradeço a meus amigos e colegas de trabalho, cada um tem um papel especial nessa pesquisa!

Agradeço aos que aceitaram o convite para tomar parte nessa aventura!

Agradeço ao apoio da pró-reitoria de pós-graduação que compreendeu as dificuldades que passei!

Agradeço ao meu gerente no STI, pelo apoio e pelo incentivo!

Agradeço às *Girls* do STI, por me animarem a chegar ao fim!

Agradeço aos que me ajudaram a preparar a rede da sala, a levar e trazer o computador durante as duas semanas do estudo, aos que deixaram o ambiente bem limpinho!

Deus abençoe a todos!

Conteúdo

Lista de Símbolos

BDD-based - Baseados em *Binary Decision Diagram*

CD - *Class Diagram*

EMF - *Eclipse Modeling Framework*

GIRL - *Graphical Invariant Language*

PoN - *Physics of Notation*

OCL - *Object Constraint Language*

OMG - *Object Management Group® (OMG®)*

SAL - *Source-code Annotation Language*

UML - *Unified Modeling Language*

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Requisitos precisos e consistentes produzem softwares de qualidade com um custo apropriado. Essa é a máxima que impulsiona a Engenharia de Requisitos. Muitos problemas em softwares surgem de falhas na maneira como as pessoas obtêm o conhecimento, documentam, concordam e modificam os requisitos do produto [38]. Requisitos incompletos, seguido de falhas de comunicação entre a equipe do projeto e o cliente e, mudança de alvos, são atualmente, os três problemas mais frequentes em engenharia de requisitos [9].

Minimizar esses problemas em requisitos passa necessariamente pela maximização da prevenção e da detecção de falhas em seus artefatos. Assim, prevenir ou encontrar inconsistências e ambiguidades nos requisitos, significa torná-los precisos e consistentes, e isso afeta positivamente o custo final do software. É mais barato corrigir um conceito do que corrigir o código em teste, ou pior, corrigir o código de um software em operação.

Para essa busca antecipada de falhas, uma solução é implantar o rigor do formalismo na representação dos requisitos, afastando as ambiguidades e possibilitando uma verificação automática da consistência dos requisitos. No entanto, os meios para aplicar tais formalismos frequentemente esbarram na cultura ou falta de habilidades de analistas de software. Esse contexto delimita o problema focalizado no presente estudo, o uso escasso de formalismos enquanto instrumento para alcançar requisitos precisos e consistentes.

Em seus escritos, Sommerville [31] e Wiegers e Beatty [39] registram pouca ênfase à especificação formal por considerá-la de difícil aplicação. Optam, então, pela melhoria da qualidade em requisitos descritos em linguagem natural. Outros autores defendem o formalismo matemático em requisitos como meio mais eficaz para elevar a qualidade do software

(Parnas [25], Jackson [13]). Os defensores da posição central apresentam argumentos em prol do formalismo sem perder o foco nos aspectos da prática, assim, consideram a necessidade de maior convergência ou cooperação entre as culturas do formalismo e da intuição [12].

Estudo sobre os mitos que envolvem a adoção de métodos formais destaca que, em ferramentas que escondem o formalismo por trás da automação e abstração, o uso é mais imediato, a exemplo de UML e da linguagem baseada em verificação SAL (*source-code annotation language*) [14].

Seguindo essa direção de conciliação entre formalismo *versus* intuição, propomos a utilização de uma linguagem que aproxime o formalismo do contexto de prática da maioria dos analistas de software. Uma linguagem que forneça uma representação formal por meio de uma notação visual, e assim, mais distanciada das notações simbólicas da matemática, mais propensas à rejeição. Isso requer uma linguagem específica de domínio, uma DSL, cujas abstrações gráficas sejam mais adequadas, ou seja, mais comuns e de mais fácil entendimento, do que as linguagens formais com notação textual ou simbólica.

A superioridade da informação gráfica sobre a textual é explicada pelas diferentes maneiras da mente humana processar essas informações [23]. O sistema de decodificação textual é serial, enquanto o sistema de decodificação visual é paralelo e muito mais poderoso (cerca de um quarto do cérebro é dedicado à visão). Por conseguinte, a adoção de uma notação visual, que seja bem projetada, promove ganho na aprendizagem, utilização e comunicabilidade do conhecimento representado [23]. Nessa perspectiva, percebemos que a notação visual da Teoria de Conjuntos na Matemática é uma opção simples que favorece a transparência semântica [23], por ser largamente conhecida e com isso, sua representação visual produz rápida inferência do seu significado.

Propomos então a linguagem GIRL, uma linguagem específica de domínio (DSL — sigla em inglês), para representar graficamente modelos formais de invariantes estáticas, sobre as entidades e seus relacionamentos, no contexto do software. Modelos em GIRL se constituem como artefatos de modelagem e verificação de requisitos, podendo ser utilizados em atividades a elicitação, análise, especificação (representação) e verificação. Nessa proposta não abarcamos a representação de requisitos de comportamento, apenas requisitos estáticos, ou estruturais, que devem permanecer inalterados em cada mudança do sistema.

GIRL é uma DSL do tipo externa com sintaxe visual baseada na linguagem declarativa Alloy [13]. Sendo assim, na verificação de consistência dos modelos em GIRL, empregamos o analisador da linguagem Alloy [13], semelhante ao que foi adotado em UML2Alloy [1] e CD2Alloy [21]. Então, além do editor gráfico, a linguagem GIRL disponibiliza um mecanismo de verificação que pressupõe uma transformação do modelo GIRL para o código Alloy, que é a entrada da análise de consistência do modelo do analisador de Alloy. Caso o modelo esteja consistente, instâncias de entidades e relacionamentos compatíveis com o modelo são apresentadas graficamente.

Uma vez implementados o editor da linguagem GIRL e as transformações de GIRL para Alloy (do modelo GIRL para modelo Alloy e do modelo Alloy para texto), realizamos um estudo empírico com profissionais, desenvolvedores de software de uma instituição pública de ensino superior. O estudo buscou avaliar a percepção da utilidade do uso de especificações formais com representação gráfica, usado o formalismo da linguagem GIRL, que abarca a verificação de consistência da linguagem Alloy [13]. A estratégia predominantemente utilizada no estudo foi o julgamento de tarefas (*judgment tasks*) [32], em que os sujeitos forneceram o julgamento sobre aspectos da linguagem GIRL como: entendimento das estruturas, registro das estruturas mais fáceis e mais difíceis de utilizar, utilidade do mecanismo de verificação e utilidade da linguagem.

O estudo teve como objetivo, avaliar a linguagem GIRL ao ser utilizada por desenvolvedores, nos seguintes aspectos: (1) facilidade de aprendizagem e utilização, (2) a precisão e cobertura dos requisitos representados, (3) a utilidade percebida na verificação automática de requisitos e (4) a efetividade da notação gráfica.

A aplicação do estudo empírico envolveu 11 profissionais de desenvolvimento de software. Eles receberam um treinamento na linguagem GIRL e, em seguida, modelaram oito requisitos invariantes de um projeto existente[1], enquanto isso, um observador tomava notas de comentários e dos passos para a construção do modelo. Por último, os sujeitos registraram seu julgamento sobre as estruturas da linguagem, o editor gráfico, o mecanismo de verificação e a utilidade de GIRL.

Os dados coletados e analisados forneceram indícios para avaliar se especificações for-

[1] Os requisitos do tipo invariantes utilizados foram extraídos de um projeto em andamento para desenvolvimento de um software de controle acadêmico numa instituição de ensino superior.

mais em notação gráfica são úteis para o bom entendimento dos requisitos e para a detecção de falhas. Trabalhamos com a hipótese de que analistas de requisitos consideram que, utilizar um formalismo embutido em uma notação gráfica (bem elaborada) provê o entendimento dos requisitos e a detecção de falhas antecipadas por meio da verificação automática dos modelos elaborados.

O estudo empírico evidenciou aspectos positivos e negativos sobre a percepção da utilidade do formalismo da linguagem GIRL, também delineou uma abordagem interessante para observações empíricas em notações visuais sendo avaliadas, na prática, por profissionais de software. Citamos a seguir, os principais resultados do estudo:

- O nível de dificuldade no aprendizado e utilização da linguagem GIRL foi considerado baixo para as estruturas mais simples e com conceitos e notações mais semelhantes aos conceitos da engenharia de software e da teoria de conjuntos. Em estruturas mais complexas e de conceitos mais próximos à lógica, como a quantificação e implicação, o nível de dificuldade foi maior.

- A representação dos requisitos em GIRL foi considerada correta em praticamente todos os requisitos, apesar das dificuldades relatadas nas estruturas mais complexas. Para um total de oitenta e oito invariantes representadas (8 invariantes para 11 participantes), apenas uma invariante de um participante foi representada equivocadamente.

- Todos os participantes consideraram que a verificação foi útil, com menções de satisfação ao poder testar requisitos.

- Efetividade da notação com o emprego de princípios *Physics of Notation* (PoN) recomendados em [23], nas seguintes evidências: (1) satisfação do uso de gráficos, (2) inferência do significado pelo desenho, (3) expressividade visual com variações consideráveis nos valores de variáveis visuais, (4) clareza nas diferenças dos símbolos visuais e (5) facilidade do desenho à mão dos elementos visuais. Por outro lado, houve um registro negativo na avaliação do participante quanto a uma variável de expressividade visual, a cor.

Vislumbramos três motivos que denotam contribuições relevantes do estudo, quanto ao seu objeto. Primeiro, em termos econômicos, a linguagem GIRL permite elaborar modelos

verificáveis, proporcionando assim, economia no custo do software com o uso de formalismos em requisitos. Segundo, o valor desmistificante de linguagens como GIRL, pois, com o uso do formalismo em gráficos, é possível driblar a resistência mítica dos desenvolvedores de software ao uso de técnicas matemáticas, ainda que a maioria dos currículos de formação desses profissionais contêm tópicos em matemática e métodos formais [14]. Por último, o valor da especialização da linguagem GIRL, que apresenta um formalismo mais específico, apesar de mais reduzido, do que os métodos formais que estão na sua base (Alloy) [14].

A relevância no método do estudo reside em dois aspectos, um deles é o mérito do envolvimento de profissionais na avaliação da linguagem GIRL. O outro é a ênfase na avaliação da utilidade e efetividade da notação visual da linguagem [35].

O restante do trabalho detalha o estudo empírico realizado e seus resultados. Iniciando com os conceitos teóricos necessários — requisitos de software, linguagem Alloy e DSLs — (capítulo 2), em seguida descreve a linguagem GIRL (capítulo 3) e seu mecanismo de verificação (capítulo 4). Prossegue com a descrição do estudo empírico (capítulo 5) e com a análise seguida da discussão dos dados coletados (capítulo 6). Logo depois, aborda os trabalhos relacionados ao estudo (capítulo 7). E por último, registra as considerações finais, além das contribuições e limitações do trabalho (capítulo 8).

Capítulo 2

Fundamentação Teórica

O presente estudo aborda conceitos da engenharia de software no tocante a requisitos, invariantes e linguagens específicas de domínio. Além desses, também enfoca um instrumento de especificação formal mais leve, a linguagem Alloy [13]. Estes conceitos são apresentados nas seções desse capítulo.

2.1 Requisitos de Software

Engenharia de Requisitos, é a área de conhecimento que envolve o conjunto de práticas em requisitos de software. Ela engloba os aspectos de desenvolvimento de requisitos e o gerenciamento de requisitos em projetos de desenvolvimento de software [38].

O objeto de estudo dessa área é o requisito, definido como sendo uma restrição de *stakeholders* associada apenas ao problema, ou apenas à solução do problema, ou a ambos [6]. Em outra definição, requisitos de software especificam o que deve ser implementado num software, descrevem o comportamento do sistema, as propriedades ou atributos do sistema, ou restrições no processo de desenvolvimento do sistema [38].

Requisitos de software são normalmente classificados em três níveis: Requisitos de Negócio, Requisitos de Usuário e Requisitos Funcionais. Se considerarmos o tipo de informação representada, os requisitos podem ser categorizados em: requisitos de negócio, regras de negócio, restrições, requisitos de interface externa, *features*, requisitos funcionais, requisitos não-funcionais, atributos de qualidade, requisitos de sistema e requisitos de usuários [38]. Desses, os requisitos funcionais, os requisitos de usuários e as regras de negócio, exprimem

informações sobre as entidades, relacionamentos, comportamento e restrições do problema que o software deve resolver.

Conceituando essas três últimas categorias de requisitos, definimos, primeiramente, um requisito funcional, como sendo uma descrição de um comportamento que um sistema exibirá sob condições específicas. Como demonstra o seguinte requisito: *O aplicativo (Serviços de Internet Banking) permite que o cliente inicie o processo de abertura de uma conta-corrente, inclusive com o envio das imagens dos seus documentos. O processo de abertura da conta fica pendente até que o cliente se dirija à agência por ele escolhida para efetuar a entrega dos documentos* [3].

Já um requisito de usuário é uma meta, ou tarefa que classes específicas de usuários devem ser capazes de executar com um sistema, ou um atributo de produto desejado [38]. Um exemplo de requisito de usuário está na seguinte declaração: *O aplicativo (Serviços de Retaguarda) permite que o funcionário responsável pela tesouraria autorize a abertura de terminais de caixa, deixando a operação pendente, aguardando autorização do outro usuário no seu próprio terminal* [3].

Por último, uma regra de negócio expressa uma política, diretriz, norma ou regulamento, que define ou restringe algum aspecto do negócio. Não é um requisito de software em si, mas a origem de vários tipos de requisitos de software [38]. Sendo assim, com a regra de negócio: *quando o valor de uma transação excede a alçada de um usuário, outro usuário deve autorizar essa transação*, derivamos o requisito funcional: *a solução deve requerer a autorização de outro usuário, quando o valor da transação excede a alçada do perfil do usuário que comandou a transação* [3].

Na perspectiva do sistema de informação, uma regra de negócio pode representar [38]:

- Fato — afirmação sobre o negócio que deve ser verdadeira, como na regra: *uma mensagem pode ser direcionada a um cliente específico (correntista, poupador ou beneficiário do INSS) ou a um grupo de clientes identificados por algum atributo em comum (tipo de conta, contratantes de cheque especial, com cheque especial vencido ou a vencer nos próximos dias, com aplicação financeira ou não etc)* [3];

- Restrição — afirmação que restringe a ação que o sistema ou o usuário pode realizar, como na seguinte regra: *uma transferência de recursos entre contas (corrente*

e de poupança) do Banco, em favor de uma conta cadastrada como conta favorita pelo cliente, para realização uma única vez (na data corrente ou agendada para uma data futura), ou de forma recorrente, com periodicidade mensal por até 12 meses ou semanal por até 52 semanas (adaptado de [3]);

- Inferência — afirmação do tipo "*if fato then novoFato*" estabelecendo que *novoFato* deve ser verdadeiro se *fato* também for verdadeiro. Um exemplo desse tipo de regra é a afirmação: *a solicitação de ativação de dispositivo móvel para uso do mobile banking só é permitida se a operação seja realizada em um computador pré-cadastrado pelo cliente* (adaptado de [3]);

- Ativação de ação — uma regra que aciona determinada atividade caso uma condição específica seja verdadeira, como na regra: *se o usuário ultrapassar o número máximo de tentativas sem sucesso para digitação da senha, o cartão de cliente deve ser bloqueado* (adaptado de [3]); e

- Cálculo — regra de transformação de dados usando fórmulas matemáticas ou algoritmos, como descrito em: *os valores limites para movimentações financeiras são definidos conforme o perfil do cliente: cliente que não possui cartão de segurança, ele não tem valor de limites definidos; cliente que possui cartão de segurança, a ele é atribuído valor de limites padrão para cartão; cliente que utiliza token, a ele é atribuído valores de limites padrão para token* (adaptado de [3]).

O termo restrição em Engenharia de Requisitos é aplicado tanto para o tipo de regra de negócio definido anteriormente, como para outros aspectos no desenvolvimento de software. Assim, ao estabelecer limitações de cronograma, equipe e orçamento, aplicamos restrições no projeto. Outro tipo de restrição ocorre quando se delimita as opções de escolha para o desenvolvedor no *design* e na construção de um produto.

No escopo deste trabalho abordamos requisitos que representam conhecimento sobre o domínio do problema a que o software se propõe solucionar, e que seja estático ou estrutural, ou seja, não descreve transformações, ações, interações ou cálculos aritméticos. São requisitos originados de regras de negócio que expressam fatos, restrições e inferências.

2.1.1 Processo de Desenvolvimento de Requisitos

Uma das principais práticas em Engenharia de Requisitos é o desenvolvimento de requisitos, ou o processo de produção de requisitos. O principal objetivo desse processo é estabelecer um conjunto de requisitos bons o suficiente, de forma que a equipe possa projetar e construir a próxima parte do produto de software, a um nível aceitável de risco [38].

Esse processo é iterativo e incremental, cujas principais etapas são: a elicitação, a análise, a especificação e a validação dos requisitos do software. Essas etapas abrangem, respectivamente, as atividades envolvidas na exploração, avaliação, documentação e validação dos requisitos de um produto [38]. Outros autores estabelecem que o processo de desenvolvimento de requisitos é composto das etapas de elicitação, modelagem, análises dos requisitos, validação e verificação [7]. Independente do ciclo de vida de desenvolvimento adotado, essas atividades sobre os requisitos serão realizadas, mas em diferentes momentos do projeto e em diferentes graus de profundidade ou detalhe.

A elicitação dos requisitos é considerada o coração do desenvolvimento de requisitos, pois nela são definidas, colaborativamente, as necessidades e as restrições dos *stakeholders* para o software. Elicitar envolve atividades de coleta, descoberta, extração e definição dos requisitos. Também é considerada a etapa mais crítica, desafiadora, passível de erros e intensa de comunicação no desenvolvimento de software.

Na atividade de análise, as necessidades e restrições coletadas são analisadas para chegar numa definição precisa dos requisitos. Um requisito é considerado preciso quando ele é: completo, correto, implementável, necessário, priorizado, não ambíguo e verificável. Assim, dado o seguinte requisito: *O aplicativo permite a realização de saque de conta-corrente e de conta de poupança mediante a leitura de um documento com código de barras, seguida da digitação da senha do cliente, aderente às normas do Banco Central, relativas ao combate à lavagem de dinheiro* [3]. Analisando esse requisito, descobre-se que está incompleto, pois falta definir quais *documentos com código de barras* são permitidos. Também está ambíguo, quando não define corretamente o que *deve estar aderente às normas*, podendo ser *a digitação da senha*, *a leitura de um documento com código de barras*, ou toda *a realização de saque de conta-corrente e de conta de poupança*.

Além disso, na análise procede-se a categorização dos requisitos e a representação de conjuntos de requisitos. Na análise, os requisitos são decompostos para um nível de detalhes

apropriado, e também priorizados para a implementação. Em certos momentos, é necessário interação com os *stakeholders* para negociações de priorizações ou de manutenção do escopo, ou mesmo, para retornar à elicitação em busca de esclarecimentos.

Após a análise, segue a especificação, ou seja, os requisitos são submetidos a uma documentação, que pode ser em linguagem natural ou gráfica por diagramas, ou em ambas. Durante a documentação, às vezes é necessário retomar a análise para esclarecer dúvidas, ou até mesmo retornar mais ainda para elicitar requisitos ausentes.

Por último, os requisitos são submetidos à validação dos *stakeholders* para definir se estão corretos e completos. Os erros encontrados podem disparar correções na especificação por falta de clareza, refazer análises, ou mesmo promover elicitações adicionais.

A incrementalidade no processo de desenvolvimento de requisitos prevê a construção do produto por partes, onde cada parte lida com um subconjunto de funcionalidades. Assim, para o primeiro subconjunto de funcionalidades na lista de prioridades, executam-se as atividades para os requisitos válidos que serão implementados. Após entregar os requisitos válidos do primeiro conjunto de funcionalidades, o ciclo se reinicia para os requisitos para o próximo da fila de prioridades de desenvolvimento.

A iteratividade refere-se ao alcance de um objetivo, no caso, requisitos corretos e completos, por tentativas sucessivas de refinamento. Ela é aplicada nos estágios iniciais no desenvolvimento do software para a definição do escopo do produto, como também para chegar ao conjunto de requisitos válidos para uma entrega (incremento) do produto.

2.1.2 Modelagem Visual de Requisitos

A atividade de documentar requisitos normalmente induz a uma ideia de escrever textos contendo os requisitos em linguagem natural. No entanto, ela é melhor definida como sendo a atividade de representar e armazenar o conhecimento envolvido nos requisitos [38].

Esse conhecimento envolvido em requisitos de software pode ser representado através de: (1) Linguagem natural, bem estruturada e redigida com base em regras preestabelecidas para evitar ambiguidades; (2) Modelos visuais que comunicam processos, estados do sistema e alterações entre eles, relacionamentos de dados, fluxos; ou (3) Especificações formais, que representam requisitos através de linguagens de especificação matematicamente precisas [38].

Uma vantagem da notação visual de determinados modelos é a diminuição das barreiras de linguagem e de vocabulários entre as pessoas envolvidas no projeto. Ela exerce papel importante na comunicação com usuários finais e clientes, pois, transmitem informação mais efetivamente para pessoal não-técnico do que um texto escrito [23]. Além disso, a representação visual é considerada mais efetiva porque é processada pelo sistema humano mais poderoso e altamente paralelo, o sistema visual. Cerca de um quarto do cérebro humano está dedicado à visão, mais do que todos os outros sentidos combinados [23].

Na prática, modelos visuais são úteis tanto para elaborar ou analisar requisitos, como também para projetar soluções de software. Na etapa de especificação de requisitos, ou mesmo na elicitação, é possível utilizar modelos para conferir mais precisão, completude e eliminar ao máximo a ambiguidade inerente à linguagem natural. Ademais, o próprio processo de criação do modelo também ajuda a revelar detalhes ausentes na elicitação, resultando em modelos que podem ser utilizados na comunicação dos requisitos aos *stakeholders* ou aos desenvolvedores [7].

Modelos visuais de requisitos podem ser representados em diagramas de fluxo de dados (DFDs) ou diagramas de entidades e relacionamentos (DERs), como demonstrado na Figura 2.1, onde (a) define o relacionamento em que uma (1) Conta possui vários (M) Clientes como titular, e (b) modela o fluxo de dados do procedimento de abertura de conta de um Cliente. Outras opções de modelos visuais são os diagramas de fluxo de processo, diagramas de transições de estados (DTEs), tabelas ou árvores de decisão, diagramas de casos de uso e diagramas de atividades [38].

Nem todo requisito do software precisa ser representado com modelos visuais. Para requisitos que envolvem mais riscos, complexidade, incertezas ou ambiguidades, o benefício da modelagem justifica o seu custo. A melhor estratégia é criar apenas o modelo necessário, quando for necessário e com o nível de detalhamento necessário, para que os *stakeholders* obtenham o entendimento adequado dos requisitos. Esse esforço adicional se justifica apenas quando o texto escrito, ou outra visão simplificada dos requisitos não forem suficientes para a sua efetiva comunicação e entendimento.

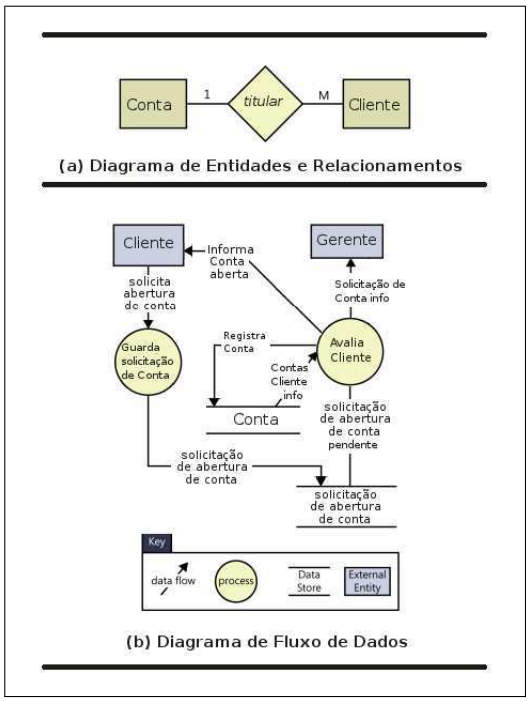

Figura 2.1: Exemplo de Modelos em Requisitos

2.1.3 Validação e Verificação de Requisitos

Idealmente, os requisitos devem ser representados, ou especificados, de tal forma que todos os envolvidos cheguem à mesma interpretação do requisito e que essa interpretação seja igual àquela que o autor tencionava comunicar. Dessa forma, a qualidade do conhecimento encerrado em requisitos especificados, individual ou em conjunto, diz respeito à boa comunicação daquilo que deve ser implementado no software, tanto para os *stakeholders*, como para os desenvolvedores [38].

Individualmente, um requisito de boa qualidade deve ser: completo (tem toda informação necessária para o entendimento e implementação), correto, implementável, necessário, priorizado, não ambíguo (provê interpretação única) e verificável (facilidade de verificar se está correto). Focalizando um conjunto de requisitos, ele é considerado de alta qualidade se for: completo (sem faltas e sem excessos), consistente (não há conflito entre os requisitos), modificável (dispõe histórico de mudanças, dependências entre requisitos e referência única) e rastreável (permite obter o encadeamento entre os requisitos, origem e artefatos derivados - requisitos, projeto, código e testes).

Para o alcance dessa qualidade é necessário validar e verificar os requisitos. A validação

de requisitos garante que os modelos e documentos expressam as necessidades dos *stakeholders*. Uma validação é uma avaliação subjetiva, realizada pelos *stakeholders* sobre requisitos descritos informalmente, ou não documentados.

Por outro lado, a verificação de requisitos pressupõem uma descrição formal dos requisitos, normalmente obtidas na validação. Com base nessas descrições, técnicas de verificação buscam provar que as especificações estão adequadas aos requisitos. Frequentemente a verificação busca checar se o modelo satisfaz a uma restrição.

São exemplos de técnicas de verificação o *model checking* e *model satisfiability* [7]. A primeira verifica modelos comportamentais em relação a propriedades de lógica temporal, sobre traços de execução (*trace execution*). A segunda técnica, *model satisfiability*, verifica se existem instanciações válidas de objetos do modelo e, se operações em modelos de objetos preservam invariantes. Alloy implementa essa última técnica em duas ações distintas, a simulação, que busca instâncias de estados ou execuções que satisfazem uma determinada propriedade, ou a checagem, que busca um contraexemplo, uma instância que viola uma dada propriedade. Nessas ações o espaço de busca, escopo, é restringido pelo usuário [13].

2.1.4 Invariantes em Requisitos

Representações formais adotam o conceito de invariante, que são propriedades que permanecem inalteradas quando transformações acontecem em instâncias do modelo. Diversas representações formais utilizam invariantes como na geometria e álgebra [28]. Nesse trabalho, utilizaremos o termo invariante no gênero feminino, por se tratar de uma propriedade ou restrição, ambas no feminino.

Na Ciência da Computação, uma invariante, de forma geral, é definida como sendo uma propriedade que se mantém verdadeira durante a execução de um programa. Também são utilizadas como recurso formal para a análise e verificação de programas [33]. Elas encerram uma expressão booleana composta por predicados atômicos.

Na teoria de Redes de Petri, invariantes representam os componentes conservativos (invariante de lugar) e os repetitivos (invariantes de transição) [28]. Em metodologias orientadas a objetos, a consistência dos dados é descrita por invariantes de objetos [19].

Parte dos trabalhos sobre invariantes, em engenharia de software, focalizam a verificação do software na etapa de codificação [19] , ou um pouco antes, na fase de projeto da solução

[2]. Alguns se debruçam na verificação de invariantes na fase de requisitos [15].

Invariantes em OCL

Neste estudo, aplicamos o conceito de invariantes em requisitos para expressar restrições em modelos. Conceito semelhante é utilizado para complementar as informações dos modelos UML - *Unified Modeling Language*) com a linguagem textual OCL (*Object Constraint Language*) [24].

A linguagem OCL (*Object Constraint Language*) implementa o conceito de invariantes para expressar restrições de integridade, no contexto de um tipo específico (*context type*). Uma invariante possui uma condição booleana (corpo da invariante) a ser verificada, e que deve ser satisfeita em todas as instâncias do *context type*. Com esse recurso, um projetista de software pode especificar as condições que um sistema deve atender [4].

As invariantes em OCL podem restringir o valor de objetos, ou expressar condições mais complexas, como limitações nos relacionamentos entre os objetos modelados em diagrama de classes ou de máquinas de estados de UML. Para que as restrições em OCL sejam verificadas, são necessárias ferramentas para assegurar que as expressões estão gramaticalmente corretas, além de checar se as expressões são representações válidas do domínio, ou seja, estão livres de incorreções, como inconsistências e redundâncias.

Exemplos dessas ferramentas são: USE (*UML-based Specification Environment*) e HOL-OCL, que utiliza Isabelle para prova de teoremas [4]. USE permite validar e verificar parte do modelo quanto às restrições descritas em OCL. Como resultado da verificação são exibidos gráficos que simulam situações permitidas pela especificação. HOL-OCL possibilita verificar se as restrições são satisfazíveis (*satisfiable*).

Outras ferramentas limitam a verificação restringindo o espaço de busca para garantir a terminação da verificação. Na ferramenta UML2Alloy, a verificação pressupõe uma tradução de UML/OCL para Alloy, que permite a verificação limitada [4].

Outras abordagens transferem a verificação para a fase de codificação, com geração de código a partir das restrições em OCL. Assim, a verificação é feita pela observação ou checagem do comportamento da execução do código gerado.

2.2 Linguagem Alloy

A linguagem Alloy [13] foi desenvolvida para fornecer uma notação precisa e expressiva, com base em um pequeno núcleo de conceitos simples e robustos. A meta é substituir a análise convencional da especificação formal, com base em prova de teoremas, por uma análise totalmente automática oferecendo *feedback* imediato. Além disso, Alloy encerra uma combinação dos quantificadores da lógica de primeira ordem com os operadores do cálculo relacional.

Alloy é uma derivação da linguagem Z, apresentando um ferramental menor e mais simples, com um mínimo de notação matemática e influências da notação de modelagem de objetos.

Um conjunto de sentenças em Alloy representa um modelo, uma abstração do mundo real. Esse modelo é composto por estruturas de Alloy que estão associadas à linguagem propriamente dita, ou ao seu mecanismo de análise, encerrando uma lógica específica para representar abstrações.

Essa lógica estabelece que todo modelo em Alloy é composto pelos átomos e pelas relações entre eles. Átomos são as entidades básicas do modelo e são considerados: indivisíveis, imutáveis e não interpretáveis (sem propriedades intrínsecas).

As relações, por sua vez, estabelecem relacionamentos entre os átomos e são constituídas por tuplas, que são uma sequência de átomos numa ordem fixa. O conceito de uma relação em Alloy é semelhante ao conceito de uma tabela, onde cada entrada é um átomo. A quantidade de colunas é chamada aridade da relação, que deve ser um ou mais. Já a quantidade de linhas, que pode ser zero, é o tamanho da relação, representando o número de tuplas. Relações com aridade, um, dois e três, são ditas, unária, binária e ternária, respectivamente. A relação chamada escalar possui apenas uma tupla de um átomo (uma linha e uma coluna).

As relações podem ser modificadas por meio de operadores de conjunto ou operadores relacionais. Os operadores de conjunto são: união (+), interseção (&), diferença (-), subconjunto (in), igualdade (=). Os operadores subconjunto e igualdade retornam um valor verdadeiro ou falso, os demais retornam uma relação com a mesma aridade das relações envolvidas.

Os operadores relacionais de Alloy transformam ou extraem informações das relações.

Considerando as relações A e B, os principais operadores sobre essas relações são: *arrow product* (A -> B) resulta numa nova relação que é combinação de todas as tuplas de A com as tuplas de B; *dot join* (A.B) resulta numa composição das tuplas de A e B, excluindo o último elemento da tupla de A e o primeiro da tupla de B, considerando apenas as tuplas que tem esses elementos iguais, as demais são descartadas; *transpose* (˜A), sendo A binária, reverte a ordem dos átomos de cada tupla; *transitive closure* (ˆA) representa a acessibilidade da relação A vista como um grafo; *reflexive transitive closure* (A) é a mesma relação ˆA unida à relação identidade (cada átomo relacionado a ele mesmo).

Além dessas operações sobre relações, é possível compor expressões entre a cardinalidade de relações (operador '#') e números inteiros. Nessas expressões aplicam-se, tanto os operadores de comparação (=, >, <, >= e =<), como os operadores aritméticos *plus*, *minus*, *mul*, *div* e *rem*, das respectivas operações de adição, subtração, multiplicação, divisão e resto.

Outras restrições aplicadas a relações podem ser expressas com operadores lógicos e quantificadores. Os operadores lógicos expressam: negação (*not* ou !), conjunção (*and* ou &&), disjunção (*or* ou ||), implicação (*implies* ou =>), alternativa da implicação (*else* ou ,) ou bi-implicação (*iff* ou <=>). Para quantificar elementos de conjuntos numa expressão, usam-se os quantificadores *all*, *some*, *no*, *lone* e *one*. Eles determinam, respectivamente, que essa expressão é verdadeira para todo, um ou mais, nenhum, um ou nenhum, e, apenas um, dos elementos quantificados.

As estruturas da linguagem Alloy que descrevem as relações são chamadas de assinaturas (*sign*). Uma assinatura pode conter restrições de cardinalidade ou de multiplicidade nas relações. As demais restrições do modelo são expressas com as estruturas: fato (*fact*), predicado (*pred*), função (*func*) e assertivas (*assert*).

Assinaturas (*sign*) introduzem no modelo uma relação de aridade um. Elas definem um conjunto de átomos, um subconjunto, uma extensão de uma assinatura ou uma assinatura abstrata. Os tipos da linguagem Alloy são associados implicitamente às assinaturas de topo, aquelas que não derivam de nenhuma outra assinatura por extensão (*extends*). As relações entre assinaturas são expressas nos campos (*fields*) da assinatura do primeiro átomo da tupla, onde o tipo dos demais átomos da tupla está definido no tipo do campo. Nessas relações, pode haver restrições de multiplicidade para os seus átomos, que pode ser: qualquer número (*set*), exatamente um (*one*), zero ou um (*lone*), um ou mais (*some*).

Para exemplificar o uso de alguns conceitos de Alloy apresentados até então, vamos considerar um requisito de uma aplicação bancária, em que: *uma conta deve ter um ou mais clientes como titular e um cliente deve ser titular de pelo menos uma conta*. Um código Alloy para esse requisito encontra-se no Código Fonte 2.1.

Nesse código, as linhas 4 a 8 definem as assinaturas $Conta$ e $Cliente$. Onde, a assinatura $Conta$, tem a relação descrita no *field*, com identificador $titular$, cujo tipo é a assinatura $Cliente$ (linha 7). Caso não houvesse nenhuma restrição nessa relação, então, para a seguinte instância do conjunto de contas $C : \{C1, C2\}$ e do conjunto de clientes $P : \{Joao, Maria, Jose\}$, podemos obter a seguinte instância da relação titular $T : \{\{C1, Joao\}, \{C1, Maria\}, \{C1, Jose\}\}$, com isso, todos os clientes são titulares da conta $C1$ e a conta $C2$ não tem nenhum titular.

Mas, existe a restrição de que a conta deve ter um ou mais titulares na multiplicidade $some$ para o tipo $Cliente$ no campo $titular$. Além da restrição de que um cliente deve ser titular de pelo menos uma conta, que está na expressão da linha 12, estabelecendo que, para todo (all) Cliente, a relação inversa (˜) de $titular$, deve ter um ou mais elementos. Assim, a instância T da relação titular sempre vai ter pelo menos uma tupla para cada conta, não havendo conta sem titular.

Código Fonte 2.1: Modulo_3.als

```
1  module ContaTitular
2
3  ------------ Signatures ------------
4  sig Cliente  {
5  }
6  sig Conta  {
7      titular: some Cliente
8  }
9
10 -------------- Facts ---------------
11 fact Conta{
12         all c : Cliente  |  some c.~titular
13 }
14
15 -------------- Commands ---------------
16 pred ContaTitular  [ ] {}
17 run ContaTitular for 3
```

As estruturas que definem as restrições do modelo são organizadas em parágrafos, podendo ser do tipo: fato (*fact*) - premissa do modelo; assertiva (*assert*) - implicação a ser checada pelo mecanismo de análise; predicado (*pred*) - restrição a ser utilizada em diferentes contextos; e função (*func*) - representando uma expressão reutilizável. Cada parágrafo possui uma coleção de restrições, como o fato $Conta$ contendo a restrição da linha 12.

Além das assinaturas e restrições, os comandos em Alloy servem para executar, ou seja, para analisar o modelo. Com o comando *run*, o analisador busca uma instância para o predicado referenciado. Já o comando *check* sinaliza para o analisador buscar um contra-exemplo da assertiva referenciada. Em ambos os comandos, pode ser definido um escopo explícito, ou espaço de busca. Esse escopo determina o limite máximo da quantidade de elementos das assinaturas a ser considerado na instância ou contra-exemplo. No Código Fonte 2.1 temos 3 assinaturas para o comando *run*, permitindo até 3 contas e até 3 clientes.

Nessa análise disparada pelos comandos mencionados, as restrições em Alloy são traduzidas para restrições booleanas (*boolean constraint*) que alimentam um solucionador da tecnologia SAT (*boolean satisfiability*). Com o comando *run* o analisador busca uma instância válida de um predicado referenciado, e, caso encontre, o modelo é considerado consistente. Já o comando *check*, dispara uma busca por uma instância que é um contraexemplo da assertiva (*assert*) referenciada. Caso seja encontrando um contraexemplo, o modelo tem inconsistências.

Para ambos os comandos, a instância encontrada corresponde a um conjunto finito de arranjos aleatórios entre átomos e relações do modelo. Cada arranjo é apresentado graficamente, exibindo seus átomos e suas relações, de acordo com o que foi definido no modelo. Também é possível avançar para o próximo arranjo da instância até chegar no último arranjo, não sendo possível voltar para arranjos anteriores. Esse recurso é muito útil para verificar se, além de consistente, o modelo está semanticamente correto ou completo.

Na Figura 2.2 temos quatro instanciações fornecidas pelo analisador Alloy para o Código Fonte 2.1, que representa o requisito: *uma conta deve ter um ou mais clientes como titular e um cliente deve ser titular de pelo menos uma conta*. Nessa figura, observamos os seguintes cenários: (a) uma conta com apenas um titular; (b) uma conta com dois titulares; (c) três contas com o mesmo cliente como titular; e (d) três contas em que duas tem dois titulares distintos, que são os titulares da terceira conta. Todos os cenários se encaixam na

especificação.

Com esse mecanismo de análise é possível verificar um modelo representado em Alloy, bem como detectar pontos para correção ou melhorias. Sendo assim, a atividade de verificação de requisitos pode ser efetuada por meio da linguagem Alloy, provendo automatização e um considerável rigor formal a essa atividade.

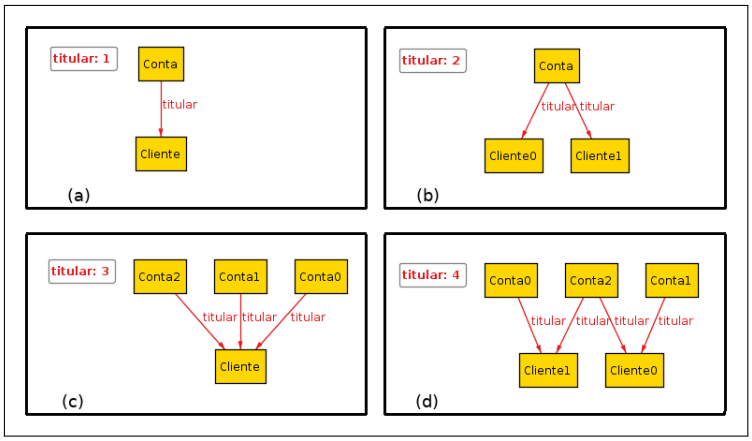

Figura 2.2: Instâncias da Verificação do Código 2.1

2.3 Linguagens Específicas de Domínio - DSL

DSLs (sigla em inglês para Linguagens Específicas de Domínio) são linguagens especializadas para certas classes de problemas - domínios. Baseia-se em abstrações mais adequadas para os conceitos do domínio da linguagem. Então, uma DSL, expressa as abstrações específicas do seu domínio por meio de notação textual, tabelas, símbolos ou gráficos [37].

Para dar "vida" às abstrações expressas na linguagem, é necessário um mecanismo para executar programas escritos na DSL. O mecanismo de execução pode ser construído com base em tradução (geração ou compilação) ou interpretação. Na primeira, é construído um novo mecanismo para a DSL. Na segunda, é feita uma tradução da DSL para uma linguagem e utilizar o mecanismo de execução dessa segunda linguagem, que normalmente é uma linguagem de propósito geral.

Como em toda linguagem de programação, uma DSL contém uma sintaxe abstrata (estrutura gramatical dos conceitos relevantes da linguagem), uma sintaxe concreta (notação), uma semântica estática (restrições e regras estruturais sintáticas), semântica de execução (signifi-

cado do programa ao ser executado) e o mecanismo de execução (mencionado no parágrafo anterior).

Uma DSL pode ser classificada como interna ou externa, caso seja, respectivamente, embutida na linguagem principal ou diferente da linguagem principal. DSLs internas são uma maneira particular de utilizar uma linguagem de propósito geral, um exemplo é a linguagem Ruby on Rails [37]. As externas permitem especificações mais livres e pouca ou nenhuma obrigação com padrões, gerando artefatos mais compactos e úteis, a exemplo de: SQL, CSS e HTML [8].

Em desenvolvimento de software direcionado por modelos (MDD ou MDSD em inglês), utilizam-se DSLs para criar representações de aspectos específicos do software, que sejam formais e processáveis por meio de ferramentas. Em seguida as representações são transformadas em código executável de linguagens de propósito geral. Uma modelagem pode gerar modelos descritivos, se representam um sistema existente, ou modelos prescritivos, que são usados para construir um dado sistema. Esses últimos devem ser mais rigorosos, formais, completos e consistentes. São a base do MDSD e de suas respectivas DSLs. Essas DSLs também podem utilizar notações que tornam as abstrações acessíveis a não-programadores [37].

O uso de DSLs favorece atividades de validação e verificação, por sua habilidade de capturar e expressar o essencial para o domínio, evitando detalhes irrelevantes de implementação. Algumas DSLs são construídas especificamente para permitir análises formais (matemáticas) complexas. Outro benefício de DSLs é preservar o foco apenas em aspectos concernentes ao domínio, pensando e comunicando apenas o essencial, relegando detalhes de implementação (acidental) para o mecanismo de execução [37].

As desvantagens de DSLs concentram-se no custo de construir uma linguagem, seja por causa do tempo gasto e da aquisição de habilidades, ou pela perda de investimento anterior. E além disso, aspectos como resistência à mudança de paradigmas, ou, por outro lado, o excesso, com a criação indevida de DSLs já disponíveis [37].

DSLs tem sido aplicadas em [8]: utilitário para desenvolvedores (*Jnario*[1]); geração de arquitetura de softwares ou plataformas complexas (AUTOSAR[2]); geração de aplicações

[1] Jnario: DSL para simplificar a definição de cenários executáveis para aplicativos Java
[2] O AUTOSAR é um padrão arquitetural para desenvolvimento de software automotivo.

inteiras (*mbeddr*[3]); domínios de aplicações a serem utilizadas por especialistas (aplicações de monitoramento em saúde); Engenharia de Requisitos (VERDE[4]); análises, checagens e provas matemáticas.

Comumente, na implementação DSLs externas são necessárias transformações entre os artefatos da linguagem. Transformações podem ocorrer entre metamodelos (AST - Abstract Syntax Tree) ou entre metamodelos e código textual, a geração de código. Algumas padronizações estão disponíveis para essas transformações: entre metamodelos - QVT (Meta Object Facility (MOF) Query/View/Transformation Specification[5]); geração de código - metamodelo para texto (MOFM2T - MOF Model to Text Transformation Language[6]), e são implementadas, respectivamente, pelas ferramentas: ACCELEO[7] e QVTOperational[8].

[3]mbeddr: Embedded C estende a linguagem de programação C com conceitos para software embarcado, incluindo máquinas de estado, tarefas e quantidades físicas.

[4]VERification-oriented & component-based model Driven Engineering for real-time embedded systems - https://itea3.org/project/verde.html

[5]https://www.omg.org/spec/QVT/About-QVT/

[6]https://www.omg.org/spec/MOFM2T/

[7]https://www.eclipse.org/acceleo/

[8]https://projects.eclipse.org/projects/modeling.mmt.qvt-oml

Capítulo 3

Linguagem GIRL

No estudo em questão foi desenvolvida a linguagem GIRL (**G**raphical **I**nva**R**iant **L**anguage), para a elaboração de modelos formais de invariantes, que estabelecem restrições sobre entidades e relacionamentos do software, em atividades de elicitação, análise e verificação de requisitos.

Modelos em GIRL são artefatos de modelagem e de verificação de requisitos e podem ser utilizados independente da metodologia de desenvolvimento adotada. Como toda atividade de modelagem tem um custo adicional, a decisão do que modelar em GIRL fica a cargo do analista de requisitos, junto à gerência do projeto, dependendo de fatores como a criticidade do software e dos requisitos, orçamento, dentre outros.

Neste capítulo descrevemos as estruturas da linguagem GIRL. No próximo abordamos a verificação das invariantes da linguagem.

3.1 Visão Geral

GIRL é uma DSL do tipo externa, com sintaxe visual e baseada na linguagem declarativa Alloy [13], que tem sintaxe textual. Para a execução de análises de um modelo em GIRL, utilizamos o mecanismo de análise da linguagem Alloy, que provê *feedback* visual.

Na definição da notação visual da linguagem GIRL, empregamos a notação da Teoria dos Conjuntos da Matemática, utilizando a representação gráfica dos Diagramas de Venn para conjuntos e operações de conjuntos [20]. Fornecemos em GIRL, notação para conceitos da lógica matemática, mais especificamente, a lógica de primeira ordem, além do cálculo

relacional e da modelagem de objetos, todos presentes em Alloy.

Assim, adotamos símbolos gráficos para os conceitos de: conjuntos (e conjuntos derivados), relações, quantificadores ("nenhum", "um ou nenhum" e "apenas um" além dos tradicionais, "para todo" e "existe"), operações lógicas (conjunção, disjunção, negação e implicação), operação fechamento transitivo sobre relações, multiplicidades nas relações, cardinalidade de relações e operações de comparação entre inteiros [13].

Uma invariante em GIRL é a estrutura básica da linguagem, que serve para organizar e acomodar as restrições sobre o domínio modelado. As demais estruturas servem para expressar as restrições do modelo e estão divididas em dois grupos, quanto a sua composição: as estruturas primárias *Entity* e *Integer*, e as estruturas compostas *Containment, Set Operation, Cardinality, Relational Operation, Logical Operation, Quantification, Relationship, Relationship Operation* e *Implication*.

Percebe-se essa distinção no metamodelo de GIRL da Figura 3.1. Nesse diagrama de classes que modela as estruturas de GIRL e seus relacionamentos, é possível observar que as estruturas *Entity* e *Integer* não têm outras estruturas obrigatórias em sua composição. Em contrapartida, as demais estruturas têm outras na sua composição ou em seus relacionamentos obrigatórios, como a *Relationship*, que requer duas *Operations* nos relacionamentos *source* e *target*.

Dentre as estruturas primárias e compostas em GIRL, apenas seis são significativas, ou seja, que expressam verdades por si só, são elas: *Entity*, *Containment*, *Relational Operation*, *Logical Operation*, *Relationship* e *Implication*. *Entity* e *Relationship*, além de representarem a estruturação das entidades do domínio, também podem conter restrições implícitas nos conceitos de cardinalidade igual a um, para entidades unitárias, e nas multiplicidades das relações. As restantes só têm razão de existir se estiverem associadas a outra estrutura.

Com base nas definições do metamodelo de GIRL, temos que, um modelo (*Model*) na linguagem GIRL é composto por um conjunto de invariantes (*Invariant*) que contém elementos de invariante (*InvariantElement*). Um *InvariantElement* pode ser do tipo:

- *Entity* — uma entidade do modelo;
- *Relationship* — um relacionamento entre um elemento de origem e um elemento destino que é uma multiplicidade sobre: entidades, operações de conjunto do modelo,

quantificação, ou operação de relacionamento;

- *Integer* — um valor inteiro; ou
- *Operation* — operação sobre as estruturas da linguagem, que pode ser um dos seguintes tipos:
  - *Cardinality* — a cardinalidade, que é a quantidade de elementos de uma entidade, ou de uma operação de conjunto;
  - *Containment* — determina que uma entidade está contida em outra;
  - *Implication* — uma implicação entre operações que resultam em valor booleano, ou entre relacionamentos;
  - *LogicalOperation* — operação lógica entre duas ou mais operações, para a negação, ou entre duas ou mais operações para a conjunção e disjunção;
  - *SetOperation* — operação de união, interseção, ou complemento entre entidades ou operações de conjunto;
  - *Quantification* — uma quantificação sobre uma entidade ou operação de conjunto;
  - *RelationalOperation* — operação relacional entre dois elementos comparando inteiros ou cardinalidades;
  - *MultiplicityOperation* — define a multiplicidade dos componentes de uma Relationship; ou
  - *RelationshipOperation*[1] — uma operação sobre um relacionamento que resulta em outro relacionamento.

Para melhor compreensão das regras de formação das estruturas da linguagem, classificamos suas estruturas nos seguintes tipos:

- *EntityType* — Tipo entidade: representa uma entidade ou uma parte de uma entidade do modelo, podendo ser: *Entity, SetOperation, Quantification* e *RelationshipOperation*;

[1] A estrutura *Relationship Operation* não entrou no escopo do estudo empírico em questão.

- *BooleanType* — Tipo booleano: resulta num valor booleano verdadeiro ou falso, sendo um dos tipos: *Containment, LogicalOperation, Implication* e *RelationalOperation*.

- *IntegerType* — Tipo inteiro: representa valores inteiros que podem ser usados em operações relacionais (*RelationalOperation*), que são: *Integer* e *Cardinality*.

- *RelationshipType* — Tipo relacionamento: representa relações entre os tipos entidade do modelo, que pode ser *Relationship* ou *RelationshipOperation*.

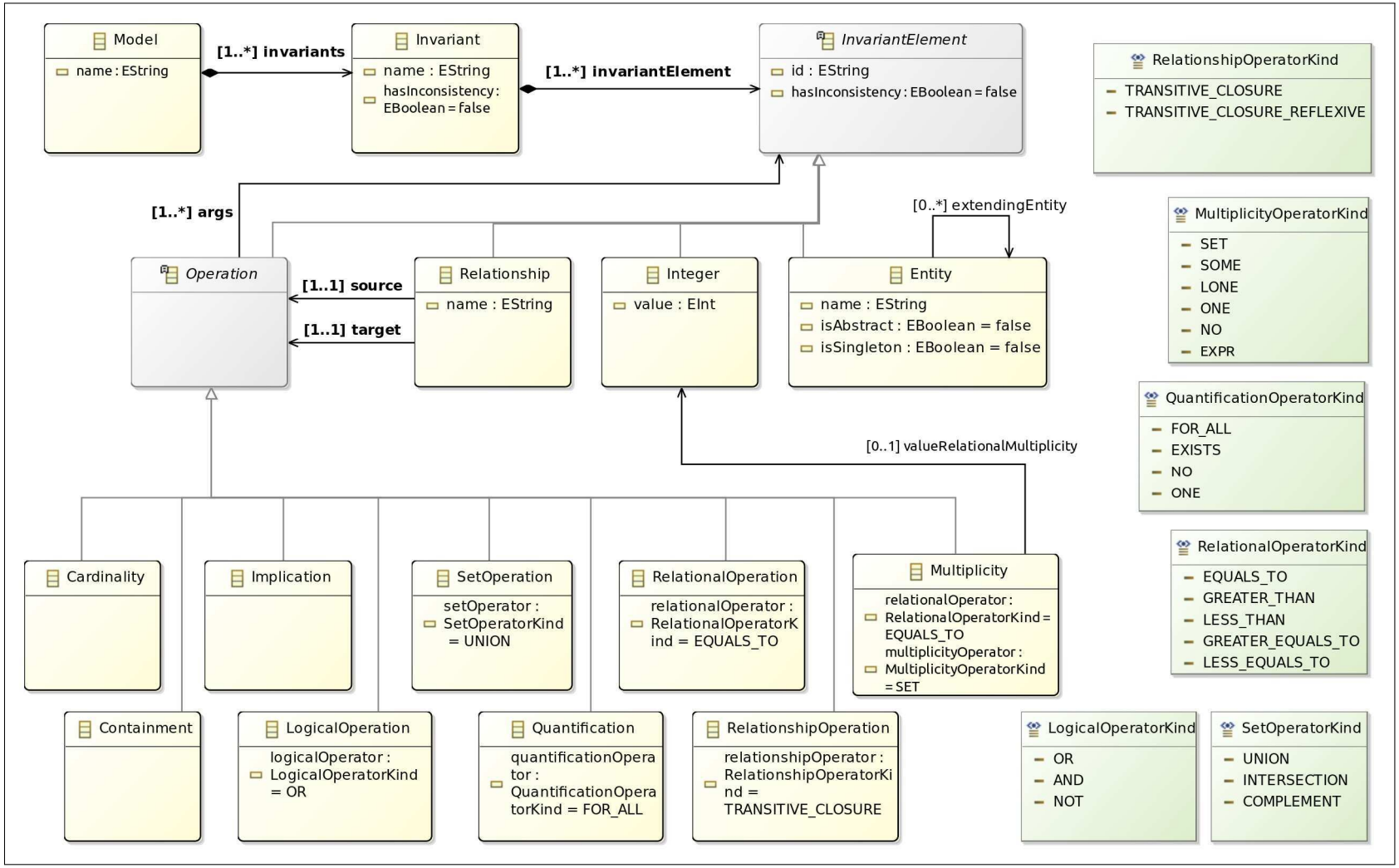

Figura 3.1: Metamodelo *Ecore* da Linguagem GIRL

A sintaxe gráfica, a semântica, as regras de composição e um exemplo de uso (não necessariamente numa única invariante) de cada estrutura da linguagem GIRL estão descritas nas próximas seções deste capítulo.

Na descrição de cada estrutura da linguagem, apresentamos a semântica, a sintaxe e os seus componentes, caso seja uma estrutura composta. Como GIRL é uma linguagem gráfica, sua sintaxe deve ser descrita em termos das variáveis para notação visual [23], que são: forma, cor, posição, textura, brilho, orientação. Também explicamos as razões para a escolha da notação visual de algumas das estruturas, para aplicar o princípio da transparência semântica (*semantic transparency*), em que a representação visual sugere seu significado [23]. Adicionalmente, apresentamos o procedimento para a construção das estruturas, ou seja, como desenhar a estrutura no editor de GIRL, que possui uma área de edição e uma palheta de elementos para a criação. Essa informação demonstra aspectos da usabilidade do editor gráfico. Além disso, acrescentamos um exemplo de uso de uma ou mais estruturas.

3.2 Estruturas da Linguagem

Nesta seção apresentamos as estruturas da linguagem GIRL, exibindo de imediato a figura com o símbolo gráfico para a estrutura, para em seguida expor o texto com a sua descrição.

3.2.1 Invariant

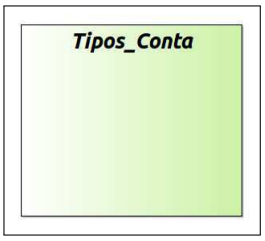

Figura 3.2: Invariant

Uma *Invariant* em GIRL representa um conjunto de restrições, em que cada restrição sempre deve ser verdade no escopo do modelo. Esse conceito é derivado do conceito de invariante em OCL *(Object Constraint Language)* [4], que apresenta invariantes como restrições associadas a determinado contexto. Uma invariante funciona como um *depósito* (*container*)

de uma ou mais restrições, que são operações do tipo booleano, entidades e relacionamentos. Qualquer expressão em GIRL deve ser criada dentro de uma invariante.

A invariante é representada por um quadrilátero redimensionável com identificador textual, originalmente preenchida com um degradê da cor branca até a cor verde (Figura 3.2). Optamos pela cor de fundo para diferenciar os conteúdos das estruturas com preenchimento em branco que são apresentadas na invariante.

Os componentes de uma *Invariant* são do tipo *InvariantElements*, como descrito no metamodelo da Figura 3.1. Na Figura 3.2 temos um exemplo de uma invariante para restrições dos tipos de contas em uma aplicação bancária.

3.2.2 Entity

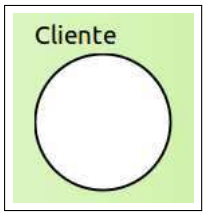

Figura 3.3: Entity

Uma *Entity* representa uma entidade, que, em requisitos de software, corresponde a um conjunto, ou uma classe. Entidades podem ter zero ou mais elementos, que seriam instâncias dessa entidade. Representamos uma *Entity* por um círculo branco com borda sólida na cor preta (Figura 3.3), numa analogia ao símbolo de conjunto nos diagramas de Venn [20]. A Figura 3.3 apresenta a entidade Cliente dentro dos requisitos de uma aplicação bancária.

Uma entidade pode ter uma associação com entidades estendidas (filhas). Duas ou mais entidades com o mesmo identificador em qualquer invariante, são consideradas a mesma entidade. Opcionalmente, uma entidade pode estender outra entidade. No entanto, só pode estender uma entidade que não estenda a si mesma direta ou indiretamente.

Entity - Abstract

Uma Entity pode ser abstrata, significando que é um invólucro (*container*) conceitual de outras entidades. O esperado é que uma entidade abstrata tenha entidades filhas (que sejam

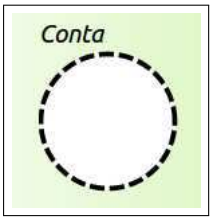

Figura 3.4: Entity Abstract

extensão dela), pois, a quantidade de elementos de uma entidade abstrata é o somatório da quantidade de elementos das suas entidades filha.

Representamos entidades abstratas por um círculo branco com borda tracejada na cor preta (Figura 3.4). A partir da segunda entidade estendida, a entidade abstrata passa a exibir partições (ou fatias) na quantidade de entidades estendidas até a décima. Para mais de dez entidades estendidas, mantém-se as dez partições e é exibida a quantidade de entidades estendidas no identificador da entidade abstrata. Além disso, caso todas as entidades estendidas sejam singleton, a entidade abstrata assume a cor cinza. A Figura 3.6 contém dois exemplos de entidades abstratas particionadas, Conta e Situação_Conta, sendo essa última com a cor cinza já que todas as entidades filhas são *singleton*.

Havendo uma entidade abstrata *A* no modelo e, existindo outra entidade *A*, definida como não abstrata, esta segunda entidade também será considerada abstrata. Uma vez que uma entidade é definida como sendo abstrata em alguma invariante, será considerada abstrata nas demais.

Entity - Singleton

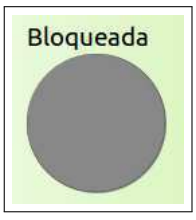

Figura 3.5: Entity Singleton

Um tipo particular de entidade pode ter apenas um elemento, sendo um conjunto unitário. A esse tipo de entidade chamamos *singleton*, que deve ser representada com um círculo na

cor cinza com borda sólida na cor preta. Na Figura 3.5, a entidade Bloqueada é uma entidade *singleton*.

Caso exista uma entidade *singleton B* e, havendo em qualquer invariante do modelo outra entidade não *singleton* com o mesmo identificador *B*, então essa segunda entidade também será considerada *singleton*. Uma vez que uma entidade é definida como sendo singleton em alguma invariante, será considerada singleton nas demais.

Uma entidade abstrata não pode ser definida como singleton. Também não faz muito sentido uma entidade *singleton* ter entidades filhas, só se fosse apenas uma e que também fosse *singleton*.

Na invariante Situação_Conta da Figura 3.6, as entidades: Aberta, Bloqueada e Fechada, são exemplos de entidades *singleton*, pois no modelo não pode haver mais de uma instância para essas entidades. Ou seja, não exite uma situação Aberta-1 e Aberta-2 para uma conta bancária.

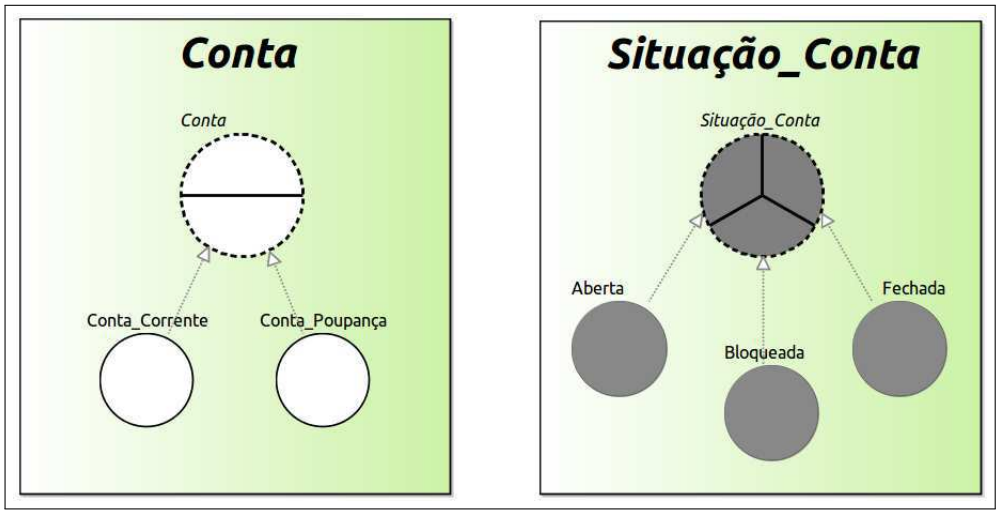

Figura 3.6: Exemplo do uso de Invariant e Entity - Aplicação Bancária

Estendendo Entidades (Extends)

Quando uma entidade estende uma outra, abstrata ou não, significa tornar-se um subconjunto da entidade que ela estende. Quando duas ou mais entidades estendem uma entidade elas são mutuamente disjuntas, ou seja, a interseção entre elas é vazia.

Representamos o conceito de extensão (*extends*) por uma seta tracejada na cor cinza,

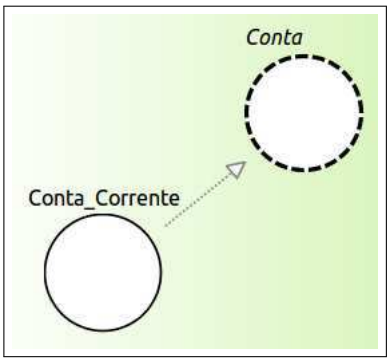

Figura 3.7: Entity Extends

com ponta a cor branca, semelhante à herança de classes em UML. A Figura 3.7 demonstra a entidade Conta_Corrente estendendo a entidade Conta. A seta parte da entidade filha Conta_Corrente, com a ponta chegando na entidade mãe Conta. A seta da extensão deve tocar a borda das duas entidades em qualquer posição. Uma entidade só pode estender uma única entidade, assim, não é possível haver herança múltipla.

Quando uma entidade abstrata é mãe de mais de uma entidade, então a partir da segunda entidade estendida, ela passa a exibir partições (fatias) na quantidade de entidades filhas. Em mais de dez entidades filhas, se mantém as dez fatias, sendo exibida a quantidade de entidades estendidas no identificador da entidade abstrata.

Outra particularidade da notação acontece quando uma entidade abstrata, cujas entidades estendidas são todas singleton, ela assume a cor cinza configurando-se assim numa *enumeration*, semelhante ao conceito de *enumeration* na programação. Na Figura 3.6, a entidade abstrata Situação_Conta é um exemplo de *enumeration*.

3.2.3 Relationship

As entidades de um dado domínio se relacionam entre si e cada relação possui um identificador. No contexto da linguagem GIRL, *Relationship* é uma relação entre dois elementos do tipo *EntityType*, sendo um a origem, ou o detentor da relação, e o outro o alvo da relação. Essas relações contêm uma multiplicidade para cada elemento da relação. São permitidas apenas relações binárias (entre dois elementos) e podem ocorrer apenas entre estruturas do tipo *EntityType*: entidades (*entity*), operações de conjunto (*setOperation*), quantificadores (*Quantification*) e operações em relações (*RelationshipOperation*). Uma relação pode acon-

tecer entre um mesmo elemento, ou seja, ela pode ser reflexiva.

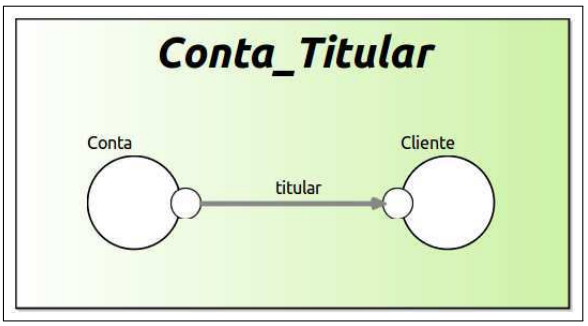

Figura 3.8: Relationship

Uma relationship é representada por uma seta na cor cinza direcionada do elemento de origem da relação (esquerda) para o elemento alvo (direita). A Figura 3.8 apresenta a relação titular entre Conta e Cliente. Cada elemento da relação tem uma multiplicidade que é um círculo menor posicionado entre o elemento e a seta da relação. A seta toca as multiplicidades desses elementos e não os elementos em si. *Relationships* possuem as seguintes restrições:

- Dado um modelo com tem uma relação desenhada inicialmente pela seta $s1$ e, uma segunda seta $s2$ que referencia a mesma relação da seta $s1$, então estas duas setas setas devem estar entre elementos de origem e alvo, que sejam baseados na mesma entidade, ou na entidade filha, respectivamente. Caso contrário, ocorrerá uma inconsistência na verificação.

- Um elemento de origem ou alvo da seta de uma relação pode ser aproveitado como elemento de outra relação, mas a multiplicidade de cada elemento compartilhado é individual para cada relação .

- A multiplicidade do elemento da origem sempre fica posicionada na direita desse elemento. A multiplicidade do elemento alvo da relação sempre fica posicionada na esquerda desse elemento. Qualquer mudança de posição desses elementos não afetará a posição das multiplicidades, nem do do sentido da seta da relação. A Figura 3.9 demonstra como fica o resultado da inversão da posição das entidades de origem e alvo da Figura 3.8.

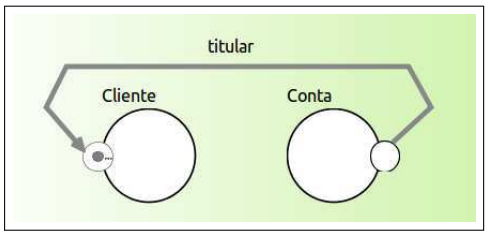

Figura 3.9: Relationship com Origem e Alvo Invertidos

Multiplicity

Uma relação contém duas multiplicidades, uma para o elemento da origem e outra para o elemento alvo (ponta da seta). A multiplicidade estabelece a quantidade de elementos da relação tanto na origem como no destino. São permitidas cinco multiplicidades visuais: SET (zero ou mais), ONE (apenas um), LONE (zero ou um), SOME (um ou mais) e NO (zero); além da multiplicidade textual EXPR representada numa expressão relacional com um inteiro. Na criação da relação o padrão de multiplicidade é SET, ou seja, zero ou mais elementos, na origem e no alvo.

Supondo a relação 'titular' de Conta para Cliente. Dada a invariante: *Toda Conta deve ter um ou mais Clientes como titular*, utiliza-se a multiplicidade SOME (um ou mais) no alvo da relação (direita, ou ponta da seta). A Figura 3.10

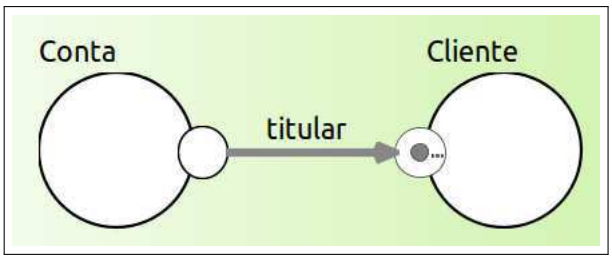

Figura 3.10: Multiplicity SOME (Direita)

Numa relação, a multiplicidade do elemento da direita (alvo) determina a quantidade de elementos da direita que podem estar associados, pela relação, a um elemento da esquerda. O inverso acontece na multiplicidade do elemento da esquerda (origem).

A multiplicidade do elemento da relação é representada por um círculo ou na ponta da seta (direita) ou na origem da seta (esquerda) da relação. A diferença do preenchimento ou cores do círculo definem o tipo de multiplicidade envolvida. As Figuras 3.11 e 3.12

Figura 3.11: Multiplicity na Direita

Figura 3.12: Multiplicity na Esquerda

exemplificam, respectivamente, as possibilidades de multiplicidades na direita e esquerda da relação Conta 'titular' Cliente da Figura 3.8.

É possível representar uma multiplicidade por meio de uma operação relacional e um inteiro, nesses casos a multiplicidade é composta por esses dois elementos (*RelationalOperation* e *Integer*). Além disso, a multiplicidade NO (nenhum) só pode ser aplicada na direita da relação, e se o elemento da esquerda da relação for uma quantificação.

3.2.4 Containment

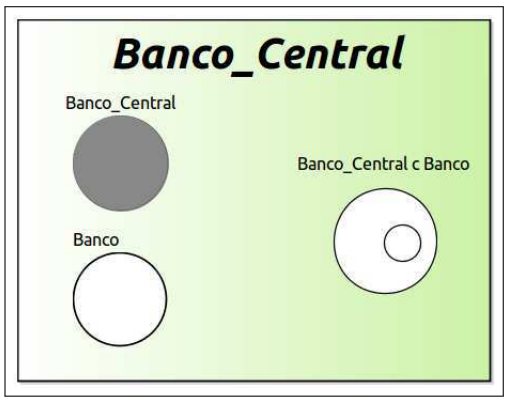

Figura 3.13: Containment

Com o *Containment* podemos expressar que os elementos de uma entidade (seja atômica, ou fruto de uma operação de conjuntos) estão contidos em outra entidade. É representado por um círculo menor dentro de um círculo maior ambos com borda preta e cor de preenchimento branca dando a ideia de uma entidade dentro de outra. Um containment é composto por duas Entidades (*Entity*) ou operações de conjunto (*SetOperation*). Um exemplo está na Figura 3.13 onde se expressa que o Banco_Central está contido em Banco.

3.2.5 Set Operation

Utilizamos uma *SetOperation* para representar uma operação de conjunto sobre duas entidades, ou operações de conjunto. A operação pode ser *Union*, *Intersection* ou *Complement* (união, interseção ou complemento). Uma operação de conjuntos não tem significado isoladamente, ela só tem sentido se for parte de outra expressão.

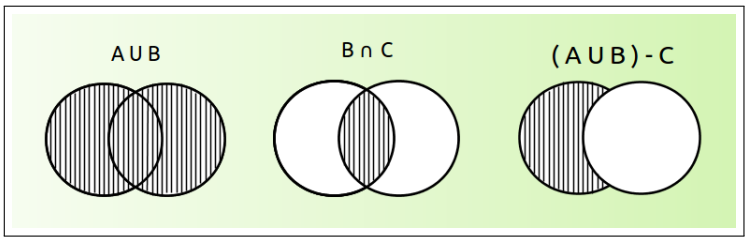

Figura 3.14: Set Operation

A representação de SetOperations é a mesma para operações de conjuntos nos diagramas de Venn [20]. A Figura 3.14 apresenta as operações de conjuntos entre as entidades A, B e C: união: $A \cup B$, interseção: $A \cap C$ e complemento: $(AUB) - C$.

A Figura 3.15 expressa as invariantes: (Tipos_Bancos) *Um banco pode ser público, ou privado, ou o Banco Central. O Banco Central é um banco Público que é único,* e (Conta_Banco) *Conta tem apenas um banco de origem que pode ser todos os bancos menos o Banco Central*). A relação *banco_origem* é composta pela Conta e a operação de conjunto $Banco - Banco\_Central$. Ou seja, o banco de origem de uma Conta não pode ser o Banco Central.

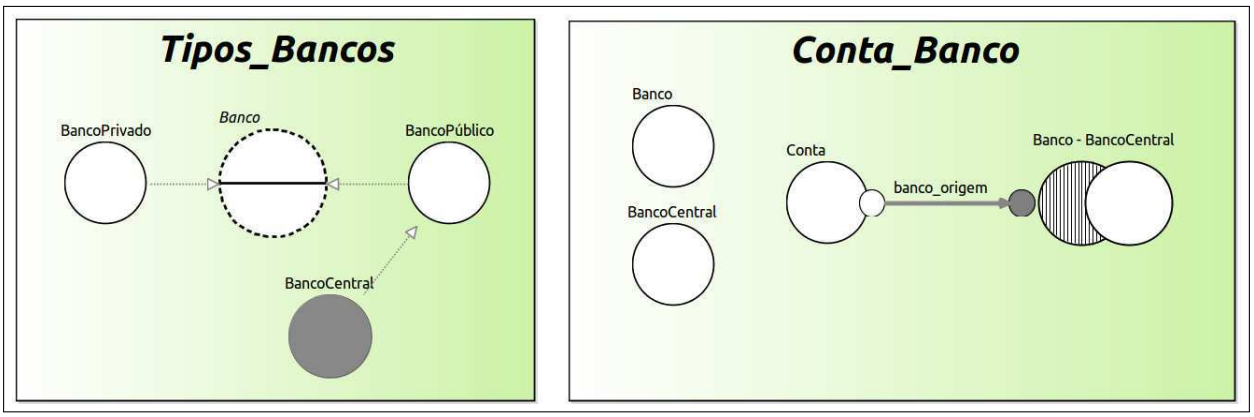

Figura 3.15: Exemplo da Set Operation Complemento

3.2.6 Cardinality

A quantidade de elementos de uma entidade ou operação de conjuntos é representada pela cardinalidade (*Cardinality*), que resulta num número inteiro positivo. Uma cardinalidade não tem significado se estiver numa invariante isoladamente, tendo sentido apenas se estiver

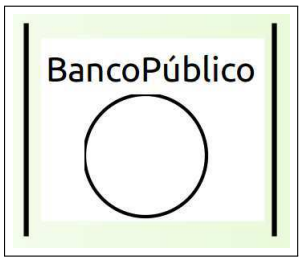

Figura 3.16: Cardinality

compondo uma operação relacional (*Relational Operation*). Assim, quando expressamos a invariante: a entidade Conta deve ter uma ou mais contas, na Figura 3.17, então sempre deve existir pelo menos uma conta nesse modelo.

A representação de *Cardinality* é feita por meio de duas barras verticais na cor preta, com o espaço entre as barras na cor branca. O elemento da cardinalidade será exibido dentro dessas barras. Essa é uma notação utilizada na Teoria dos Conjuntos [20]. A Figura 3.16 exibe a cardinalidade da entidade $BancoPublico$.

3.2.7 Relational Operation

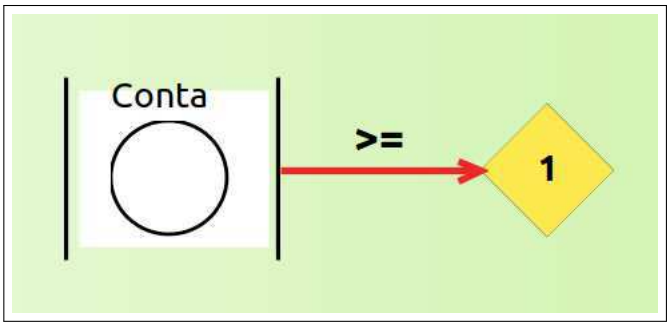

Figura 3.17: Relational Operation

Uma operação relacional, *RelationalOperation* compara duas estruturas do tipo *EntityType*. As operações relacionais podem ser: EQUALS (=), GREATER (>), LESS (<), GREATER_EQUALS (>=) ou LESS_EQUALS (<=). A *SetOperation* é representada por uma seta na cor vermelha, com o identificador textual do tipo da relação. Não há restrições para o posicionamento da seta. A Figura 3.17 exibe operação relacional GREATER_EQUALS (>=) entre a cardinalidade de $Conta$ e o inteiro 1.

3.2.8 Integer

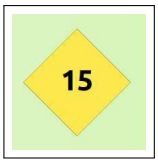

Figura 3.18: Integer

A estrutura *Integer* representa um número inteiro positivo. Sua sintaxe é um losângulo na cor amarela com bordas na cor preta. Assim como na *Cardinality*, um *Integer* só tem significado se estiver compondo uma *Relational Operation*.

3.2.9 Logical Operation

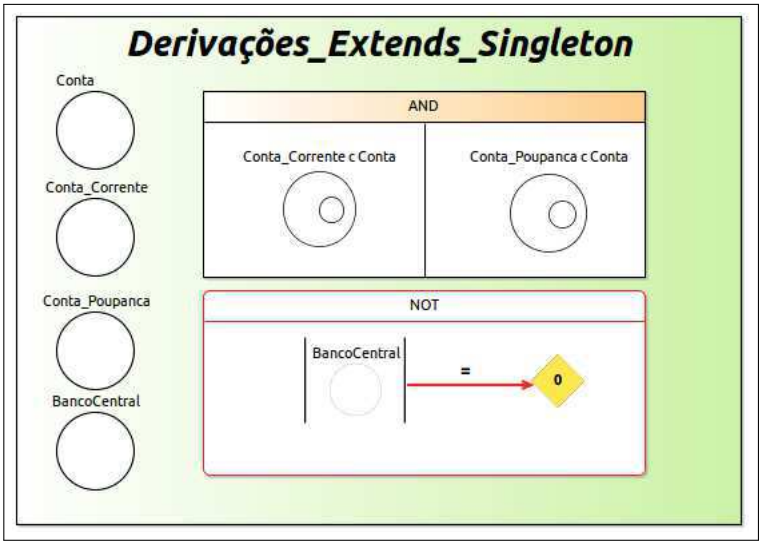

Figura 3.19: Logical Operation

Uma operação logica (*LogicalOperation*) expressa uma conjunção (AND), disjunção (OR) ou negação (NOT) de outras expressões em GIRL do tipo *BooleanType*. A Figura 3.19 exibe duas operações lógicas que representam as restrições de que as entidades $Conta\_Corrente$ e $Conta\_Poupanca$ estão contias em $Conta$, e a restrição da entidade singleton $BancoCentral$. Operações lógicas são representadas por Retângulos que se distinguem da seguinte forma:

- AND: Retângulo com área na cor branca, com bordas na cor preta e identificado com AND no cabeçalho na cor laranja. Pode conter dois ou mais elementos que ficam no interior um em cada divisão separada por uma linha sólida na cor preta.

- OR: Retângulo com área na cor branca com bordas na cor preta e identificado com AND no cabeçalho na cor azul. Pode conter dois ou mais elementos que ficam no interior em divisões separadas por uma linha tracejada na cor preta.

- NOT: Retângulo com arestas arredondadas, com área na cor branca, com bordas na cor vermelha e identificado com NOT no cabeçalho. Comporta apenas um elemento.

3.2.10 Quantification

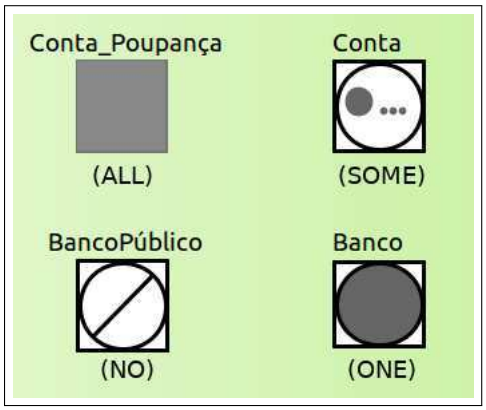

Figura 3.20: Quantification

As quantificações são utilizadas no cálculo dos predicados em lógica. GIRL oferece os quantificadores comuns da lógica, *para todo* e *existe* (ALL e $EXISTS$). Além desses, outros dois quantificadores derivados: *nenhum* (NO) e *um único* (ONE). As expressões com quantificação expressam as restrições que não foram possíveis de expressar com as entidades, relações e suas multiplicidades, containment e relações operacionais.

Quantificadores só tem algum significado se estiverem compondo alguma relação, como demonstra o exemplo da Figura 3.21, em que os quantificadores são usados para expressar a invariante: *Toda Conta Poupança deve ter apenas um Cliente como titular.* A representação desse exemplo não choca com a multiplicidade da SOME (um ou mais) da mesma relação

na Figura 3.10, ela apenas restringe a representação mais geral da relação $titular$, definindo que deve ser apenas um Cliente quando a Conta for $Conta\_Poupana$.

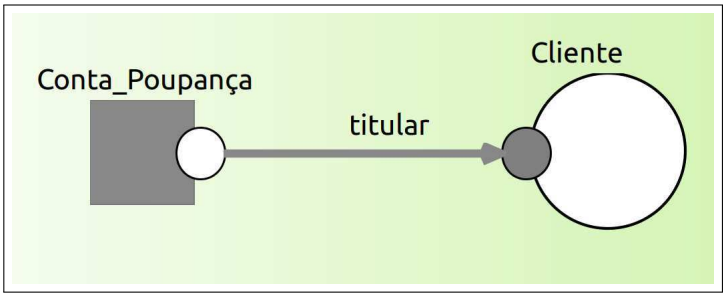

Figura 3.21: Exemplo de Uso da Quantification

Os quantificadores disponíveis em GIRL são: ALL (para todo), EXISTS (um ou mais), NO (nenhum) e ONE (um único). A definição da notação de quantificadores, busca uma proximidade com a notação de multiplicidades para melhor retenção da semântica dos conceitos por associação.

Quantifications têm apenas um elemento, que pode ser uma entidade ou operação de conjuntos. A Figura 3.20 exibe a representação dos quantificadores permitidos em GIRL, que são:

- ALL: Quadrado na cor cinza.
- EXISTS: Quadrado contendo o mesmo símbolo da multiplicidade SOME.
- NO: Quadrado contendo o mesmo símbolo da multiplicidade NO.
- ONE: Quadrado contendo o mesmo símbolo da multiplicidade ONE.

3.2.11 Implication

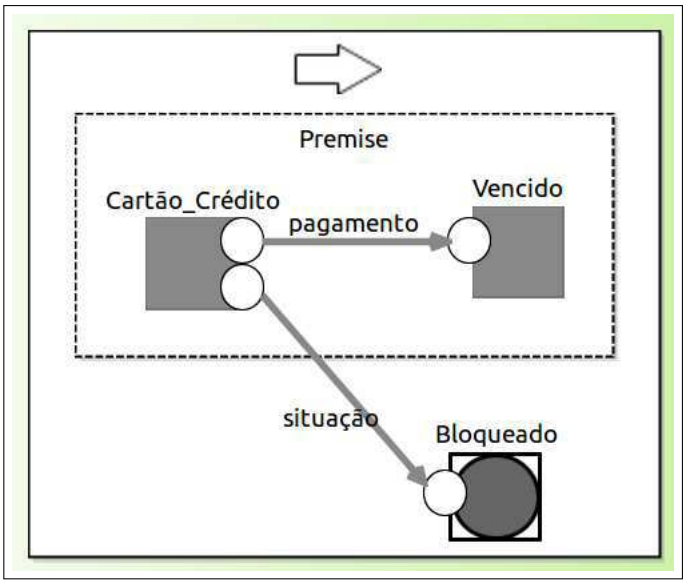

Figura 3.22: Implication

Na linguagem GIRL, uma *Implication* (implicação) descreve restrições do tipo: Se algo é verdadeiro (premissa), então conclui-se que outra coisa é verdadeira (conclusão). São semelhantes à estruturas condicionais em lógica e programação *if/then*.

As implicações são compostas por estruturas do tipo *BooleanType* ou *RelationshipType* e são formadas por um elemento na premissa e um, ou mais elementos, na conclusão. As relações que compõem uma implicação devem conter pelo menos um quantificador na premissa. Caso haja uma relação na premissa, então a conclusão só pode ter relações e os quantificadores da relação da premissa devem ser apenas do tipo ALL. Caso a premissa seja um *BooleanType*, então só pode haver *BooleanTypes* na conclusão.

A Figura 3.23 exibe um modelo que contém uma implicação para restringir a relação $situacao$ (conclusão), dependendo da relação $pagamento$ (premissa), ambas representadas genericamente em outras invariantes. O modelo representa os seguintes requisitos:

- *Um Cartão de Crédito deve ter apenas uma Situação que pode ser ou Ativo ou Bloqueado* - $Situacao\_Cartao\_Credito$ e $Cartao\_Credito$
- *Um Cartão de Crédito deve ter apenas um indicador de Pagamento indicando se está Vencido, ou Pago o Mínimo.* - $Pagamento\_Cartao$ e $Cartao\_Credito$

- *Se um Cartão de Crédito está com o pagamento Vencido, então a Situação dele deve ser Bloqueado. -* $Cartao\_Bloqueado$

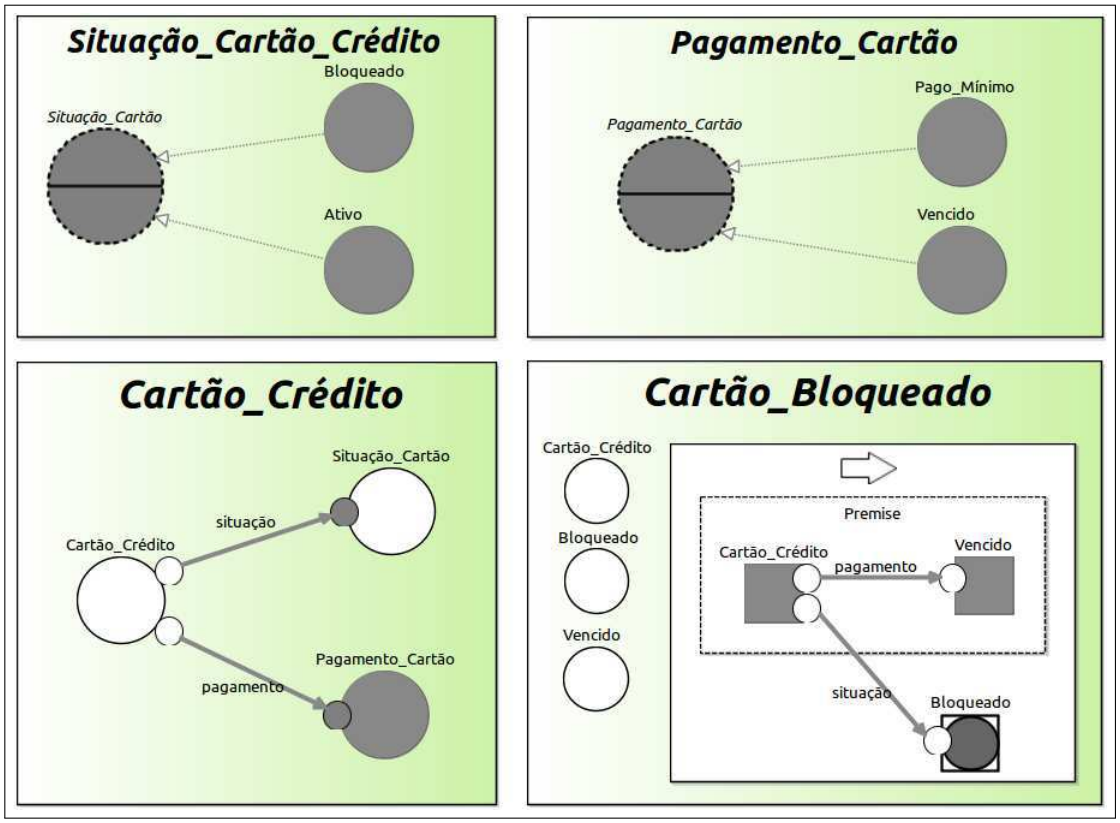

Figura 3.23: Exemplo de Uso da Implication

Uma implicação em GIRL (*Implication*) é uma estrutura condicional contendo uma premissa e uma conclusão. Ela é composta por duas ou mais estruturas do tipo *BooleanType* ou *RelationshipType*, sendo a primeira a premissa e as demais a conclusão. Caso haja mais de um elemento na conclusão, eles serão considerados uma conjunção implícita. Na Figura 3.22 (extraída do modelo completo na Figura 3.23), apresenta os seguintes elementos da implicação:

- **Implicação**: Retângulo com área na cor branca e no cabeçalho uma seta, todos com contorno na cor preta.

- **Premissa**: Retângulo com área na cor branca com bordas tracejadas na cor preta. Os elementos da premissa ficam nesse retângulo tracejado.

- **Conclusão**: Toda a área na implicação que está fora do retângulo da premissa. Relationship podem ter origem e alvo na premissa e conclusão, ou vice-versa. Uma *relationship*, cuja seta cruza a premissa e a conclusão, como a relação $situacao$ na Figura 3.22, indica que a relação pertence à conclusão. Isso acontece porque um dos elementos da relação também faz parte de elementos de uma relação da premissa, no exemplo, o quantificador *ALL* sobre a entidade *Cartão_de_Crédito*. Relações cujos elementos são exclusivamente da premissa, sua seta não ultrapassa os limites da premissa, como é o caso da relação *pagamento*, Figura 3.22. Decidimos manter apenas um elemento repetido na origem ou alvo de relações diferentes numa implicação. Assim, diminuímos a quantidade de elementos visuais numa implicação com relações.

3.2.12 Relationship Operation

Dada uma relação de paternidade de pessoas, onde uma Pessoa tem apenas um pai que é um Homem (vide Figura 3.24). Uma operação sobre essa relação é o fechamento transitivo (*Transitive Closure*). O fechamento transitivo sobre uma relação resulta em outra relação entre os elementos da entidade de origem da relação (esquerda) e todos os elementos que a relação original alcança direta e indiretamente (transitividade). Na relação 'pai', o fechamento transitivo resulta na relação entre cada pessoa e todos os seus ancestrais homens (pai, avô(pai do pai), bisavô (pai do pai do pai), etc.). Dada a seguinte instanciação da relação 'pai' na Figura 3.25, onde:

- José e Filipe têm João como pai;
- Marcos, Pedro e Isabel têm José como pai;
- Tiago, Mateus, Lucas e Maria têm Filipe como pai;
- Paulo tem Tiago como pai;

Então, a operação fecho transitivo sobre a relação 'pai' da Figura 3.25, resultaria na relação 'ancestrais' de cada pessoa, como mostra a Figura 3.26, onde:

- Marcos tem como ancestrais: José (pai) e João (avô);
- Pedro tem como ancestrais: José (pai) e João (avô);

- Isabel tem como ancestrais: José (pai) e João (avô);
- Tiago tem como ancestrais: Filipe (pai) e João (avô);
- Mateus tem como ancestrais: Filipe (pai) e João (avô);
- Lucas tem como ancestrais: Filipe (pai) e João (avô);
- Maria tem como ancestrais: Filipe (pai) e João (avô);
- José tem como ancestrais: João (pai);
- Filipe tem como ancestrais: João (pai);
- João **NÃO** tem ancestrais: { };

Dada a invariante: *Uma pessoa não pode ter ela mesma entre os seus ancestrais homens*, podemos representá-la em GIRL utilizando a operação Transitive Closure sobre a relação 'pai', como mostra Figura 3.27.Em outras palavras, o que a invariante quer dizer é: nenhuma pessoa está no conjunto resultante dessa operação na relação de pai (seus ancestrais homens - ou seja, o conjunto de todos os ancestrais homens da pessoa).

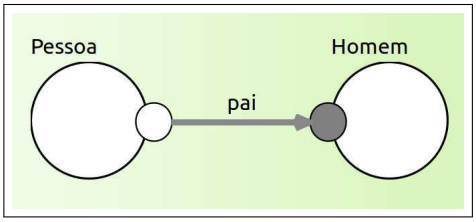

Figura 3.24: Exemplo Relationship 'Pai'

Uma *RelationshipOperation* do tipo *Trasitive Closure* não contém elementos, e só tem significado se for o alvo de uma relação cuja origem é uma quantificação.

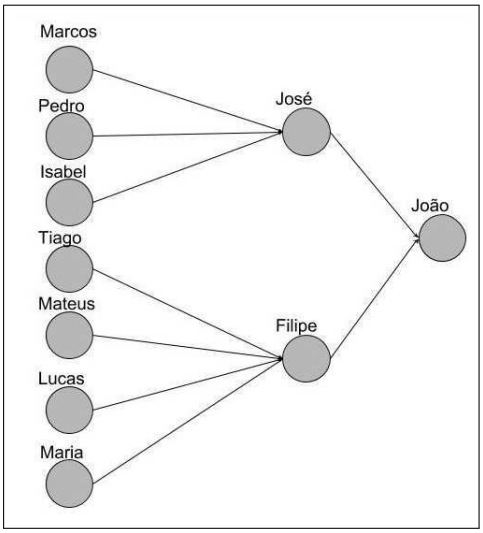

Figura 3.25: Instância da Relação 'Pai'

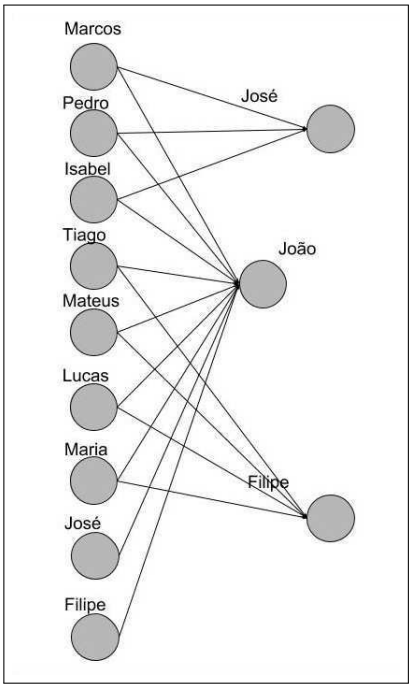

Figura 3.26: Instância da Relação 'Ancestrais' Resultante da Operação *Transitive Closure* na Relação 'Pai'

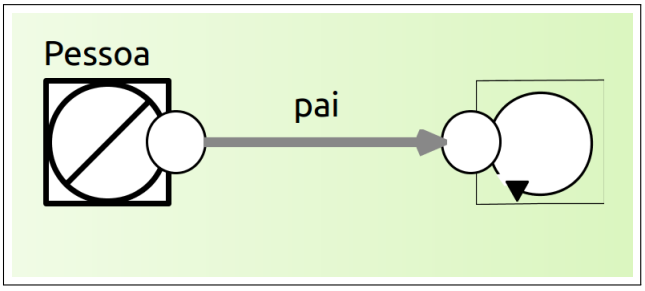

Figura 3.27: Exemplo Relationship Operation Sobre a Relationship 'Pai'

Capítulo 4

GIRL - Implementação e Execução

Este capítulo apresenta detalhes da implementação da linguagem GIRL. Além do editor gráfico, foi necessário implementar o mecanismo de execução da linguagem, que foi construído com base numa interpretação da linguagem GIRL para a linguagem principal, Alloy. Esse mecanismo foi implementado com transformações para gerar código de GIRL para Alloy.

4.1 Implementação

O ambiente para a criação de modelos em GIRL é um editor gráfico, que foi implementado com o *framework* de criação de editores gráficos *Sirius* - *Eclipse*[1]. No *Sirius*, a partir de um modelo EMF (*Eclipse Modeling Framework*), é possível especificar um editor de forma declarativa e observar o resultado em tempo real. Essas características facilitaram a criação do editor de GIRL.

Na implementação do mecanismo de execução da linguagem GIRL, foram utilizadas as ferramentas de transformação QVTOperational[2] para transformar o modelo GIRL no formato EMF, para o modelo EMF Alloy. Com esse modelo Alloy, geramos o código de Alloy com a ferramenta ACCELEO[3].

A verificação das invariantes na linguagem GIRL é realizada por meio do mecanismo de análise do Alloy [13], que traduz um código Alloy para alimentar um dos *SAT-Solvers* disponíveis na linguagem. Optamos por executar uma tradução de um modelo GIRL para

[1] Sirius - informações disponíveis em: https://www.eclipse.org/sirius/
[2] https://projects.eclipse.org/projects/modeling.mmt.qvt-oml
[3] ACCELEO: disponível em: http://www.eclipse.org/acceleo/

código Alloy através de duas transformações. A primeira transforma do modelo GIRL para o modelo Alloy (Girl2Alloy) e a segunda, do modelo Alloy para o código Alloy (Alloy2Text).

A transformação *Girl2Alloy* requer, além do metamodelo GIRL (Figura 3.1), o metamodelo de Alloy[4] , que está descrito na Figura 4.1. Essa transformação foi implementada na ferramenta *QVTOperacional*[5] que implementa a linguagem de mapeamentos operacionais definido pelo *Meta Object Facility* ™ (MOF ™) 2.0 *Query/View/Transformation* ™ (QVT ™)[6].

A transformação de GIRL para Alloy, não é um mapeamento direto de uma estrutura GIRL para uma estrutura Alloy. É necessário um processamento para: evitar duplicidade de informações de entidades e relacionamentos do modelo GIRL, transformar estruturas simples de GIRL para expressões mais complexas em Alloy, como a quantificação e implicações com relações.

Para a segunda transformação, Alloy2Text, que implementa a transformação *Model to Text*, foi utilizada a ferramenta ACCELEO, projeto Eclipse que implementa o padrão definido pelo *Object Management Group* (OMG) para a linguagem *Meta Object Facility* ™ (MOF ™) *Model to Text Language* (MTL). Essa transformação, sendo mais simples, requer apenas o metamodelo Alloy. Ela implementa uma tradução direta entre os elementos do modelo Alloy para o texto Alloy, gerando um arquivo do tipo .als, que alimenta o analisador Alloy.

[4]O metamodelo de Alloy é uma adaptação do metamodelo criado automaticamente com a ferramenta Kermeta, para software Pramana, Projeto Triskell (http://triskell.irisa.fr/). https://www.irisa.fr/triskell/Software/pramana/AlloyMetamodel/Alloy.ecore.html. A tese de doutorado de Sagar Sen referencia o seguinte endereço desse metamodelo de Alloy para download, embora com acesso restrito: https://www.irisa.fr/triskell/members/sagarsen/papers/thesis/ThesisSources/alloyecore

[5]Projeto Eclipse disponível em: https://projects.eclipse.org/projects/modeling.mmt.qvt-oml

[6]Especificação disponível em: https://www.omg.org/spec/QVT

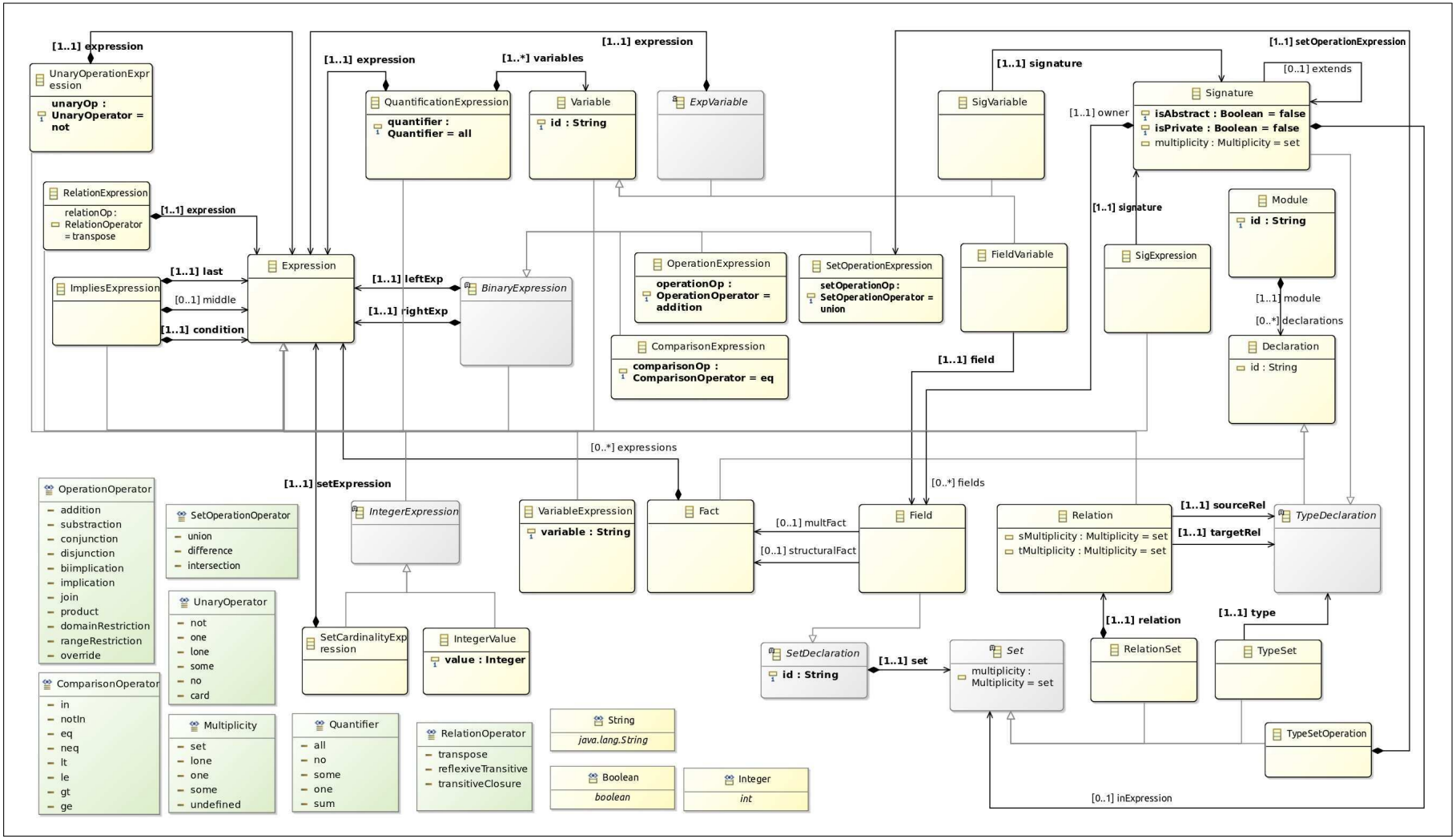

Figura 4.1: Metamodelo Alloy

4.2 Editor Gráfico

O desenho das estruturas da linguagem GIRL é realizado pelo seu editor gráfico implementado no ambiente *Sirius*. Esse editor provê os recursos para a criação de cada estrutura e, quando necessário, os recursos de copia/cola para agregar os elementos a uma estrutura.

O editor apresenta uma área de edição e um menu com as opções de criação de elementos. São as palhetas de criação de elementos da linguagem. A palheta de elementos simples ou únicos, contém as opções de criação de invariante, entidade, relacionamento, cardinalidade, inteiro, *containment* e implicação. As outras palhetas apresentam os elementos com mais de uma opção, como a operação de conjunto que pode ser união, interseção e complemento. Essas palhetas permitem a criação de: operações relacionais, operações de conjunto, operações lógicas, quantificações e operações de relacionamento.

A forma como as estruturas de GIRL são criadas depende do seu tipo e conteúdo, conforme listado a seguir:

- *Invariant*: clicar no elemento *Invariant* da palheta *Simple Elements*, clicar na área de edição e atribuir o identificador da invariante.

- *Entity*: clicar no elemento *Entity* da palheta *Simple Elements*, clicar na invariante desejada e atribuir seu identificador. Caso seja necessário estendê-la de outra entidade, realizar o que está descrito no item de extensão de entidades. As formas de criar ou diferencia entidades são:

  - *Entity - Abstract*: selecionar o elemento *Abstract Entity* na palheta *Simple Elements* e trazer (clicar) para a invariante desejada, em seguida atribuir o identificador.

  - *Entity - Singleton*: uma entidade é marcada (ou desmarcada) como sendo *singleton*, clicando-se duas vezes sobre a entidade desejada. Opcionalmente é possível alterar diretamente o atributo *isSingleton* nas propriedades da entidade.

  - *Extensão de entidades*: Para estender duas entidades, a entidade de origem (entidade filha) e de destino (entidade mãe) devem ter sido criados anteriormente numa mesma invariante. Só então, traça-se a seta de extensão, acionando o elemento *Extends* da palheta *Simple Elements*, em seguida, clicando primeiro na

entidade filha e depois na entidade mãe. Caso a extensão seja válida a seta é exibida entre as duas entidades.

- *Relationship*: só é possível estabelecer uma relação entre elementos de origem e alvo que tenham sido criados anteriormente. Com isso, cria-se uma relação acionando o elemento *Relationship* da palheta de *Siple Elements*, em seguida, clicando primeiro no elemento de origem e depois clicando no elemento alvo. Caso a relação entre os elementos selecionados seja possível, então são criadas as multiplicidades padrão para os elementos e a seta entre elas. Em seguida, deve-se atribuir o identificador textual da relação e ajustar as multiplicidades, quando necessário.

  A forma de definir a multiplicidade é a mesma no alvo e no destino da relação. Define-se uma multiplicidade diferente do padrão (SET), dando duplo clique na multiplicidade até aparecer a figura da multiplicidade desejada, ou teclar <F2> e inserir um operador relacional seguido de um inteiro de um dígito.

- *Containment*: para representar um *Containment* é necessário que seus elementos sejam previamente criados. Só então se cria a estrutura acionando o elemento *Containment* da palheta de *Simple Elements*, em seguida, clicando na invariante desejada. Isso cria um *containment* sem elementos. Depois, copiam-se os dois elementos, com o recurso de selecionar, copiar, selecionar o *containment* recém criado e colar. O primeiro elemento copiado estará contido no segundo. O rótulo é relacionado ao identificador dos seus dois elementos.

- *Set Operation*: para criar um *SetOperation*, é necessário que os dois elementos que compõem a operação tenham sido criados anteriormente. Então cria-se a estrutura acionando a operação desejada da palheta de *Set Operation*, em seguida, clicando na invariante desejada para criar uma *SetOperation* sem elementos. Copia-se, então, seus dois elementos, onde o primeiro elemento copiado fica na esquerda da operação e o segundo na direita.

- *Cardinality*: para se representar uma cardinalidade, é preciso que seu conteúdo, uma entidade ou uma operação de conjunto, tenha sido criado antes. Só assim criamos uma cardinalidade, acionando o elemento *Cardinality* da palheta de *Simple Elements*, em

seguida, clicando na invariante desejada, criando então uma *Cardinality* sem elementos. Por último, copia-se o elemento desejado na cardinalidade recém criada.

- *Relational Operation*: para criar uma *RelationalOpearation*, cria-se primeiro as duas estruturas que servirão de elementos da operação relacional. Em seguida, cria-se a operação acionando a operação relacional desejada na palheta de *Relational Operations*, clicando primeiro no elemento da esquerda da operação e depois clicando no elemento da direita. Caso a relação entre os elementos clicados seja possível, é criada uma seta com o identificador da operação entre os dois elementos envolvidos.

- *Integer*: acionar o elemento *Integer* da palheta de *Siple Elements*, clicar na invariante desejada para o posicionamento da estrutura. É criado um *Integer* com o valor inteiro 0. Por último, ajusta-se para o valor inteiro desejado clicando na tecla <F2> ou com duplo clique no valor do inteiro.

- *Logical Operation*: para criar uma operação lógica, seus componentes devem ter sido criados anteriormente. Só assim criamos uma operação lógica, acionando a operação desejada da palheta de *Logical Operations*, em seguida, clicando na invariante desejada. Por último, copia-se os elementos desejados para essa operação lógica recém criada.

- *Quantification*: para se criar uma quantificação, o seu elemento já deve ter sido previamente criado. Ao acionar o elemento desejado na palheta *Quantifications*, em seguida, clicar na invariante desejada, criamos uma *Quantification* sem elementos. Só então copia-se o elemento desejado na *quantification* recém criada.

- *Implication*: para compor uma implicação, é necessário criar primeiro os seus elementos no corpo da invariante desejada, em seguida, acionar o item *Implication* na palheta de *Simple Elements* e trazer para a invariante. Só então copiar os elementos para a implicação vazia. É recomendável copiar os elementos separadamente. Primeiro copia-se o elemento da premissa, depois o(s) elemento(s) da conclusão.

- *Relationship Operation*: para criar uma operação de relação, basta clicar no item *Trasitive Closure* na palheta *Relationship Operation*, em seguida, clicar na invariante desejada. Assim, fica disponível um elemento *Trasitive Closure* para compor uma relação.

4.3 Transformações na Verificação

Ao executar um modelo em GIRL, procede-se a verificação do modelo, utilizando o mecanismo de análise da linguagem Alloy. Como resultado da execução, o modelo é dito satisfazível ou não. Em caso positivo, cenários de instâncias de entidades e relacionamentos são exibidos um por vez, permitindo uma análise adicional da completude e correção do modelo.

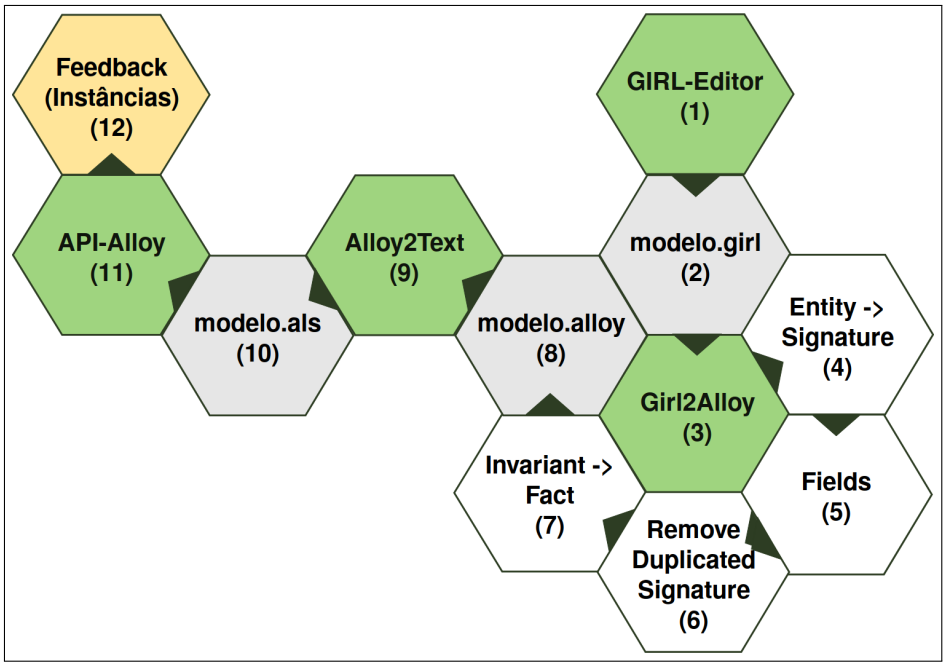

Figura 4.2: Processo de Execução da Linguagem GIRL

As atividades necessárias para transformar um modelo GIRL em código Alloy estão dispostas na Figura 4.2. O detalhamento dessas atividades, além de exemplos da transformação do modelo GIRL da Figura 3.23 para o código Alloy exibido no Código Fonte 4.1, estão descritos nos seguintes passos:

1. Atividade (1 a 2): criação do modelo GIRL no arquivo modelo.girl, que é submetido à transformação Girl2Alloy, na atividade (3);

Código Fonte 4.1: Exemplo de Implicação na Figura 3.23

```
1  module Modulo_6
2
3  ------------ Signatures ------------
4  one sig Ativo
5      extends Situacao_Cartao  {
6  }
7  one sig Bloqueado
8      extends Situacao_Cartao  {
9  }
10 sig Cartao_Credito  {
11     situacao: one Situacao_Cartao ,
12     pagamento: one Pagamento_Cartao
13 }
14 abstract sig Pagamento_Cartao {
15 }
16 one sig Pago_Minimo
17     extends Pagamento_Cartao  {
18 }
19 abstract sig Situacao_Cartao {
20 }
21 one sig Vencido
22     extends Pagamento_Cartao  {
23 }
24
25 -------------- Facts ---------------
26 fact Cartao_Bloqueado{
27         all var_3d57b0a6 : Cartao_Credito  |
28         all var_b0ccb64 : Vencido  |
29         one var_3b48930e : Bloqueado  |
30                 (   var_b0ccb64  in  ( var_3d57b0a6 . pagamento ) )
31                implies
32                (   var_3b48930e  in  ( var_3d57b0a6 . situacao ) )
33 }
34 fact Cartao_Credito{
35 }
36 fact Pagamento_Cartao{
37 }
38 fact Situacao_Cartao_Credito{
39 }
40
41 -------------- Commands ---------------
42 pred Modulo_6  [ ] {}
43 run Modulo_6
```

2. Atividade (4) — *Entity* -> *Signature*: a partir de todas as entidades GIRL (*Entity*), criar as assinaturas Alloy (*Signature*), definindo atributos de abstract ou singleton e assinatura que estende, caso se aplique. No Código Fonte 4.1, nas linhas 4 e 5, a assinatura $Ativo$ com multiplicidade one, que estende a assinatura $Situacao\_Cartao$, corresponde à entidade $Ativo$ que é *singleton* e estende a entidade $Situacao\_Cartao$ da Figura 3.23.

3. Atividade (5) — *Fields*: Criar todos os *fields* de assinaturas, que são as relações em Alloy, a partir das relações em GIRL (*Relationships*), onde:

   - A entidade envolvida no elemento *source* (entidade, quantificação ou operação de conjunto) da *relationship* será a assinatura correspondente em Alloy que é a detentora da relação, ou seja, o *field* correspondente à relação pertence à assinatura *source* correspondente em Alloy. Como exemplo, as *relationships* $situacao$ e $pagamento$ na invariante $Cartao\_Credito$ da Figura 3.23, pertencem à assinatura $Cartao\_Credito$, linhas 10, 11 e 12 no Código 4.1, porque a entidade $Cartao\_Credito$ em GIRL é o elemento *source* das relações;
   - O identificador da *relationship* será o identificador do *field* na assinatura Alloy que detém a relação. Como exemplo, as *relationships* $situacao$ e $pagamento$ na invariante $Cartao\_Credito$ da Figura 3.23 tem o mesmo identificador nos *field* correspondentes $situacao$ e $pagamento$ da assinatura $Cartao\_Credito$, linhas 10, 11 e 12 no Código 4.1;
   - O tipo do *field* (a expressão após o ':') na assinatura Alloy correspondente ao elemento *target* da *relationship* de GIRL será:
     - A assinatura Alloy correspondente à entidade do elemento *target* da *relationship*. Nas linhas 11 e 12 no Código 4.1, os *fields* $situacao$ e $pagamento$ possuem os tipos que são, respectivamente, as assinatura correspondentes às entidade $Situacao\_Cartao$ e $Pagamento\_Cartao$, ambos os elemento *target* da relação em GIRL na invariante $Cartao\_Credito$, da Figura 3.23.
     - A assinatura Alloy correspondente à entidade mãe, da entidade envolvida no elemento *target* da *relationship*. Isso acontece quando a relação tam-

bém for aplicada na entidade mãe em outra expressão. Na transformação do modelo da Figura 3.23, as entidades $Ativo$ e $Bloqueado$ associadas às quantificações nos elementos *target* das relações $situacao$ e $pagamento$, na invariante $Cartao\_Bloqueado$, não foram aplicadas como tipos dessas relações, mas sim as assinaturas associadas às entidades mãe $Situacao\_Cartao$ e $Pagamento\_Cartao$. Isso porque essas relações também foram aplicadas à essas entidades mãe na invariante $Cartao\_Credito$. As linhas 11 e 12 do Código 4.1 demostram o código Alloy nesses casos.

  - A operação de conjunto do elemento *target* da *relationship*, sendo composta pelas assinaturas Alloy que correspondem às entidades envolvidas na operação.

- A multiplicidade do *field* na assinatura Alloy, é a mesma para multiplicidade do elemento *target* da *relationship* de GIRL, quando a relação não tem elementos quantificados, e caso a multiplicidade seja: *one, lone, some* e *set*. Caso seja uma expressão relacional. Será criada uma expressão quantificada no fato associado à invariante, comparando a cardinalidade da relação em Alloy com o operador relacional e o inteiro envolvidos na expressão da multiplicidade.

  As multiplicidades ONE no elemento *target* das relações $situacao$ e $pagamento$ no modelo GIRL da Figura 3.23, são aplicadas no tipo das respectivas relações, como descrito nas linhas 11 e 12 do Código 4.1.

- A multiplicidade associada ao elemento *source* da *relationship* de GIRL, que não tem quantificação, produz a criação de uma expressão envolvendo apenas as multiplicidades *one, lone, some* sobre a inversa da relação. Caso a multiplicidade seja uma expressão relacional, produz uma expressão quantificada comparando a cardinalidade sobre a inversa da relação com o operador relacional e o inteiro envolvidos na expressão da multiplicidade. Ambas as expressões são criadas como um fato associado à invariante da relação.

4. Atividade (6) — *Remove Duplicated Signature*: retirar as assinaturas duplicadas, transferindo os atributos de *abstract* e *singleton* e *fields* para a única assinatura que permanecer. Ao transformar o modelo da Figura 3.23, as entidades $Situacao\_Cartao$,

$Pagamento\_Cartao$, $Ativo$ e $Bloqueado$ aparecem em duas invariantes e a entidade $Cartao\_Credito$ em três, mesmo assim, o Código Fonte 4.1 gerado, exibe apenas uma ocorrência das assinaturas correspondentes.

5. Atividade (7) — *Invariant -> Fact*: a partir de cada invariante GIRL (*Invariant*), criar um fato em Alloy (*fact*) e inserir as expressões da invariante no fato, conforme as seguintes definições:

   - Inserir apenas *invariantElements* que sejam diferentes de: *Entity, Interger, Quantification, Cardinality, RelationshipOperation* e *SetOperation*. Como essas informações ficam expressas nas assinaturas, é possível haver fatos no código Alloy sem expressões;
   - Para *Containments* de GIRL que não estejam em *LogicalOperation* ou *Implication*, criar uma *ComparisonExpression* de Alloy com operador *in*;
   - Para *RelationalOperations* de GIRL que não estejam em *LogicalOperation* ou *Implication*, criar uma *ComparisonExpression* de Alloy com um dos operadores: *lt, le, eq, gt* ou *ge*; correspondendo, respectivamente, aos operadores do *RelationalOperation*: LESS, LESS_EQUALS, EQUALS, GREATER ou GREATER_EQUALS;
   - Para *LogicalOperations* de GIRL que não estejam em *LogicalOperation* ou *Implication*, criar uma *OperationExpression* de Alloy com operador *conjunction* ou *disjunction*, para AND ou OR; ou criar uma *UnaryOperationExpression* de Alloy com operador *not*.
   - Para *Relationships* de GIRL com *Quantification* em seus elementos, e que não estejam em *Implication*, criar uma *QuantificationExpression* de Alloy com operador *all, one, some* ou *no*;
   - Para *Implications* de GIRL, criar *ImpliesExpression*, cujos argumentos são:
     - Existindo relationships na implicação, criar uma *QuantificationExpression* de Alloy com operador *all, one, some* ou *no*, envolvendo expressões *ComparisonExpression* de Alloy com operador *in* para as relações dos elementos quantificados na premissa e na conclusão. O fato em Alloy

$Cartao\_Bloqueado$ é o único que contém uma expressão no Código Fonte 4.1 nos fatos Alloy, sendo uma implicação, como exibido nas linhas 26 a 39;

- Implicações sem relationships, geram expressão de implicação em Alloy sendo a premissa e a conclusão uma expressão Alloy associada a estrutura *Containments*, *RelationalOperations* ou *LogicalOperations* de GIRL. A transformação da estrutura segue o descrito em item anterior relacionado a um desses tipos de expressão.

6. Atividade (8) — modelo.alloy: gera modelo alloy no formato EMF.

7. Atividade (9 e 10) — Alloy2Text: a partir do modelo.alloy gera código Alloy em modelo.als.

8. Atividade (11 e 12) — API-Alloy: o modelo.als é submetido ao analisador Alloy via API do Alloy[7], que, caso seja satisfazível, gera o conjunto de instâncias — Atividade (12) — que será detalhada na próxima seção deste capítulo.

A Figura 4.3 demonstra um modelo GIRL para os requisitos da estrutura *Entity* própria linguagem GIRL[8]. Também apresentamos a correspondência entre os elementos gráficos de GIRL para uma instância do modelo EMF de GIRL num arquivo do tipo XMI com extensão girl. Com base nesse modelo EMF executamos a transformação GIRL para Alloy, gerando uma instância do modelo EMF de Alloy (arquivo XMI resultante com extensão alloy). A figura 4.4 identifica a correspondência entre os elementos após a transformação Girl2Alloy. Por último, a transformação Alloy2Text gera o texto correspondente ao código Alloy da instância do modelo EMF de Alloy, como descrito no Código Fonte 4.2.

Para validar as transformações implementadas, adotamos a estratégia de aplicar alguns casos de teste de cada estrutura transformada, conferindo o código Alloy gerado e as instâncias geradas no analisador. Por limitações de recursos e tempo, não executamos testes mais completos das transformações, ficando como melhoria a ser executada nos trabalhos futuros.

[7] API Alloy4.2 - disponível em: http://alloy.lcs.mit.edu/alloy/documentation/alloy-api/index.html

[8] Os detalhes da construção do modelo da estrutura *Entity* estão na próxima seção, onde abordamos o processo de verificação dos modelos em GIRL.

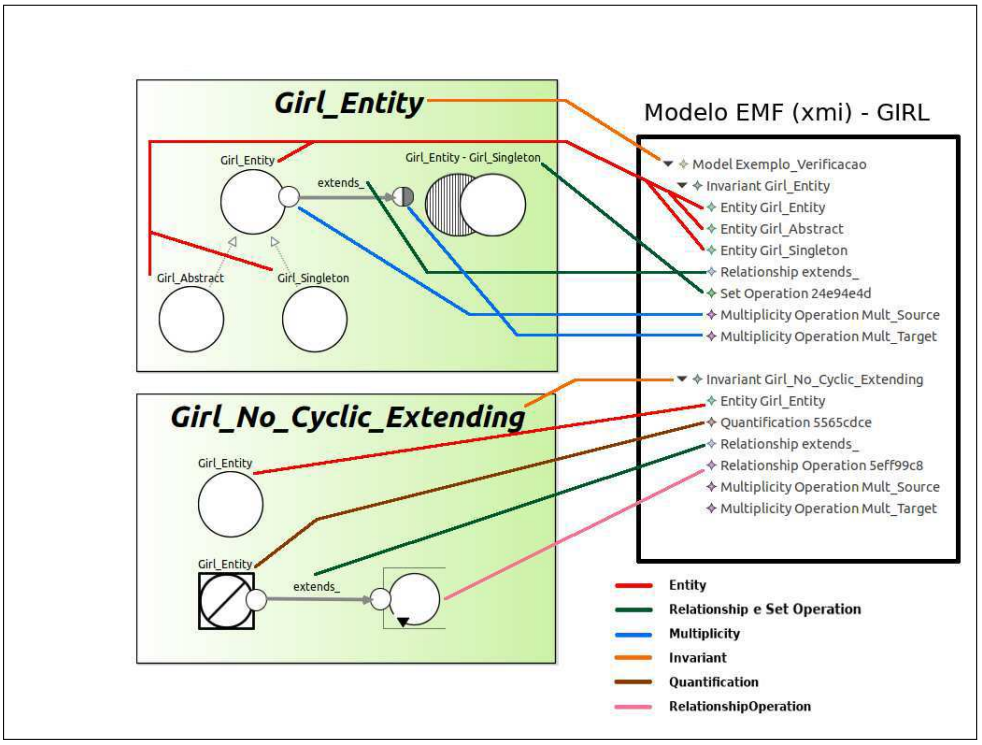

Figura 4.3: Mapeamento do Gráfico GIRL para o modelo EMF

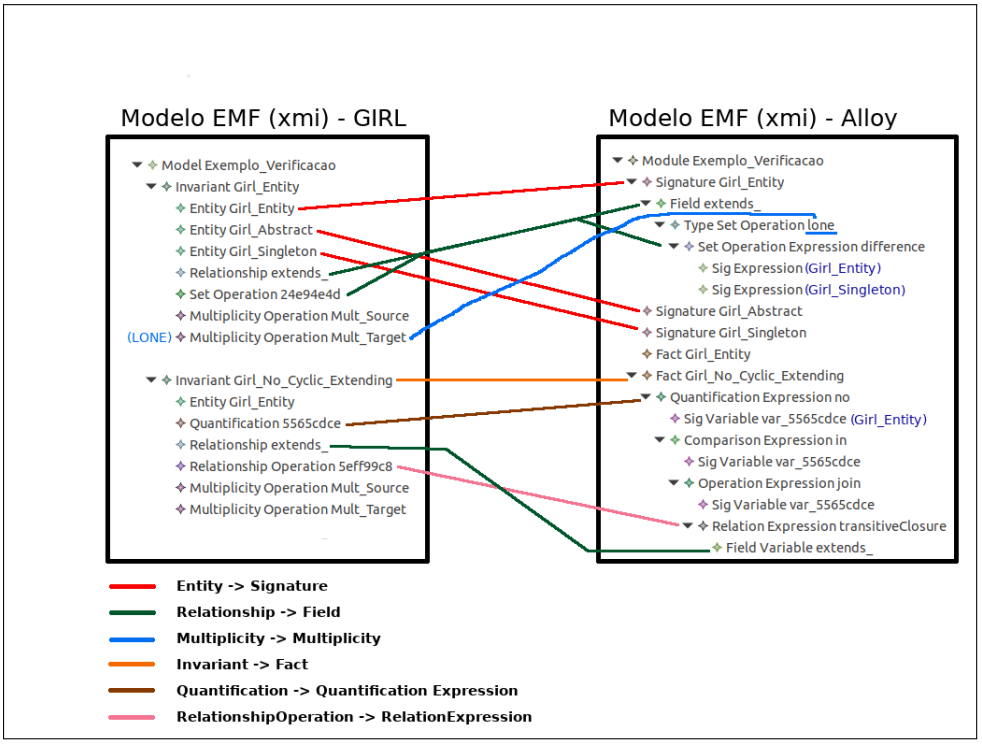

Figura 4.4: Mapeamento do modelo EMF GIRL para Alloy

Código Fonte 4.2: Invariantes de Entity (Girl_Entity.als)

```
module  Exemplo_Verificacao

------------ Signatures ------------
sig Girl_Abstract
    extends Girl_Entity  {
}
sig Girl_Entity  {
    extends_: lone   (Girl_Entity  - Girl_Singleton)
}
sig Girl_Singleton
    extends Girl_Entity  {
}

-------------- Facts ---------------
fact Girl_Entity{
}

fact Girl_No_Cyclic_Extending{
        no var_5565cdce : Girl_Entity  |
            (   var_5565cdce  in  ( var_5565cdce . ^(extends_) ) )
}

-------------- Commands ---------------
pred Exemplo_Verificacao  [ ] {}
run Exemplo_Verificacao
```

4.4 Processo de Verificação

Com o código Alloy gerado, a verificação das invariantes de GIRL pode ser submetida ao analisador do Alloy, que gera um conjunto de instâncias das entidades e dos relacionamentos com base nas restrições estabelecidas no modelo GIRL. Cada instância gerada é apresentada de forma gráfica, sendo permitido passar para a próxima instância até a última. A Figura 4.6 apresenta exemplos de instâncias geradas a partir do modelo expresso na Figura 4.5.

Para demonstrar uma verificação de invariantes, modelamos e verificamos os requisitos da própria linguagem GIRL. Assim, analisaremos as invariantes para a estrutura *Entity* apresentando modelos e instâncias. Para evitar confusão entre as nomenclaturas, denominaremos as entidades do modelo com o sufixo 'Girl_'.

Obtemos o modelo mais preciso e consistente na terceira rodada de verificação. Ini-

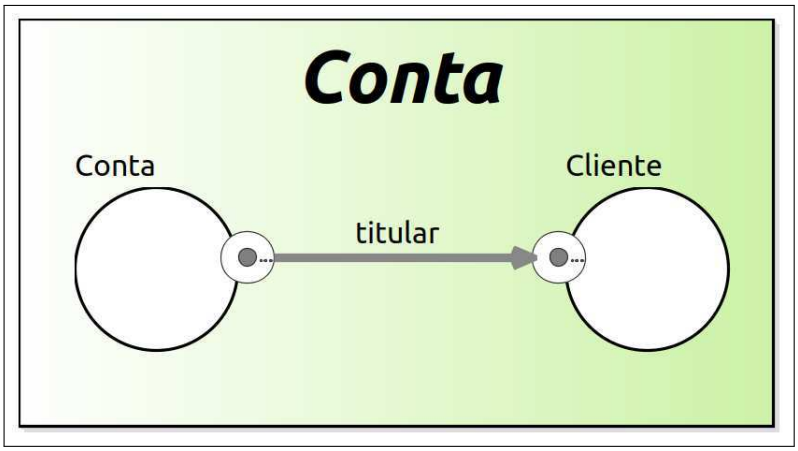

Figura 4.5: Exemplo Relação com duas Multiplicidade

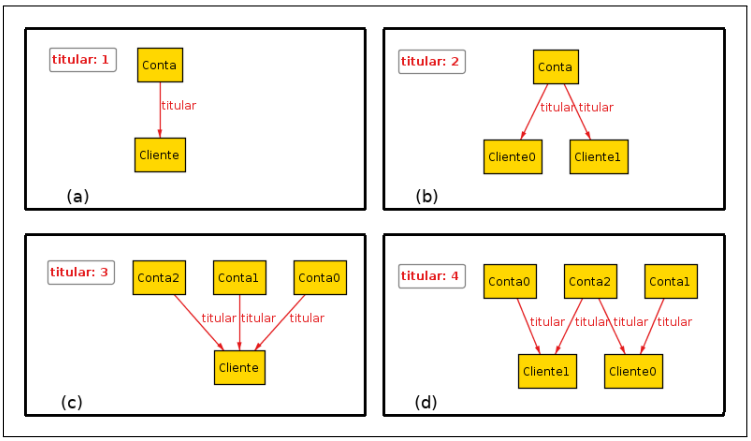

Figura 4.6: Instâncias da Verificação

ciamos a modelagem com as seguintes invariantes para a implementação da estrutura *Girl_Entity*: (1) Uma *Girl_Entity* pode ser *Girl_Abstract* ou *Girl_Singleton*; (2) Uma *Girl_Entity* estende uma ou nenhuma *Girl_Entity*.

Na primeira verificação usamos o modelo da Figura 4.7. Encontramos então uma instância inconsistente na Figura 4.8, onde uma entidade *Girl_Singleton* estendia ela mesma.

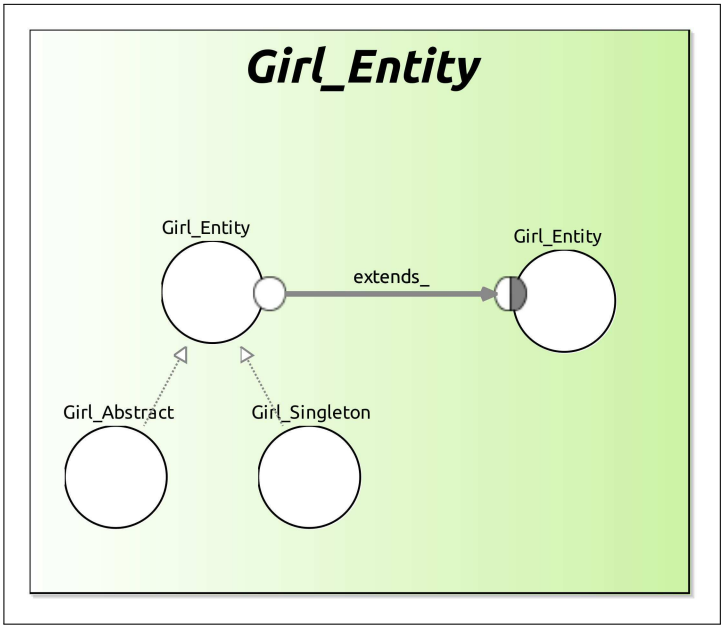

Figura 4.7: Requisitos Entity - Versão 1

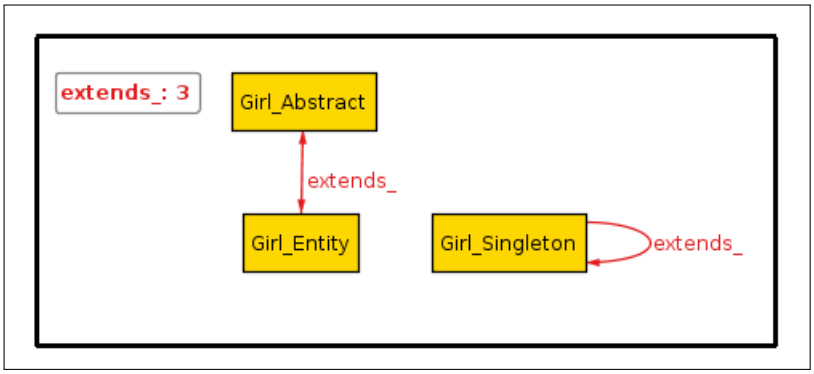

Figura 4.8: Instâncias da Verificação - Versão 1

Acrescentamos então a invariante (3) Uma *Girl_Entity* não pode estender a si mesma, direta ou indiretamente. Elaboramos a segunda versão do modelo, que está na Figura 4.9. A instância da Figura 4.10 apresentou uma situação inválida para o modelo, três *Girl_Entities* configuravam um ciclo na extensão.

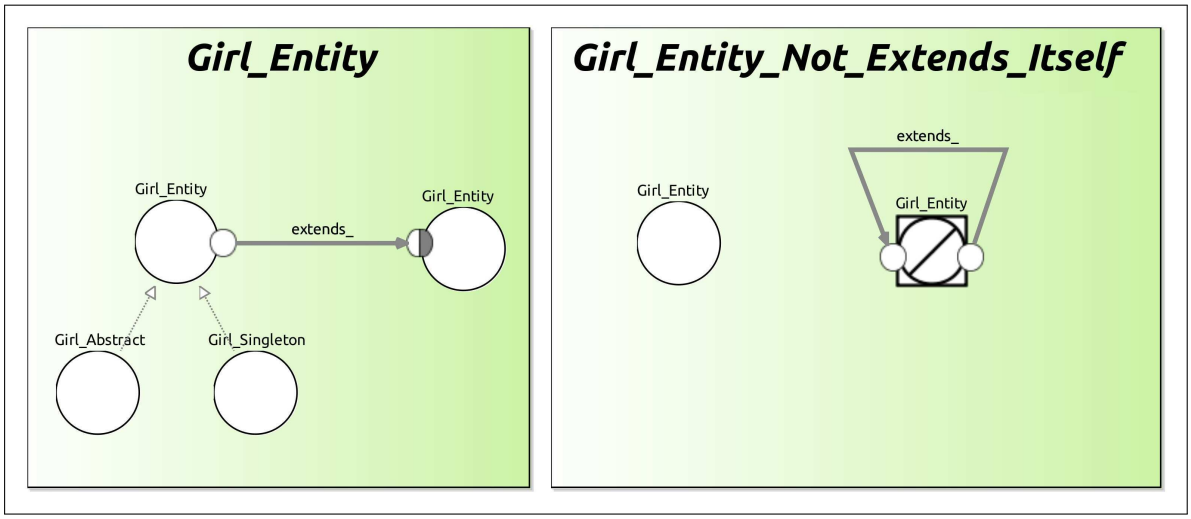

Figura 4.9: Requisitos Entity - Versão 2

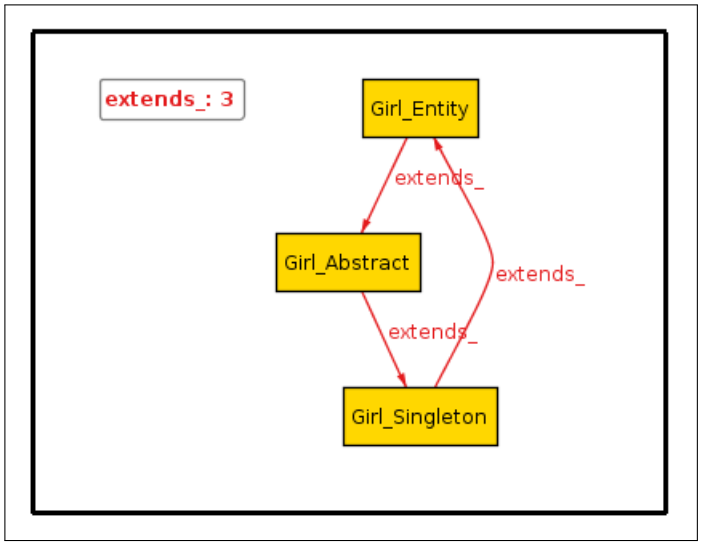

Figura 4.10: Instâncias da Verificação - Versão 2

Por fim, corrigimos a invariante (3) para: Uma *Girl_Entity* não pode estender a si mesma, direta ou indiretamente - não pode haver um ciclo nas extensões de *Girl_Entity*. Elaboramos a terceira versão do modelo, que está na Figura 4.11. Consideramos o modelo estável, a exemplo da instância apresentada na Figura 4.12.

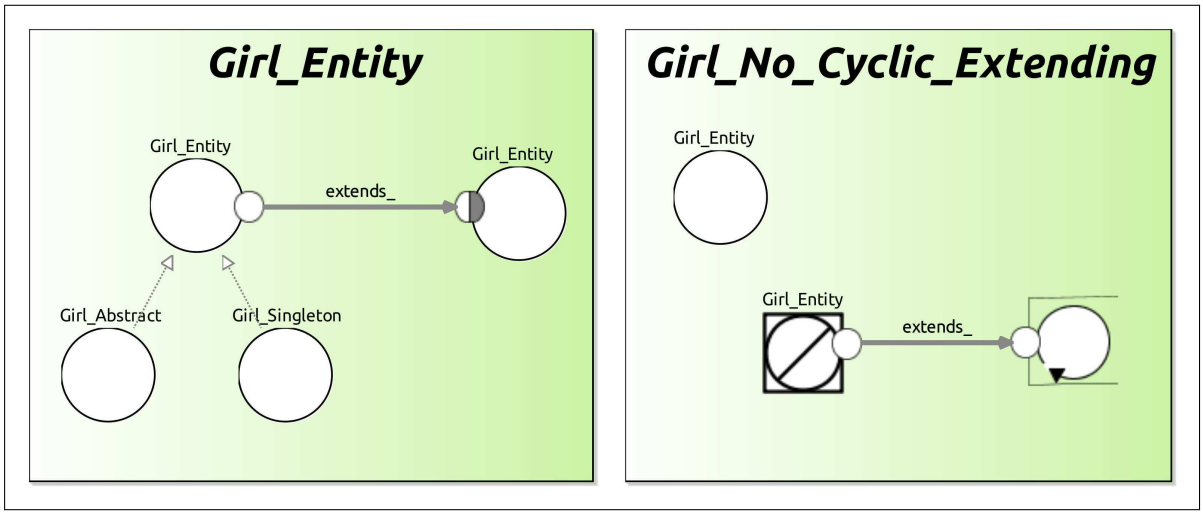

Figura 4.11: Requisitos Entity - Versão 3

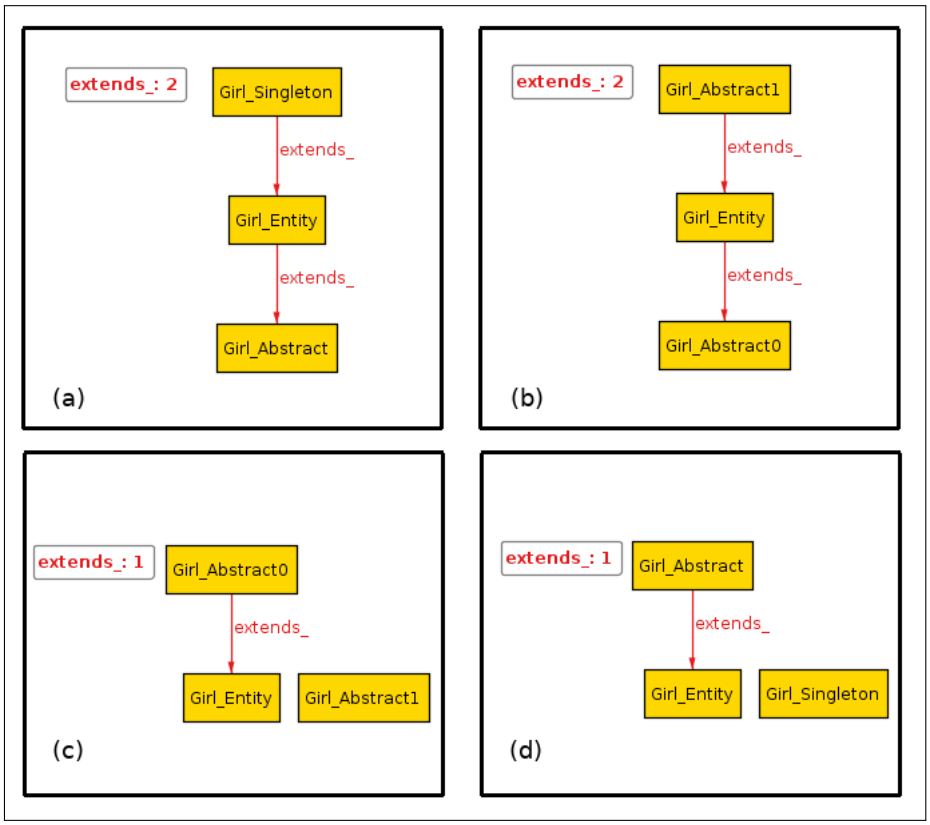

Figura 4.12: Instâncias da Verificação - Versão 3

Capítulo 5

Estudo Empírico

A linguagem GIRL se propõe a ser uma alternativa mais conciliadora de especificação formal de requisitos em relação a métodos formais tradicionais. Para isso, é necessário que seja: (1) utilizável por desenvolvedores de software menos afeitos a formalismos matemáticos, (2) passível de verificação automática de ambiguidades e inconsistências e (3) comunicável a todas as partes interessadas no processo de desenvolvimento de software. Resta então avaliar se essas metas da linguagem são alcançadas durante sua utilização. Temos então o norte do estudo empírico, ou pesquisa, que será detalhado nas próximas seções deste capítulo.

5.1 Metodologia da Pesquisa

O estudo empírico conduzido adotou a estratégia do tipo julgamento de tarefas (*judgment tasks*) [32]. Conforme classificação compilada por Stol e Fitzgerald, nesse tipo de estudo, os participantes são solicitados a julgar ou mensurar comportamentos, ou mesmo discutir um tópico de interesse. Também esclarecem que a forma de estabelecer a amostra é sistemática e não representativa.

Em linhas gerais, os sujeitos do estudo realizaram tarefas de julgamento, mensurando a utilidade da linguagem GIRL. Além disso, observações adicionais foram realizadas para avaliar a receptividade da linguagem por parte dos sujeitos.

5.1.1 Contextualização

É determinante para o sucesso no desenvolvimento de softwares que a descoberta de falhas, como imprecisões ou inconsistências, ocorram o mais cedo possível, isto é, bem antes da efetiva utilização do software, antes ainda da implementação do software, mais precisamente enquanto se define os requisitos do software. A solução mais recomendada para atender essa necessidade é utilizar especificações formais para detectar antecipadamente as possíveis falhas no software. Assim, o cenário mais promissor é antecipar estas descobertas utilizando especificações formais de requisitos que sejam adequadas às habilidades comuns de analistas de software.

Com base nesse cenário, pretendemos investigar qual a percepção de analistas de requisitos da utilidade de uma linguagem gráfica de modelagem de requisitos, baseada em lógica, como um instrumento para facilitar a detecção antecipada de inconsistências em requisitos de software.

Traduzindo essa meta para o contexto da nossa pesquisa: deseja-se analisar o comportamento de analistas de requisitos durante o aprendizado e utilização da linguagem GIRL em atividades de representação e verificação automática de requisitos, com o objetivo de averiguar se a linguagem é útil para detectar falhas em requisitos.

5.1.2 Objetivos da Pesquisa

Os objetivos do estudo foram expressos utilizando o método GQM (Goal, Question, Metric) proposto por Solingen e Berghout [36]. Os objetivos a serem atingidos durante a pesquisa devem ser expressos em termos do objeto de estudo, finalidade, foco de qualidade, perspectiva e contexto.

O objetivo principal da pesquisa pode ser resumido na seguinte sentença: **Avaliar** a linguagem GIRL na representação e verificação automática de requisitos (*objeto de estudo*), **com a intenção** de caracterizar a utilidade do uso de especificação formal com recursos gráficos (*finalidade*); **com respeito** a efetividade do uso de especificações formais em termos da qualidade da especificação e do grau de dificuldade no aprendizado da linguagem (*foco de qualidade*); **do ponto de vista** de analistas de requisitos (*perspectiva*), **no contexto** de elicitação, análise e especificação de requisitos invariantes, de baixa complexidade, em projetos

de desenvolvimento de software num órgão público, envolvendo analistas de requisitos.

Questões de Pesquisa

Para cumprir o objetivo principal do estudo, buscou-se responder às seguintes questões de pesquisa:

QP1 : *Qual o nível de dificuldade percebida no aprendizado e utilização da linguagem gráfica GIRL?* - Tencionamos identificar se a linguagem GIRL é de fácil aprendizado e se é considerada atrativa e útil para a modelagem de requisitos.

QP2 : *Qual a corretude dos requisitos representados durante a utilização da linguagem GIRL?* - Outra informação importante é saber se o conhecimento da linguagem foi bem aplicado gerando requisitos precisos e completos durante a modelagem em GIRL.

QP3 : *Qual a utilidade percebida na verificação automática de requisitos representados durante a utilização da linguagem GIRL?* - A verificação automática para detecção de falhas é um dos benefícios do uso de especificações formais. Pretendemos descobrir se esse benefício foi percebido pelos sujeitos da pesquisa.

QP4 : *Qual a efetividade da representação gráfica durante a utilização da linguagem GIRL?* - Investigamos se a notação gráfica da linguagem GIRL favoreceu seu aprendizado e utilização com base nos princípios que estabelecem a efetividade de notações visuais, denominados *Physics of Notation* (PoN) [23].

5.1.3 Descrição da Pesquisa

A pesquisa se configura como uma investigação na área de conhecimentos da Engenharia de Requisitos no processo de desenvolvimento de software. A estratégia de pesquisa adotada foi do tipo julgamento de tarefas (*judgment tasks*) [32], com algumas características de experimento, sendo conduzida num ambiente simulado, em que os requisitos invariantes utilizados eram reais.

Neste estudo, avaliamos a percepção da utilidade de especificações formais com repre-

sentação gráfica. Para tanto, requisitos de um projeto existente[1] foram representados pelos sujeitos da pesquisa, com o objetivo de avaliar se especificações formais são úteis para o entendimento dos requisitos e para a detecção de falhas. Foi utilizado o formalismo da linguagem GIRL com a verificação lógica de Alloy [13].

Na condução do estudo, requisitos foram representados e verificados na linguagem GIRL. Cada requisito utilizado, tratava-se de uma invariante de baixa complexidade, por conter no máximo quatro entidades e quatro relacionamentos entre entidades.

Os sujeitos da pesquisa foram profissionais que atuam no desenvolvimento de software com diferentes níveis de experiência em análise de requisitos. Adicionalmente, três sujeitos tomaram parte no estudo, um docente da área de Matemática e também cliente em projetos de software, um estudante graduando em Ciências da Computação e um docente de Administração. Estes dois últimos participaram do estudo piloto.

O estudo transcorreu em dois momentos: primeiro, o estudo piloto para validar os instrumentos e o planejamento da pesquisa como um todo. Em seguida, a aplicação do estudo propriamente dito, dividido em sessões individuais para cada sujetito. Cada sessão passava por três etapas: (1) a ambientação, ou treinamento, (2) a utilização da linguagem GIRL para representar requisitos, e (3) por último, a avaliação ou julgamento da linguagem GIRL e, opcionalmente, algum comentário sobre a condução do estudo.

As atividades necessárias nas etapas de uma sessão do estudo foram guiadas por um questionário online (vide Apêndice C). Esse questionário serviu para coletar as informações requisitadas do participante, e também para registrar os tempos de permanência do participante nos módulos da ambientação e nas etapas de utilização da linguagem.

A qualquer momento, o participante teve livre acesso ao material disponível sobre a linguagem, como também pôde expor suas dúvidas ao observador, que possuía conhecimento sobre os conceitos e operacionalização da linguagem GIRL.

Em cada sessão do estudo, o observador tomou nota de: (1) dúvidas e comentários do participante sobre a linguagem GIRL, (QP2) expressões corporais do participante que denotem fadiga ou tédio, (3) verbalizações do raciocínio do participante no momento da modelagem das invariantes.

[1] Os requisitos do tipo invariantes utilizados foram extraídos de um projeto em andamento para desenvolvimento de um software de controle acadêmico numa instituição de ensino superior.

O estudo utilizou duas abordagens, uma quantitativa e outra qualitativa. Na abordagem qualitativa foram registradas observações que serviram de métricas para identificar dificuldades no aprendizado e utilização de GIRL (QP1), auxiliar no entendimento da utilidade percebida de GIRL (QP2) e identificar efetividade da notação (QP4), são elas:

- **ODIA e ODIU:** Dificuldades ou outras observações redigidas pelo sujeito na ambientação (ODIA) e utilização (ODIU).
- **ODVA e ODVU:** Dúvidas ou outras observações identificadas pelo observador na ambientação (ODVA) e utilização (ODVU).

Na abordagem quantitativa, foram coletados dados relacionados a variáveis que basearam as respostas das questões de pesquisa quanto ao nível de dificuldade na linguagem (QP1), a utilidade da linguagem (2), a precisão e cobertura dos requisitos representados em GIRL (QP3) e a efetividade da notação visual (QP4). Trabalhamos então com as seguintes variáveis e questões de pesquisa associada:

- Durante a ambientação:
  - **DA:** Quantidade de dúvidas reportadas por estrutura - QP1, QP2 e QP4.
  - **GE:** Grau de Entendimento em cada estrutura - QP1, QP2 e QP4.
- Durante a utilização da linguagem:
  - **TU:** Tempo de representação dos requisitos - QP1, QP2 e QP4.
  - **DU:** Quantidade de dúvidas reportadas por estrutura - QP1, QP2 e QP4.
  - **CR:** Corretude dos requisitos representados em GIRL - QP3.
  - **GU:** Grau de utilidade da ferramenta GIRL - QP2 e QP4.

5.2 Protocolo de Aplicação do Estudo

O protocolo de aplicação do estudo empírico descreve o procedimento para a realização do estudo. No presente estudo, o protocolo foi único para todos os participantes, exceto para os participantes do estudo piloto e para o participante do perfil de cliente. No protocolo foram previstas as seguintes atividades:

Escolha dos Sujeitos da Pesquisa

Dois sujeitos participaram do estudo piloto, um estudante de graduação e um profissional de administração com atuação em pesquisa acadêmica nos últimos 18 meses. Eles foram submetidos ao procedimento de aplicação da pesquisa com os artefatos iniciais em ambiente distinto do que seria utilizado no estudo final.

Para sujeitos do estudo final, foram selecionados profissionais que atuam, ou atuaram nos últimos 18 meses, em projeto de desenvolvimento de software, executando atividades de elicitação de requisitos, independente do nível de formalismo do processo de desenvolvimento de software utilizado. Por limitações de recursos na pesquisa, foram convidados apenas analistas de tecnologia da informação de uma instituição de ensino superior e mais um docente da área de matemática e cliente de projetos de software na mesma instituição.

Convite

Os sujeitos foram convidados por meio de um e-mail (vide Apêndice A), descrevendo resumidamente a pesquisa e fornecendo as principais características esperadas para um participante. O convidado que aceitou, preencheu o questionário online de convite (vide Apêndice B), que continha uma descrição mais detalhada da pesquisa, e questões para coletar as principais característica do participante: experiência com engenharia de requisitos, conhecimento em lógica e formação acadêmica. Além disso, o questionário forneceu sugestões de agendamento para a escolha da sessão de aplicação.

Esse agendamento inicial previa dois participantes por turno. No entanto, após a aplicação do estudo piloto, optamos por agendar apenas um participante por turno. Isso por constatarmos que o tempo de uma sessão poderia se estender afetando o agendamento de outro participante. E também percebemos que duas sessões por turno geraria fadiga no observador, comprometendo a qualidade das observações. Então, em acordo com o participante, ficou reagendado apenas um participante por turno.

Cada sujeito do estudo, participou de uma aplicação, ou operacionalização, do estudo de forma individual num mesmo ambiente. O observador presente em cada aplicação fez as anotações na ficha individual de cada participante e pôde realizar intervenções facilitadoras.

Setup

O ambiente físico foi único e em condições idênticas para todos os sujeitos. Para isso, seria necessário checar e anotar o resultado da vistoria em formulário específico para *setup* associado a cada sujeito. Foi anotada qualquer variação das seguintes características:

1. **Horário** agendado e confirmado
2. **Iluminação** completa
3. **Climatização** com a possibilidade de ajustes para o conforto de cada sujeito.
4. **Quadro** com os gráficos das estruturas e lembretes explicativos mais críticos para a utilização da linguagem GIRL, ambos dispostos de tal forma que as estruturas mais críticas sejam visualizados mais facilmente - Figura 5.2 exposta na Figura 5.1.
5. **Cola**[2] exemplos do quadro, impressos numa única folha como - Figura 5.2.
6. **Computador** que será utilizado pelo participante conectado à internet.
7. **Aplicação do Eclipse Project**[3] aberta na *workspace* individual de cada participante, para o ser utilizada após o treinamento.
8. **Tutorial** - Arquivo PDF aberto e versão impressa disponível para fácil acesso.
9. **Questionário online** - aberto em navegador de internet na página inicial.
10. **Material Impresso para o Observador -** para uso ou ser entregue ao participante posteriormente: Ficha para Anotações, Texto com os requisitos para a etapa de utilização, Gabarito (representação gráfica em uma folha da solução para os requisitos propostos na utilização), Questionário impresso (se necessário caso ocorra falha de conexão da internet).
11. **Canetas** para uso do participante e do observador.
12. **Suprimentos** de alimentos e água.

[2]Na região onde o estudo foi realizado, o termo 'cola' referencia material indevido de consulta usado durante exames escolares. No caso da pesquisa, a cola é um artefato de uso permitido e incentivado.

[3]Disponível em: http://www.eclipse.org/eclipse/

Figura 5.1: Sala da Aplicação do Estudo

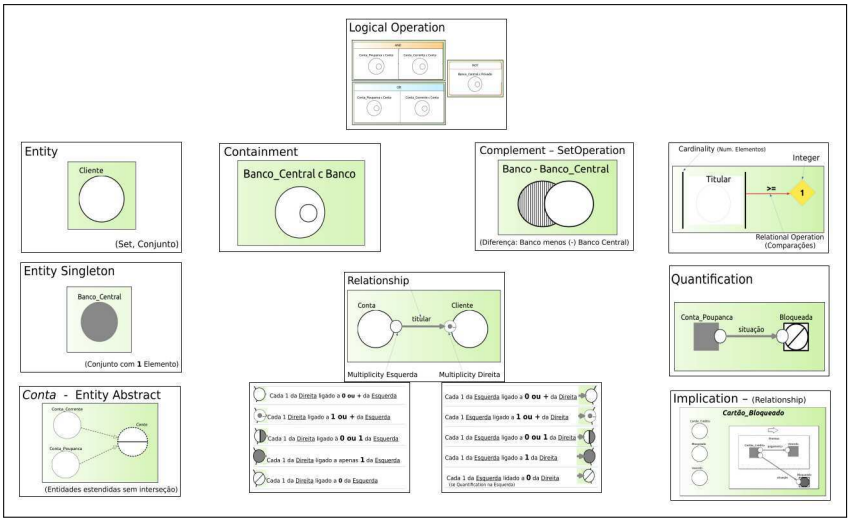

Figura 5.2: Quadro - Painéis com Estruturas de GIRL

Introdução

Observador anota a hora do início da sessão e explica ao sujeito o procedimento a ser realizado durante o estudo com a ferramenta GIRL para modelagem gráfica de requisitos estáticos do tipo invariante.

Ambientação ou Treinamento

A ambientação, ou treinamento, será dividida em módulos e enfoca uma ou mais estruturas da linguagem por vez. Optamos por essa divisão para poder aplicar um conjunto de conhecimento pequeno por vez, e assim facilitar o aprendizado e a retenção, promovendo intervalos para que o sujeito pudesse pensar e avaliar o seu entendimento. Em cada módulo procedemos:

1. Exibição do vídeo explicativo da(s) estrutura(s) contendo um pequeno exemplo.

2. Após a exibição do vídeo, o sujeito é convidado a responder questões sobre o seu grau de entendimento de cada estrutura abordada no módulo e a relatar alguma dificuldade encontrada.

Abaixo estão listados os 6 módulos com as respectivas estruturas e invariantes abordadas para um contexto de uma aplicação bancária:

Módulo 1: Explicação das estruturas *Invariant*, *Entity*, *Containment*. A Figura 5.3 exibe o modelo elaborado no vídeo do módulo[4] para as seguintes invariantes:

- *Uma Conta pode ser Conta Corrente ou Conta Poupança.*
- *A situação de uma Conta pode ser Aberta, Bloqueada ou Fechada.*
- *O Banco Central está contido em Banco.*

Módulo 2: Explanação das estruturas *SetOperation*, *Cardinality*, *Integer*, *Relational Operation*. A Figura 5.4 exibe o modelo construído no vídeo do módulo[5] para as seguintes invariantes:

[4]Vídeo do Módulo 1: https://drive.google.com/open?id=1bWDlBwQvsE7t7CFSDfx7Yuv2XolDBj9q
[5]Vídeo do Módulo 2:https://drive.google.com/open?id=1QKjdrlhIFAe-VmMJEdpi9iWpCzbDgPBB

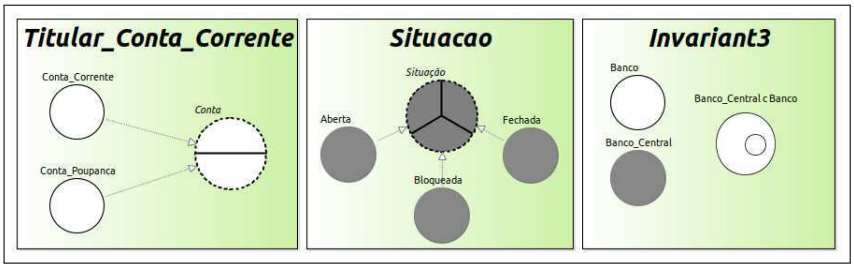

Figura 5.3: Módulo 1 - Invariantes Apresentadas no Vídeo

- *Representação de Banco - Banco Central.*
- *A quantidade de Titulares é maior ou igual a um.*

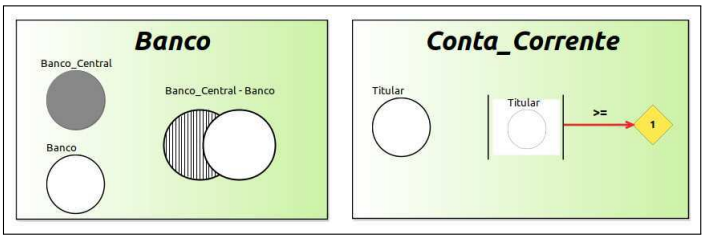

Figura 5.4: Módulo 2 - Invariantes Apresentadas no Vídeo

Módulo 3: Exposição da estrutura *Relationship* e da Verificação. A Figura 5.5 exibe o modelo elaborado no primeiro vídeo do módulo[6] e verificado no segundo vídeo do módulo[7], para a seguinte invariante:

- *Uma Conta tem um ou mais Clientes como titular.*

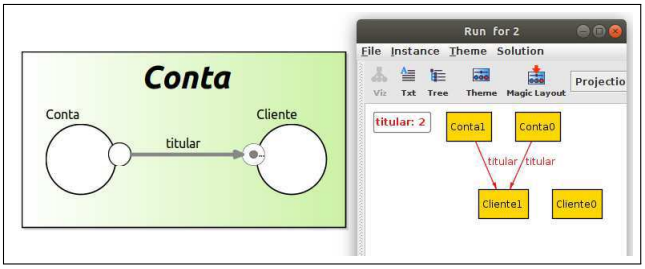

Figura 5.5: Módulo 3 - Invariante e Verificação Apresentadas no Vídeo

[6]Vídeo 1 do Módulo - 3: https://drive.google.com/open?id=1DZ6Anc2teJuZxiAz69XQQVzJGfv0M61W
[7]Vídeo 2 do Módulo - 3: https://drive.google.com/open?id=1yXSqFdyGDNAVI0FcpANRdBduQ7Wjyocp

Módulo 4: Apresentação da estrutura *LogicalOperation*. A Figura 5.6 exibe o modelo elaborado no vídeo do módulo[8] para a seguinte invariante:

- *Uma Conta Corrente está contida em Conta E uma Conta Poupança está contida em Conta.*

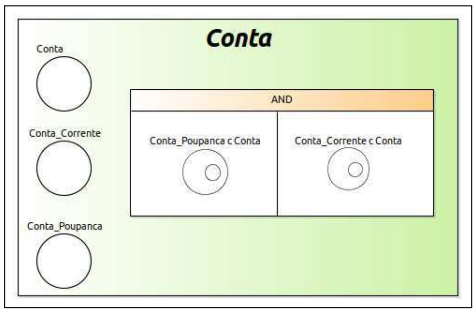

Figura 5.6: Módulo 4 - Invariante Apresentada no Vídeo

Módulo 5: Exposição da estrutura *Quantification*. A Figura 5.7 exibe o modelo construído no vídeo do módulo[9] para a seguinte invariante:

- *Toda Conta tem apenas uma Situação.*

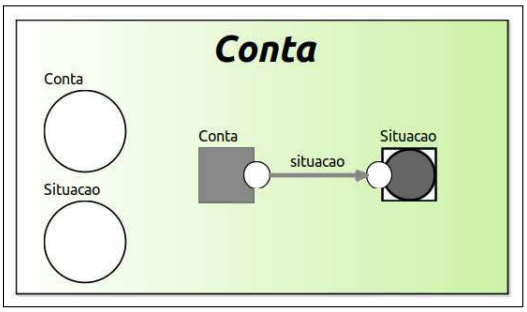

Figura 5.7: Módulo 5 - Invariante Apresentada no Vídeo

Módulo 6: Demonstração da estrutura *Implication* (com validação). A Figura 5.8 exibe o modelo elaborado no vídeo do módulo[10] para as seguintes invariantes:

- *Um Cartão de Crédito deve ter apenas uma Situação que pode ser ou Ativo ou Bloqueado*

[8] Vídeo do Módulo - 4: https://drive.google.com/open?id=1ujo94S2UTpCbPOKlmKcSp6qC8xGT1Uk1
[9] Vídeo do Módulo - 5: https://drive.google.com/open?id=1xgcsh1lhk47I3OABd_n3ZoDg80cRtdpO
[10] Vídeo do Módulo - 6: https://drive.google.com/open?id=1uk_enN69kUtMBYYKweHGZEwAmmWDu_V7

- *Um Cartão de Crédito deve ter apenas um indicador de Pagamento indicando se está ou Vencido ou Pago o Mínimo.*
- *Se um Cartão de Crédito está com o pagamento Vencido, então a Situação dele deve ser Bloqueado.*

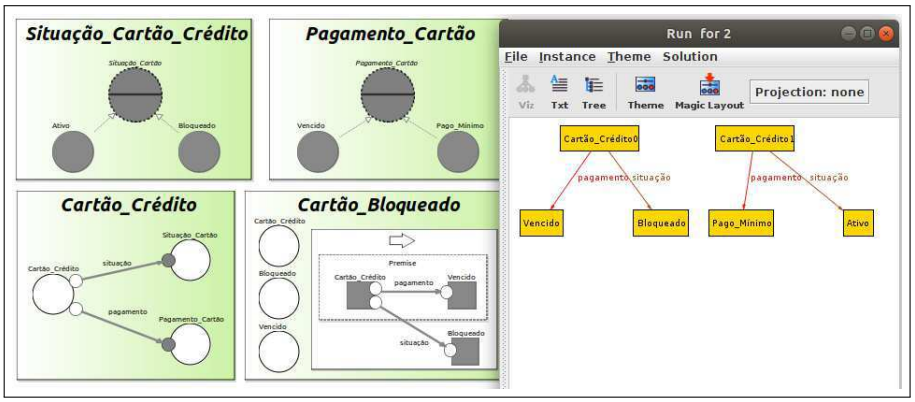

Figura 5.8: Módulo 6 - Invariantes e Verificação Apresentadas no Vídeo

Utilização

Logo após a ambientação, o sujeito foi convidado a modelar quatro conjuntos de requisitos, no contexto de uma aplicação de controle acadêmico da pós-graduação. Para modelar esses requisitos, não era necessário um conhecimento prévio da aplicação. Os requisitos foram entregues em folha impressa e com opção de abrir arquivo PDF na pasta *modelos.girl* na área de trabalho do computador do participante. Cada conjunto de requisito continha 2 invariantes em linguagem natural para serem modeladas em GIRL:

Conjunto de Requisitos - 1: Era esperado o uso das estruturas *Entity* e *Entity Abstract* nas seguintes invariantes:

- *Um Aluno deve ser ou Regular ou Especial.*
- *Um Docente deve ser ou Permanente, ou Colaborador, ou Visitante.*

Conjunto de Requisitos - 2: Era esperado o uso das estruturas Entity, Relationship e Multiplicity (ONE e SOME) nas seguintes invariantes:

- *Um Aluno deve ter uma Área de Concentração.*

- *Um Docente deve ser ou Permanente, ou Colaborador, ou Visitante.*

Conjunto de Requisitos - 3: Era esperado o uso das estruturas *Entity*, *Entity Abstract*, *Relationship* e *Multiplicity* (ONE) nas seguintes invariantes:

- *Um Curso pode ser ou Doutorado ou Mestrado.*
- *Um Aluno só pode cursar um Curso.*

Conjunto de Requisitos - 4: Era esperado o uso das estruturas *Entity*, *Relationship*, *Multiplicity* (expressão relacional), *Quantification* e *Implication* nas seguintes invariantes:

- *Um Aluno pode ter até dois Docentes como orientadores.*
- *Se um Aluno cursa Doutorado, então ele deve ter um ou mais Docentes orientadores.*

Para cada conjunto de requisitos:

1. O sujeito foi convidado a modelar cada requisito entregue impresso e com opção de abrir arquivo PDF na pasta GIRL-Treinamento na área de trabalho do computador do participante. O requisito continha invariantes em linguagem natural e requeria o uso de estrutura(s) recém-aprendidas na modelagem em GIRL.
2. Durante a modelagem, caso desejasse, o sujeito realizaria os seguintes passos da Verificação de modelo criado:
   (a) Salvar o gráfico do modelo (foto) identificado.
   (b) O participante, junto com o observador, procede a verificação do modelo. A atuação do observador visou apenas facilitar a chamada das transformações. Como as chamadas não estavam automatizadas, buscamos evitar a perda de tempo com dificuldades para encontrar as opções no menu de transformações, o que não era o foco do estudo.
   (c) O participante navega pelas instâncias geradas.
   (d) O sujeito considera necessário refinar o modelo?

Sim: retornar para o editor gráfico da ferramenta, fazer os ajustes que julgar necessários, realizar a verificação e repetir o procedimento até considerar o modelo finalizado.

Não: o modelo do módulo foi considerado finalizado. O observador auxilia o sujeito a salvar o gráfico do modelo final para o conjunto de requisitos identificado pela numeração. O sujeito carregava o arquivo no questionário online.

Comparação Entre os Modelos

Após a etapa de Utilização, o observador entregou ao participante uma folha impressa com uma solução da modelagem dos requisitos em GIRL (ver Figura C.15), podendo acessar o *link* do modelo entregue e também o vídeo com a solução sendo construída.

Em seguida, o participante foi convidado a realizar uma comparação entre o seu modelo e a solução proposta e registrar algum comentário que julgasse relevante.

Avaliação da Linguagem GIRL

Nessa etapa o participante relatou o que considerou de positivo e de negativo na linguagem GIRL buscando abordar qual estrutura foi mais fácil ou difícil de utilizar, se o mecanismo de verificação foi útil, ou não, para a detecção de inconsistências no modelo e se o editor gráfico foi satisfatório, ou não.

Avaliação da Condução do Estudo

O sujeito é convidado a responder/preencher questões com observações gerais sobre o estudo (treinamento, questionário, condições).

5.3 Aplicação do Estudo

O estudo foi aplicado entre os dias 12 de julho de 2018 e 09 de agosto de 2018 nas dependências da Universidade Federal de Campina Grande - UFCG - Campus Campina Grande. Duas sessões, para o Estudo Piloto, ocorreram nos dias 12 e 24 de julho em uma sala que abriga

analistas de tecnologia da informação da instituição. Foram dois os participantes dessa primeira fase: um estudante de graduação em Ciências da Computação e um profissional mestre em Administração.

Durante o estudo piloto, os seguintes pontos de modificações foram detectados:

1. Permitir que o participante tenha acesso aos exemplos modelados na Ambientação;
2. Fracionar a página do questionário online na etapa de ambientação, dispondo uma por módulo para coleta de tempos de permanência;
3. Disponibilizar material gráfico de fácil acesso da visão;
4. Realizar a modelagem em um mesmo arquivo .girl;
5. Melhoria dos textos dos requisitos propostos para modelagem em GIRL;
6. Agendar apenas uma aplicação do estudo por turno para permitir mais tempo por sessão e evitar fadiga do observador;
7. Enfatizar para o participante a liberdade de realizar consultas e questionamentos durante o estudo.

Um e-mail convite foi enviado para analistas de tecnologia da informação da UFCG do setor de desenvolvimento de software. Um dos convidados questionou se interferiria nos resultados o fato dele não utilizar um processo de levantamento e gestão de requisitos mais formalizado.

Dez analistas de tecnologia da informação atenderam ao convite e responderam o questionário de caracterização do participante e sugeriram data e hora para agendamento da sua sessão. Outro participante da área da Matemática foi convidado para participar do estudo, em caráter especial, para fornecer uma análise do ponto de vista de um entendido das teorias matemáticas utilizadas na ferramenta GIRL.

Foi utilizada uma sala de aula (vide Figura 5.1) para a aplicação do estudo, contendo todos os itens dispostos na Seção 5.2. Entre 31 de julho e 09 de agosto de 2018, o estudo foi aplicado individualmente, como demonstram as Figuras 5.10 e 5.11, com a presença do observador que é a autora deste trabalho. Os horários agendados para os participantes estão dispostos na Tabela 5.1.

Tabela 5.1: Agendamento dos Participantes

*	Agendamento						*
Hora/Data	Terça 31/07	Quarta 01/08	Quinta 02/08	Sexta 03/08	Segunda 07/08	Quarta 09/08	Quinta 10/08
08:00						P03	P10
09:00			P05	P07	P08	P03	P10
10:00	P01		P05	P07	P08		
11:00	P01						
14:00	P02						
15:00	P02		P06		P09		P11
16:00		P04	P06		P09		P11
17:00		P04					

5.3.1 Características dos Participantes

Os sujeitos do estudo, também denominados participantes, foram caracterizados quanto à experiência em Engenharia de Software, conhecimento em lógica e formação acadêmica. A Tabela D.1, nos Apêndices, apresenta essas características por participante, inclusive dos participantes do estudo piloto.

As características dos participantes que são profissionais de desenvolvimento de software (10) estão apresentados nos histogramas da Figura 5.9. Verificamos que a maioria tem experiência considerável em requisitos (de 6 a 10 anos), tem conhecimento em lógica de intermediário a avançado e tem mestrado ou pós-graduação. Essas características foram consideradas na análise dos dados do estudo empírico.

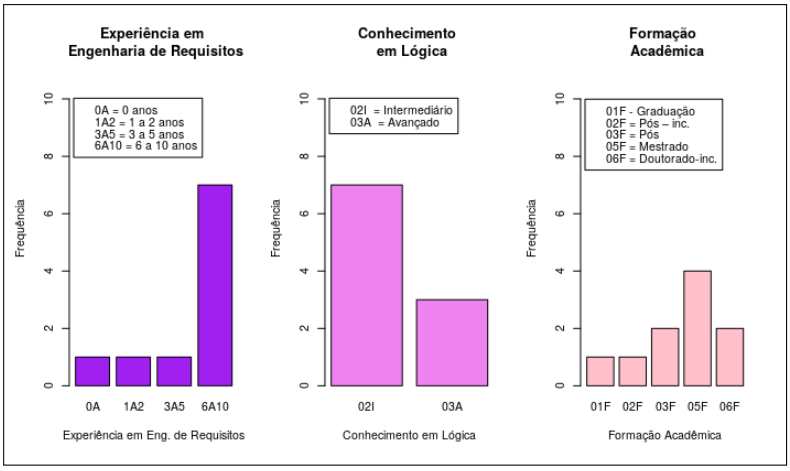

Figura 5.9: Histogramas das Características dos Participantes

5.3.2 Avaliação do Estudo

Os participantes realizaram uma avaliação do estudo, que está registrada nos Apêndices, na Tabela D.12. Alguns pontos de melhoria foram detectados, como a possibilidade de gravação das interações no computador e das falas do participante. O tempo foi apontado como um fator crítico pois poderia se alongar dependendo do participante. Ainda nessa avaliação, um participante relatou que a implicação e quantificadores deveriam ser simplificadas.

No mais, os artefatos, em geral, a estratégia de condução com a apresentação dos vídeos para a ambientação na linguagem, além do material ilustrativo, foram considerados relevantes para o estudo.

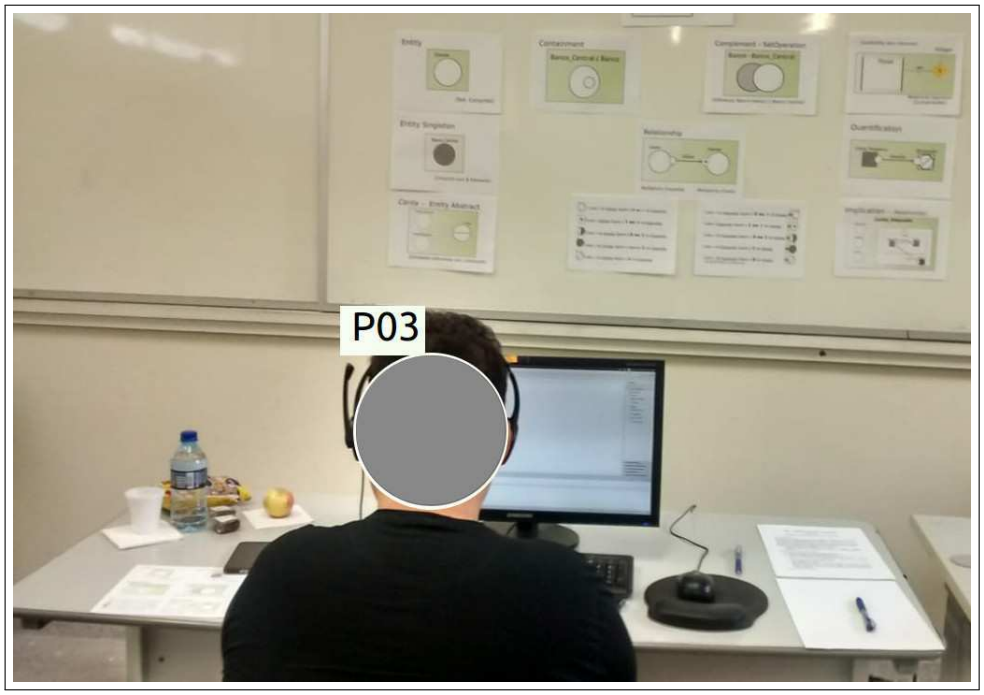

Figura 5.10: Participante P03 Assistindo Vídeo

Figura 5.11: Participante P06 Assistindo Vídeo

Capítulo 6

Análise e Discussão dos Dados

O estudo foi aplicado em 11 sessões individuais, uma para cada participante, que denominamos P01, P02, ..., P10 e P11. Os dados coletados foram classificados e avaliados. Detectou-se a necessidade de homogeneização do grupo de participantes, assim, para a análise quantitativa descritiva, foram considerados apenas os participantes que são profissionais de desenvolvimento de software. Com isso, os dados do participante P11, que não é profissional de software, não foram considerados nessa etapa da análise.

Conforme descrito na Seção 6.2.4, que discute a modelagem do Conjunto de Requisitos 4, os tempos de utilização dos participantes P02 e P04 também não foram considerados na análise apenas no que tange ao tempo de utilização para esse conjunto de requisitos. Isso se deu por causa de falhas na verificação, o que tornou os tempos de utilização não representativos para o estudo. Para a análise qualitativa, os dados desses dois participantes foram utilizados.

As observações anotadas durante a sessão de cada participante foram digitadas e classificadas em termos de dúvidas e achados. Agregamos a esses dados as dúvidas ou pontos negativos que o participante registrou no questionário. Os registros das questões abertas do questionário foram agregados às observações e consideradas como resumo de dúvidas e achados, ou até mesmo, para novo registro, caso não tenha sido registrado nas observações, resultando na Tabela D.4. A partir dessa tabela, as dúvidas foram totalizadas e classificadas quanto ao momento do seu registro: na Ambientação ou na Utilização (vide Tabela D.5). Os dados utilizados para compor essas duas últimas tabelas encontram-se no Apêndice G.

A seguir apresentamos a análise dos dados e a discussão das constatações nessa aná-

lise, divididas nas seguintes seções: (1) avaliação da linguagem GIRL sobre as estruturas, a verificação, a representação gráfica e a avaliação geral da linguagem; (2) análise dos modelos elaborados, uma para cada conjunto de requisitos; (3) análise quantitativa das medidas de tempo de utilização, características do participante, graus de entendimento fornecidos e quantidade de dúvidas observadas; e (4) discussão dos resultados, direcionada pelas respostas às questões de pesquisa.

6.1 Avaliação da Linguagem GIRL

Ao longo da etapa de ambientação na linguagem GIRL, os participantes puderam registrar no questionário o grau de entendimento de cada estrutura abordada nos vídeos dos seis módulos. Esses dados constam na Tabela D.2. Enquanto o participante assistia ao vídeo, o observador tomava notas das dúvidas e expressões verbais ou corporais do participante, possibilitando realizar uma análise triangulada para descobrir registro tendencioso à supervaloração ou contradição no grau de entendimento registrado. Além disso, o momento da utilização forneceu pistas, ainda que discretas, para detectar se o entendimento registrado estava de acordo com a realidade. Tais pistas são ditas discretas, quando consideramos a possibilidade da volatilidade do novo conhecimento com o passar do tempo entre a ambientação e a utilização.

Com base nesses dados, avaliamos a linguagem GIRL com respeito às suas estruturas, ao mecanismo de verificação, à representação gráfica e à avaliação dos sujeitos da pesquisa. A meta é descobrir dificuldades no aprendizado e na utilização da linguagem pela observação e pelo julgamento dos sujeitos, bem como, identificar a utilidade da verificação e a efetividade da representação gráfica.

6.1.1 Estruturas da Linguagem

Sobre o entendimento das estruturas da linguagem GIRL na opinião dos participantes, realizamos uma análise descritiva dos histogramas das estruturas na Figura 6.1. Pela tendência dos gráficos, observamos que o grau de entendimento foi elevado (nos graus mais altos 7 a 10) em praticamente todas as estruturas. Para *Invariant*, *Implication*, *Quantification* e *Set Operation*, a tendência horizontalizou mais. Dessas, a *Implication* e a *Quantification* foram consideradas as mais difíceis durante a utilização da linguagem, ou seja, a impressão inicial

dos participantes na ambientação foi confirmada após a utilização.

As dúvidas na ambientação e utilização foram classificadas e totalizadas por estrutura da linguagem GIRL. A Figura 6.2 exibe os totais de dúvidas encontrados. Nos casos em que a dúvida estava entre duas estruturas, atribuímos meia dúvida (0,5) para cada estrutura envolvida. Ao verificar que a maior parte das estruturas que não foram necessárias na utilização não tinham registro de dúvidas nessa etapa (*integer, logical operation, relational operation e set operation*), percebemos que os dados não se contradizem, ainda que as dúvidas em *cardinality e containment* surgiam na escolha entre outra estrutura.

Comparamos o grau de entendimento com a quantidade de dúvidas na ambientação e utilização, buscando identificar algum fator que justificasse o cenário encontrado. Detalhes dessas análises estão na Tabela D.7. Identificamos que, em estruturas com graus de entendimento mais baixos, a quantidade de dúvidas foi maior. A exceção foi a *relationship* que, por encerrar outros conceitos como a *multiplicity* e *quantification*, foi a estrutura que apresentou mais dúvidas, e, mesmo assim, obteve um grau de entendimento elevado e foi avaliada como umas das mais fáceis de utilizar. Em outras estruturas, percebemos que, na ambientação ou até na utilização, algumas dúvidas estavam relacionadas aos exemplos utilizados e que podiam ser expressos, ou estavam embutidos, em outras estruturas, como: *containment*, *logical operation* e *quantification*.

A Tabela D.8 exibe a avaliação dos participantes quanto as **estruturas** mais fáceis e as mais difíceis de utilizar. As estruturas consideradas mais fáceis foram: entity (entidade) e relationship (relacionamento). Alguns mencionaram a facilidade em invariante que, na prática, o conceito não era tão difícil quanto alguns pensaram na ambientação (ver histograma na Figura D.1). Apenas um considerou a implication (implicação) fácil.

Seis participantes mencionaram a dificuldade para utilizar a implicação, quatro a quantificação, três a cardinalidade e apenas um o *Containmet*. A dificuldade na implicação pode estar na pouca intuitividade na sua construção, como salienta P07, "*implicação não é intuitiva, pois até para inserir os dados é necessário se atentar para a ordem em que é feita*". E ainda sugeriu "*pensar numa expansão da ferramenta para geração de um código fonte base, já que demanda um tempo para especificar tudo e esse tempo poderia ser reaproveitado nas etapas posteriores do projeto*".

Na etapa de utilização da linguagem GIRL, a percepção da complexidade da implicação

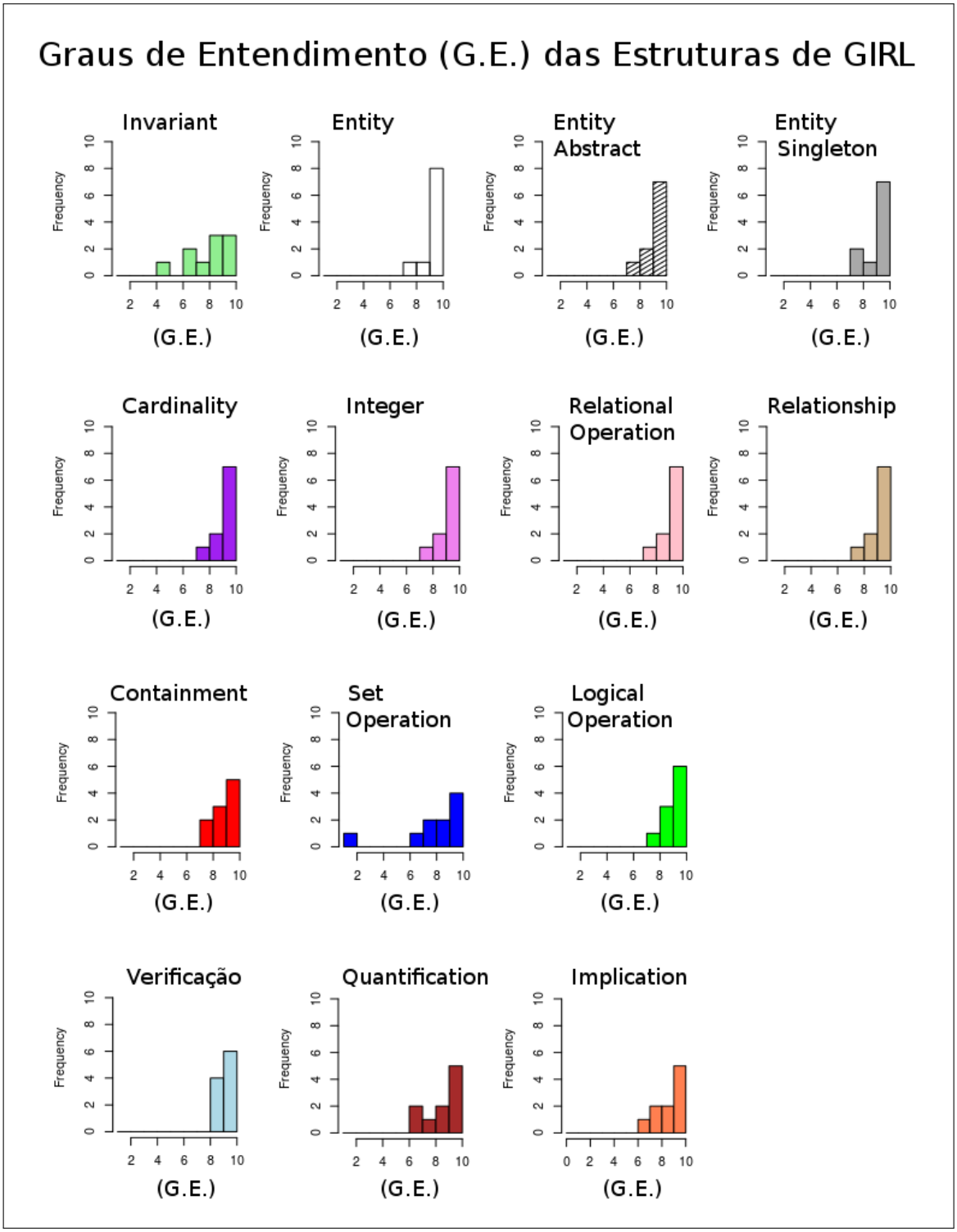

Figura 6.1: Histogramas dos Graus de Entendimento (G.E.) das Estruturas em GIRL

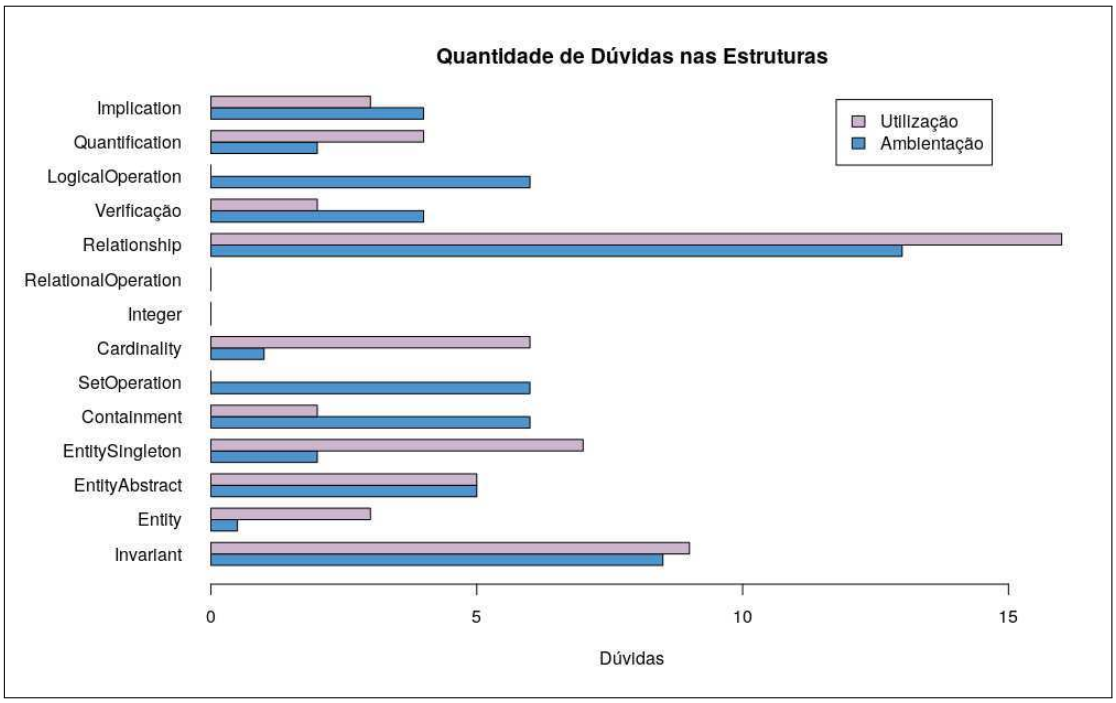

Figura 6.2: Histogramas das Dúvidas por Estrutura de GIRL

se deu já na leitura da segunda invariante do requisito 4, como demonstra a Tabela 6.4, na Seção 6.2.4. Possivelmente porque: (1) consideraram complexa a utilização da implicação na exibição do vídeo, ou (2) consideram que o conceito de implicação é complexo.

6.1.2 Avaliação da Verificação das Invariantes

As avaliações registradas pelos participantes quanto a verificação na linguagem GIRL estão na Tabela D.9. Praticamente todos consideraram o mecanismo útil para validar, verificar, executar, debugar, descobrir inconsistências. Um deles destacou a forma incremental de verificação. Outro mencionou como a simplicidade da representação a torna útil até para o cliente em projetos de software.

Uma das questões de pesquisa busca investigar a utilidade percebida pelos sujeitos quanto ao mecanismo de verificação de requisitos da linguagem GIRL. Para amenizar a possibilidade de contradições ou supervaloração, registramos as reações positivas dos participantes ao mecanismo de verificação, durante a etapa de utilização, e que não foram registradas no questionário. Essas observações estão descritas na Tabela 6.1 com o relato das falas do próprio participante ou da percepção do observador.

As expressões verbalizadas durante a utilização do mecanismo de verificação estão registradas na Tabela I.1 do Apêndice I, que exibe no final a avaliação do próprio participante. Percebe-se que o fato de se "rodar" para "ver" as instâncias foi um diferencial catalisador para a descoberta de falhas nos modelos. Percebia-se interesse e receptividade para o uso da verificação com *feedback* gráfico. Mais um ponto para a representação gráfica da linguagem, que será avaliada na próxima seção.

Tabela 6.1: Reações Positivas à Verificação

*	Expressões Positivas	*
Part.	**Descrição**	
P01	"A cereja do bolo foi a Verificação, ver os negocinhos aparecendo! Simulação no final é bem interessante, se você modelou errado!"	
P02	“Gostei mais do validador! De tudo! É fantástico!" "Validar... muito legal!"	
P05	"É massa, vice! Principalmente a parte para testar! Até para o cliente fica melhor de mostrar! Análise é massa!"	
P06	"Só as e-num." Disse quando só apareceram apenas as entidades singletons nas instâncias. "Essa tela é similar a algo que vi... acho que em teoria da computação... algo de grafos... até que eu gostava de fazer! Era parecido. Dava as possibilidades."	
P07	"O mais legal foi poder criar instâncias e ver o modelo sendo aplicado."	
P07	"Ele gera todos os casos de uso, né?" "Você pode até tirar uns *prints* e considerar como Caso de Uso (sugerido nos registros da avaliação de GIRL).	
P09	"É bacana ver as possibilidades!"	

6.1.3 Avaliação da Representação Gráfica

Editor Gráfico

Cada participante foi convidado a registrar sua avaliação do editor gráfico da linguagem GIRL em termos de pontos negativos e positivos. A maioria considerou que o editor gráfico foi satisfatório. Destacamos os seguintes pontos positivos mencionados: simples, poucas funcionalidades, fluido, funcional, agradável e fácil de usar, fácil para iniciantes, usabilidade, intuitivo, necessita de pouco treinamento.

Por outro lado, os pontos negativos abordaram: dificuldade para desenhar a implicação, que requeria atenção na ordem em que se colocava os elementos; dificuldade para redimensionar alguns componentes gráficos; e pouco uso de cores.

Indicativos da Efetividade da Representação Gráfica

A representação gráfica ou, informação visual, é considerada mais efetiva do que a representação textual para a comunicação. Isso se dá por sua capacidade de precisão e síntese, além de ser mais facilmente lembrada, devido à superioridade da imagem em relação ao texto [23]. Durante a utilização da linguagem GIRL, foram registrados alguns episódios que exemplificam essa efetividade da representação gráfica do modelo e da verificação, que estão descritos na Tabela D.11.

Ficaram evidentes: (1) a associação imediata do fatiamento das entidades abstratas que estendiam outras entidades (P02, P09, P10 e P11), analogia com uma pizza fatiada; (2) o significado distinto das cores cinza para as entidades únicas ou multiplicidade das relações; (3) a inferência do significado dos símbolos gráficos com a teoria dos conjuntos; (4) a maior capacidade de entendimento da representação gráfica sem muito esforço de "leitura", quando o participante P07 estava representando outra invariante (focado em uma área próxima do gráfico), detectou por associação visual que na invariante anterior havia colocado um símbolo errado (uma multiplicidade errada); (5) a satisfação de receber informação visual na verificação do modelo. Tais constatações estão descritas no trabalho de Moody [23] como princípios para projetar uma notação visual efetiva, ou como argumentos motivadores para correto *design* na construção dessas notações.

6.1.4 Avaliação Geral da Linguagem

A avaliação geral da linguagem foi fornecida pelos participantes após registrarem os pontos positivos e negativos da linguagem. Essa avaliação constou do grau de utilidade da linguagem, como demonstra a última linha Tabela D.2. O histograma da Figura 6.3 denota uma tendência a uma avaliação geral positiva da linguagem. Ainda na Tabela 6.2, algumas expressões positivas dos participantes corroboram com a avaliação descritiva do gráfico.

Tabela 6.2: Reações Positivas a GIRL e Verificação

*	Expressões Positivas	*
Tipo	**Part.**	**Descrição**
Sobre Girl	**P02**	"O soft em si tá até muito bom!"
	P03	"Gostei, de verdade!" Quando perguntei se gostou da linguagem.
	P04	"Esse ferramenta é boa pra dar aula!"
	P05	Em geral gostei de usar a ferramenta (Registro dos pontos positivos).

Tabela 6.2: (continuação)

*	Expressões Positivas	*
Tipo	**Part.**	**Descrição**
	P07	"Gostei dessa ferramenta. Nunca tinha usado uma ferramenta para requisitos."

6.2 Modelos Elaborados

Nas próximas seções avaliamos os modelos em GIRL elaborados para cada invariante dos quatro conjuntos de requisitos propostos durante a etapa de utilização da linguagem GIRL. Os passos para elaborar as soluções estão descritos no Apêndice F, onde cada invariante corresponde a uma tabela e cada coluna da tabela, um participante. Para a descrição dos passos, utilizou-se uma codificação para diminuir o texto explicativo, cujos códigos e significados estão descritos na Tabela J.1 do Apêndice J. Os gráficos dos modelos elaborados por cada participante, e que foram adicionados ao questionário, estão dispostos no Apêndice E, e estão agrupados por conjunto de requisitos. Apenas três gráficos dos modelos mais relevantes nas discussões constam no corpo da respectiva seção.

As próximas seções dissertam sobre a corretude dos modelos elaborados e sobre aspectos relevantes das decisões de modelagem tomadas pelos participantes e possíveis causas. Com isso tentamos levantar evidências para responder à questão de pesquisa sobre a precisão e cobertura dos requisitos representados durante a utilização da linguagem GIRL (QP2).

Cada seção foca um dos quatro conjuntos de requisitos e inicia com a apresentação das invariantes do conjunto, seguido da análise mencionada. Um resumo das constatações da análise se encontra na Tabela 6.8 que compila os resultados da análise.

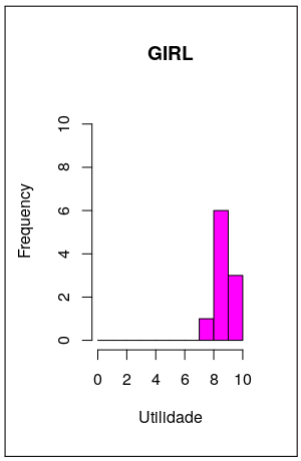

Figura 6.3: GIRL - Grau de Utilidade

6.2.1 Modelos do Conjunto de Requisito 1

Invariantes propostas para o conjunto de Requisitos 1:

- Um Aluno deve ser ou Regular ou Especial.

- Um Docente deve ser ou Permanente, ou Colaborador, ou Visitante.

Para essas invariantes, esperava-se a utilização de Entity, Entity Abstract e Extends (a extensão) entre entidades.

As figuras E.1 à E.11, apresentam a representação gráfica do Conjunto de Requisitos 1 dos onze participantes e as tabelas F.1 e F.2 contêm os passos que cada participante empreendeu para construir seu modelo gráfico das duas invariantes propostas, além das dúvidas do participante e as intervenções do observador.

Cinco participantes (P03, P04, ,P05, P06 e P09) optaram pela representação semelhante à *enumeration* do Java e utilizaram duas entidades adicionais para representar os tipos de Alunos e Docentes (vide Figura 6.5). Em contrapartida, o participante P01 (vide Figura 6.4) elaborou um modelo sem entidades singleton.

O participante P09 modelou inicialmente Aluno e Docente com Singleton, mas retirou de Aluno com a seguinte justificativa: "Não faz sentido eu instanciar um discente sem saber que tipo ele é (se Regular ou Especial)"((vide Figura 6.6)). Outro participante (P08) cogitou a possibilidade de utilizar Entity Singleton, mas desistiu, optando por entidades não singleton.

O participante P11, por não ter experiência alguma em engenharia de requisitos, não tinha conhecimento prévio do conceito de classes abstratas ou de extensão entre classes do

UML. Isso produziu uma dificuldade inicial em realizar a extensão de entidades. Todos os demais participantes aplicaram com facilidade o conceito de entidade abstrata e de extensão.

Na segunda invariante do Conjunto de Requisitos 1, o participante P03 utilizou o mesmo nome de entidade para modelar o que chamou de Situação para Aluno e para Docente. O observador interviu apresentando a restrição da necessidade do uso de nomes distintos, caso contrário, na verificação, todas as situações seriam aplicadas tanto para Aluno como para Docente. A intervenção foi justificada para dar celeridade à utilização por haver construções distintas para as situações de Aluno e Docente, evidenciando apenas uma falha de identificação das entidades e não nos conceitos modelados.

Apenas dois participantes, P03 e P06, realizaram a verificação após a modelagem das duas invariantes. Talvez esse comportamento se deva à falta de costume de verificação de requisitos ou modelagem. Durante a verificação, o participante P06, acompanhando atentamente as instâncias, afirmou: "Só as e-num", quando uma instância exibiu apenas as entidades singletons.

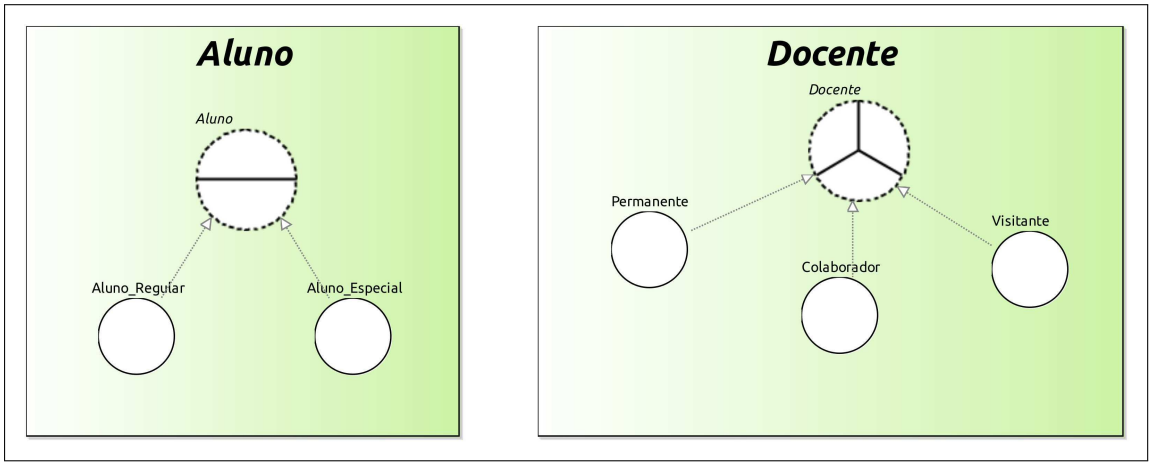

Figura 6.4: Participante 01 – Modelo Conjunto 1

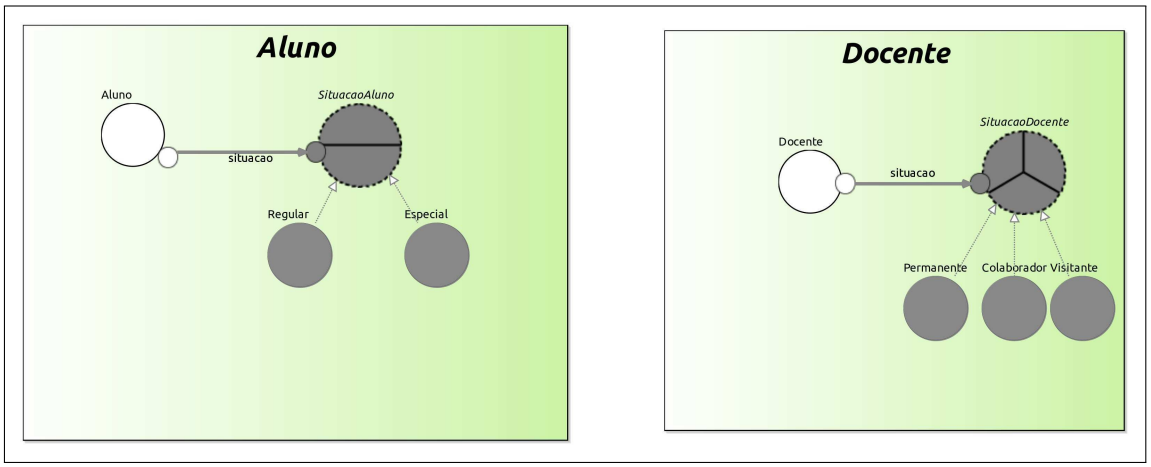

Figura 6.5: Participante 03 – Modelo Conjunto 1

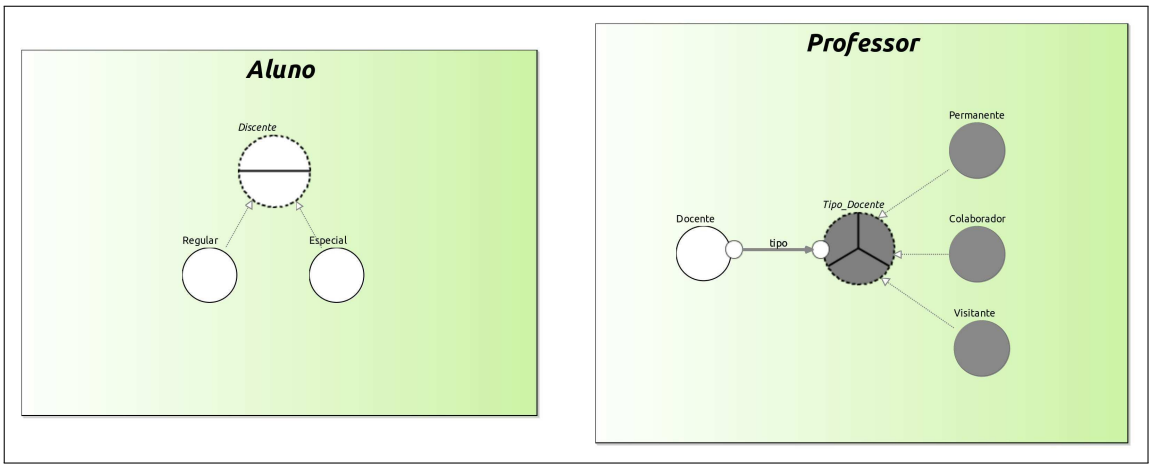

Figura 6.6: Participante 09 – Modelo Conjunto 1

6.2.2 Modelo do Conjunto de Requisito 2

Invariantes propostas para o conjunto de Requisitos 2:

- Um Aluno deve ter uma Área de Concentração.

- Um Docente deve ter uma ou mais Áreas de Concentração.

Nessas invariantes, seria necessário a utilização de Entity e Relationship com Multiplicity "apenas um" (ONE) na primeira invariante e "um ou mais" (SOME) na segunda.

Os modelos criados pelos onze participantes estão apresentados nas figuras E.12 à E.22. Nas tabelas F.3 e F.4 estão registrados os passos de cada participante para a construção da sua solução, suas dúvidas e as intervenções do observador.

A maioria dos participantes optou por representar as invariantes com relação e multiplicidade. Apenas um deles, no primeiro requisito, utilizou a estrutura Containment. No segundo requisito utilizou relação com multiplicidade.

Três participantes (P01, P04 e P11) cogitaram a utilização de Cardinality para representar as quantidades. Entretanto, todos eles optaram por Relationship com Multiplicity. Cinco participantes escolheram nomes das relationships com base na conotação de "ter", "possuir" a relação. Outros, identificaram as relações com nomes das entidades, "area_ docente", "area_ aluno". Apenas um escolheu um nome que expressava o significado da relação: 'pesquisa'.

O participante P08, representou o conceito do texto "Aluno deve ter uma Área..." com a estrutura Containment (vide Figura E.19). Uma provável associação entre o significado do "ter" do texto ao "conter" do Containment. O participante P01 também cogitou utilizar o Containemt, mas desistiu e seguiu com uma Relationship.

Em sua solução, o participante P10 introduziu três áreas de concentração singletons, Area1, Area2 e Area3, estendendo a entidade abstrata Area (vide Figura E.21). A partir daí criou as duas relações partindo de Aluno e Docente para Area. Indicação de resíduo do conceito de enumeration.

O sujeito P04 atribuiu multiplicity à entidade da esquerda das duas relações (vide Figura 6.7). Possivelmente, para prevenir erros, ainda que não havia indicação para isso no texto das invariantes. Entretanto, na etapa de comparação dos modelos, mencionou que sempre confunde as multiplicidades da direita ou esquerda, inclusive em UML.

Quatro participantes realizaram verificação (P03, P04, P06 e P07). A intenção seria encontrar nas instâncias geradas durante a verificação a comprovação de que o modelo estava

correto, como expressou o participante P03: "Quero um Docente com duas Áreas." E, ao encontrar se disse: "Satisfeito!"

O observador questionou o participante P04 se gostaria de verificar, ao que replicou com a seguinte pergunta: "Isso é para verificar, ou só no final?". Essa dúvida, fornece uma indicação de um padrão de comportamento de verificar, ou testar, apenas ao final de um conjunto maior de itens modelados. Informado que poderia verificar, iniciou a verificação das invariantes, no entanto, o fato de haver duas relações com mesmo nome partindo de entidades diferentes e apontando para uma mesma entidade, produziu uma inconsistência no Alloy, o que impossibilitou a verificação. A correção não foi sugerida pelo observador nesse momento.

Durante a verificação do seu modelo, o participante P06 identificou uma relação com o nome padrão "relationship" (sendo I a I-ésima relação da invariante) atribuído na sua criação, dizendo: "Eita.... 'relationship' aqui!", percebendo que não tinha nomeado o relacionamento que apontava para Docente. Em seguida, renomeou a relação para "areaConcentracao", voltou a verificar e finalizou dizendo: "Enfim, tá ok!".

Um fato interessante reforçou a evidência de que a representação gráfica, por si, auxilia a compreensão e a verificação do modelo, por meio da "leitura" dos símbolos visuais. Isso ocorreu quando o participante P07 estava modelando outra invariante. Enquanto atribuía a multiplicidade da relação entre Docente e Área de Concentração, dizia: "Docente deve ter 'uma ou mais' Área de Concentração!". Nesse momento exclamou: "Peraí...", pois percebera que, no lugar da multiplicidade SOME (uma ou mais) em Área de Concentração, na relação Aluno 'pesquisa' Área de Concentração, deveria ter a multiplicidade ONE (apenas uma). O interessante é que ele descobriu a falha numa relação quando construía a outra, que estava numa invariante diferente, mas perto. O sinalizador visual do erro foi que havia duas multiplicidades SOME (uma ou mais) em Área de Concentração para as duas relações, quando, na verdade, deveria ser apenas na relação com o Docente, enquanto que, na do Aluno deveria ser ONE (apenas uma). Com a descoberta, corrigiu-se a multiplicidade e, só então, iniciou-se a verificação. Como não identificou instâncias com incoerência, considerou o modelo válido dizendo: "Beleza!".

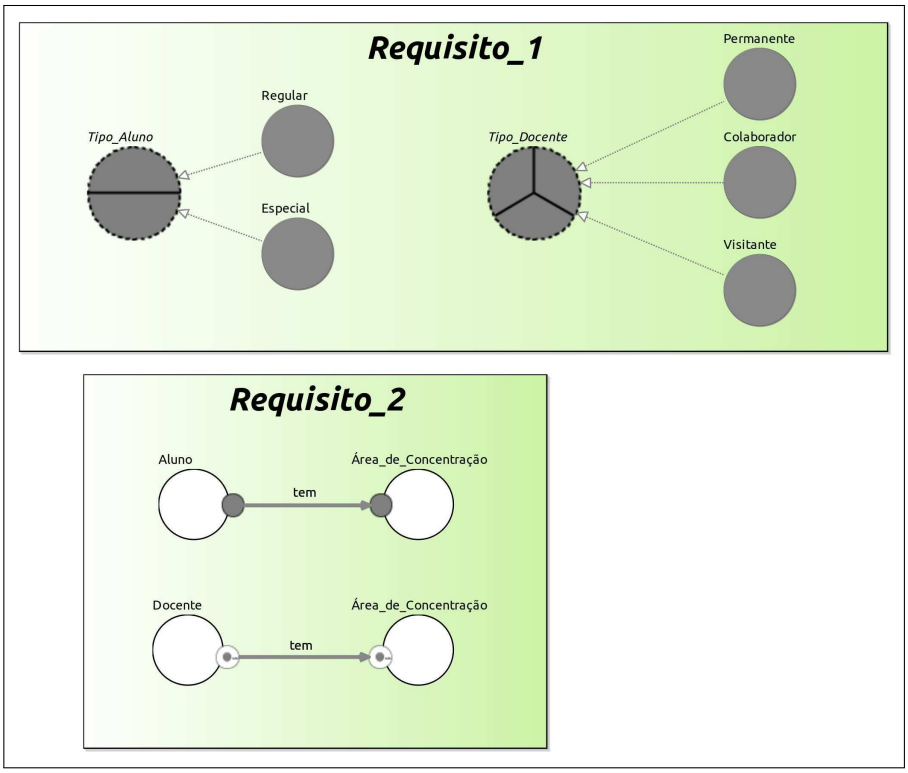

Figura 6.7: Participante 04 – Modelo Conjunto 2

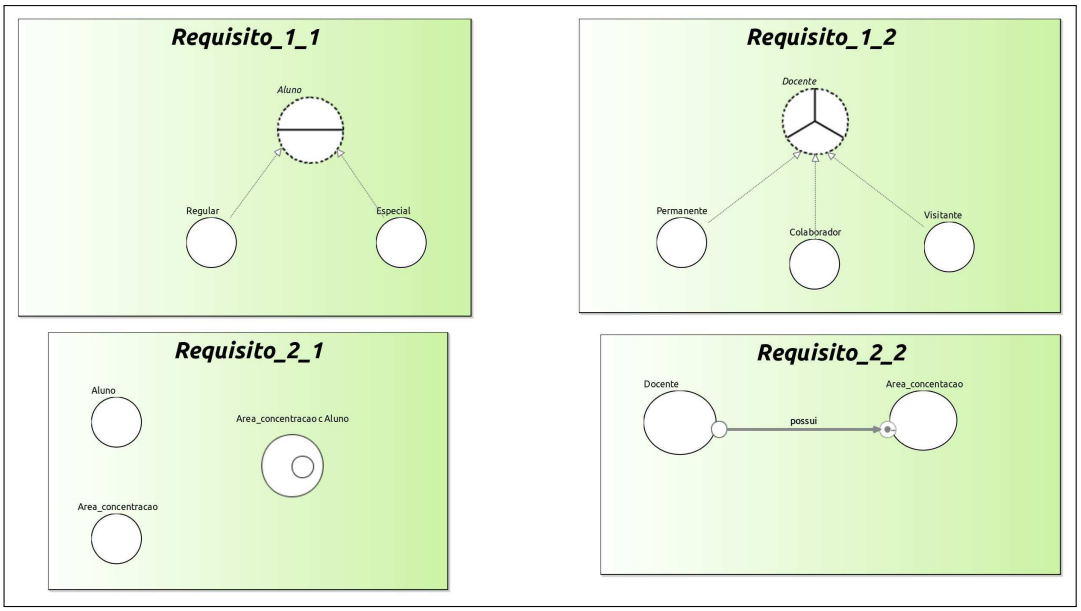

Figura 6.8: Participante 08 – Modelo Conjunto 2

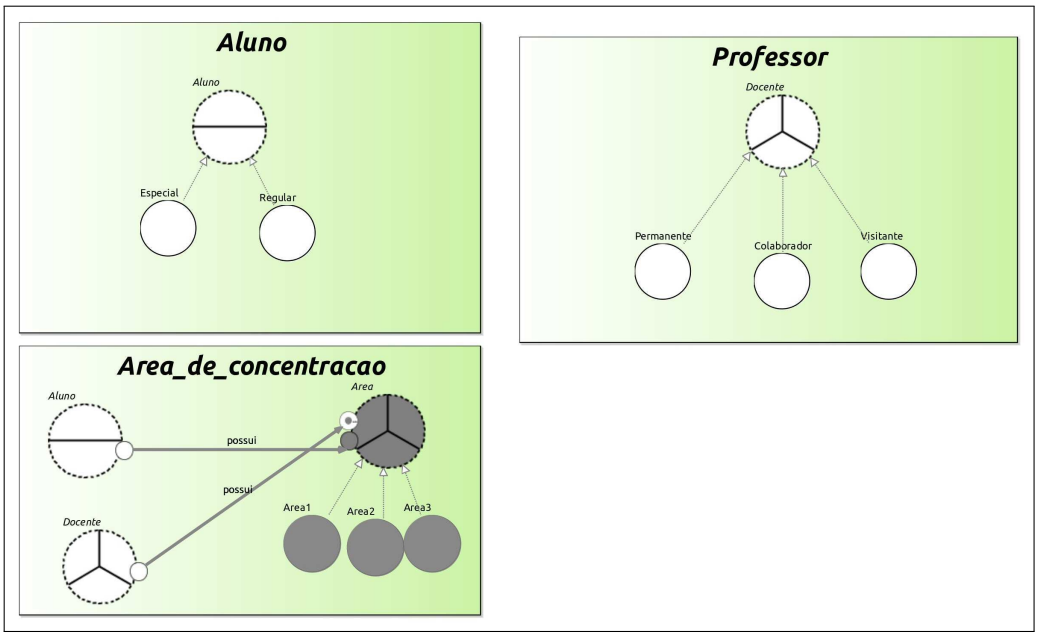

Figura 6.9: Participante 10 – Modelo Conjunto 2

6.2.3 Modelo do Conjunto de Requisito 3

Invariantes propostas para o conjunto de Requisitos 3:

- Um Curso pode ser ou Doutorado ou Mestrado.

- Um Aluno só pode cursar um Curso.

No Conjunto de Requisitos 3, esperava-se que o participante utilizasse Entity, Entity Abstract, Relationship e Multiplicity ONE (apenas um) na construção do modelo gráfico das invariantes em GIRL. As invariantes são semelhantes às dos conjuntos de requisitos 1 e 2, sem necessitar de outras estruturas além das já utilizadas.

Os modelos criados pelos participantes para as invariantes do Conjunto de Requisitos 3 estão dispostos nas figuras E.23 à E.33. Nas tabelas F.5 e F.6 estão registrados os passos de cada participante para a construção da sua solução, suas dúvidas e as intervenções do observador.

Neste conjunto de requisitos, os termos utilizados na linguagem natural foram modificados, como naturalmente ocorre na prática da elicitação de requisitos. O que antes fora escrito por meio dos termos "deve ser", para indicar extensão de entidades, e "deve ter" para relacionamentos, no Conjunto de Requisitos 3, foi expresso com os termos "pode ser" e "só pode", respectivamente.

Alguns perceberam a diferença, o que gerou dúvidas no entendimento das invariantes para uns e, para outros, apenas uma expressão de destaque do fato, sem comprometer o entendimento. A Tabela 6.3 descreve as reações desses participantes, e esclarecimentos quando solicitados. Vale salientar que o participante P01 gastou cerca de um minuto relendo o requisito e avaliando para decidir qual estrutura utilizar.

Tabela 6.3: Percepção da Diferença de Termos no Conjunto de Requisito 3

*	Percepções dos Termos do Requisito 3	*
Part.	**Dúvida**	
P01	Expressou a dificuldade para entender o significado de 'deve' (do Requisito 1 e 2) e 'pode' (do Requisito 3). Achava que deveria mudar a representação dos requisitos anteriores. O observador não interviu. Estava para mudar a representação do requisito 1 em Aluno, mas desistiu. Quando iniciou a representação da segunda invariante mencionou: "Só pode cursar um curso. Diferentes formatos de requisitos.". Em seguida questionou: "Usar relacionamento e multiplicidade?".	
P03	"Dúvida para entender o que significa 'deve' (do Requisito 1 e 2) ou 'pode' (do Requisito 3)."	
P07	"A diferença do outro, do Aluno... O Aluno 'deve ser' e Curso 'pode ser' Doutorado e Mestrado." Viu no tutorial que tinha um 'pode ser' para Contas que estendem uma abstrata. Então concluiu que o requisito se encaixaria no abstract como Aluno e Docente.	
P08	"Curso pode ser Doutorado. Mas, tem que ser um dos dois né? Não pode ser nada."	
P09	"Palavras diferentes aqui: Aluno 'deve ser', Docente 'deve ser' e Curso 'pode ser'! Não tem diferença do Deve e do Pode."	

Análise das Representações da Primeira Invariante

Quatro participantes, P03, P04, P06 e P09, optaram pela representação da primeira invariante utilizando o conceito semelhante à *enumeration* na programação. Tal como fizeram nas invariantes dos requisitos anteriores. Por outro lado, o P02 utilizou entidades singletons pela primeira vez, mas, com o intuito de representar seu entendimento de que "só pode ser um" (Mestrado ou Doutorado) (vide Figura E.24). Possivelmente por considerar que, na invariante "Um Aluno só pode cursar **um** Curso", o termo **um** corresponderia à existência de apenas um curso de Mestrado e um de Doutorado. Um exemplo de ambiguidade na representação com linguagem natural.

Análise da Segunda Invariante do Conjunto de Requisitos

Na segunda invariante, todos os participantes representaram uma relação entre Aluno e Curso, sendo que, dez deles utilizaram diretamente as entidades Aluno e Curso com a multiplicidade ONE em Curso, conforme o esperado pela semelhança com as invariantes dos requisitos anteriores (vide Figura 6.12). Já os participantes P06 e P07 aplicaram quantificadores em substituição à multiplicidade ONE da relação. O participante P03, levantou a possibilidade de utilizar quantificadores, mas manteve a decisão inicial de usar multiplicidade.

O participante P06 justificou sua escolha por quantificadores (vide Figura 6.11) com a seguinte afirmação: "Vou modelar diferente! Com quantificadores!". Isso por considerar a expressão com quantificadores similar à expressão com multiplicidade, pois havia feito o seguinte questionamento na exibição do vídeo do módulo 5: "Qual a diferença de usar isso (quantificadores) e a multiplicidade?".

Por outro lado, o participante P07 transpareceu o entendimento de que a representação só poderia ser feita por meio de quantificadores. A razão disso pode estar na diferença entre os termos "só pode cursar" e "deve ter", com base nas seguintes afirmações do participante: "Só pode cursar 1 curso... Agora faz sentido a quantificação. Quando isola ... agora vendo objeto maior (diagrama), várias variantes, então quantificador faz sentido."

Analisando as duas representações com quantificadores para a segunda invariante **"Um Aluno só pode cursar um Curso"**, identificamos a aplicação de sentidos distintos:

- **Para P06**: ALL(Aluno) "cursando" ONE(Curso) - uma representação adequada para o conceito de que Alunos (ALL) devem estar associados a um Curso (ONE). Ao executar a verificação, as instâncias geradas produziam o resultado esperado, cada Aluno associado a apenas um curso.

- **Para P07**: ONE(Aluno) "cursar" ONE(Curso) - representação parecida com uma tradução, palavra por palavra, da expressão em português. Ao submetê-la à verificação, surgem instâncias em que um Aluno aparece com mais de um Curso, violando a invariante. Esse entendimento distinto pode se explicar na diferença no nível de conhecimento em Lógica dos dois participantes: P06 se caracterizou com conhecimentos avançados e P07 com conhecimentos iniciais. Podemos estar diante de mais um exem-

plo de ambiguidade oriunda da linguagem natural, aliada à dificuldade de formalizar uma invariante com base nos conceitos de lógica. Ou diante da baixa intuitividade do formalismo lógico em relação a quantificadores.

Outras Análises

O participante P09 pensou na possibilidade de usar Set Operation já na primeira invariante, achando que não estaria fazendo a coisa certa, pois não se via utilizando muitas estruturas da linguagem. Esse temor foi expresso em: "Tô conseguindo só com isso? Minha modelagem tá simplória!". Mesmo com todos os cuidados de explicar que não haveria atribuição de certo ou errado na solução elaborada, e que o maior interesse da pesquisa era conhecer o caminho do raciocínio do participante na busca de soluções, ainda assim, o participante se considerou 'sendo avaliado'. Duas ameaças à validade de construto não foram plenamente mitigadas: Interação do Teste e Tratamento e Apreensão na Avaliação [40].

Algo semelhante se deu com o participante P06, mas não caracterizou uma ameaça à validade, porque não causou mudanças ou revisões excessivas. E, além disso, suas verbalizações não denotavam preocupação, e sim, um tom de brincadeira e descontração: "Quero ver se tem aluno que tá cursando nada!", "Ok, acho que tá certo! Quero submeter... se tá Sete, tá bom!". Ou seja, após buscar evidências de que seu modelo estava consistente na verificação, deu-se por satisfeito. E, em seguida, ao realizar o *upload* do arquivo do seu modelo, fez uma analogia com a submissão de atividades avaliativas em cursos que havia participado, desejando, pelo menos, a nota mínima sete na avaliação do quanto seu modelo estaria certo ou errado.

Cardinalidade *versus* multiplicidade também foi uma dúvida do participante P04 na primeira invariante. Mas, quando tentou representar 'um Curso', expressando a cardinalidade da entidade Curso, percebeu: "Cardinalidade igual a um... acho que fica redundante". Ou seja, dizer que a cardinalidade na relação 'cursar' deveria ser igual a um seria uma redundância, pois havia registrado a multiplicidade ONE na entidade Curso. A dedução foi válida porque, a impossibilidade de aplicar cardinalidade a relações levou a considerar que o que já estava modelado era suficiente.

Sete participantes realizaram verificação (P02, P03, P04, P05, P06, P07 e P11). A verificação do modelo do participante P04 trouxe à tona um problema operacional no Analyser

(analisador que implementou a comunicação com a API do Alloy). Apesar disso, o erro disparou uma revisão visual em que o participante evidenciou que o modelo estava incompleto. Faltavam entidades e relações para ligar Aluno e Docente aos respectivos tipos. Também sinalizou a necessidade de trocar os nomes das relações do Conjunto de Requisitos 2, onde duas relações com mesmo nome apontavam para uma mesma entidade. Apesar da descoberta e correção dessa última falha que gerava inconsistências, a correção do problema só ficou disponível na verificação dos próximos participantes.

Ao ser questionado se gostaria de verificar, o participante P09 expressou: "Não quero rodar... fazer como faço ... eu sempre faço tudo, depois é que testo em programação". Um estilo pessoal de teste em codificação, extrapolado para a modelagem de requisitos.

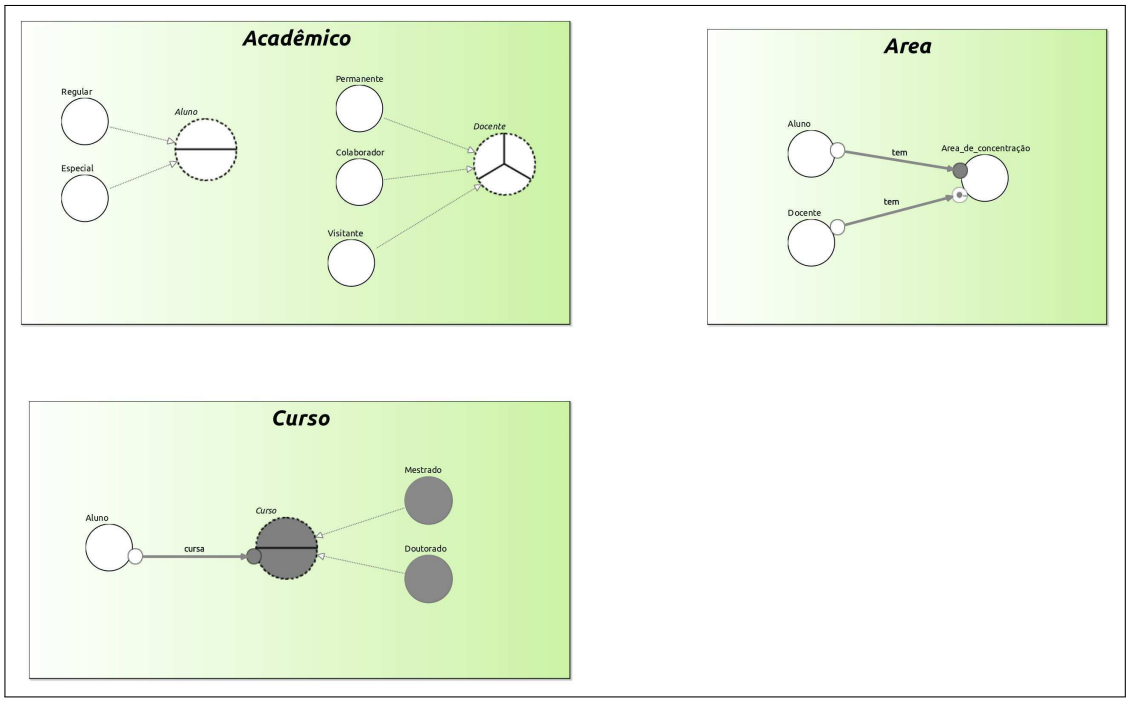

Figura 6.10: Participante 02 – Modelo Conjunto 3

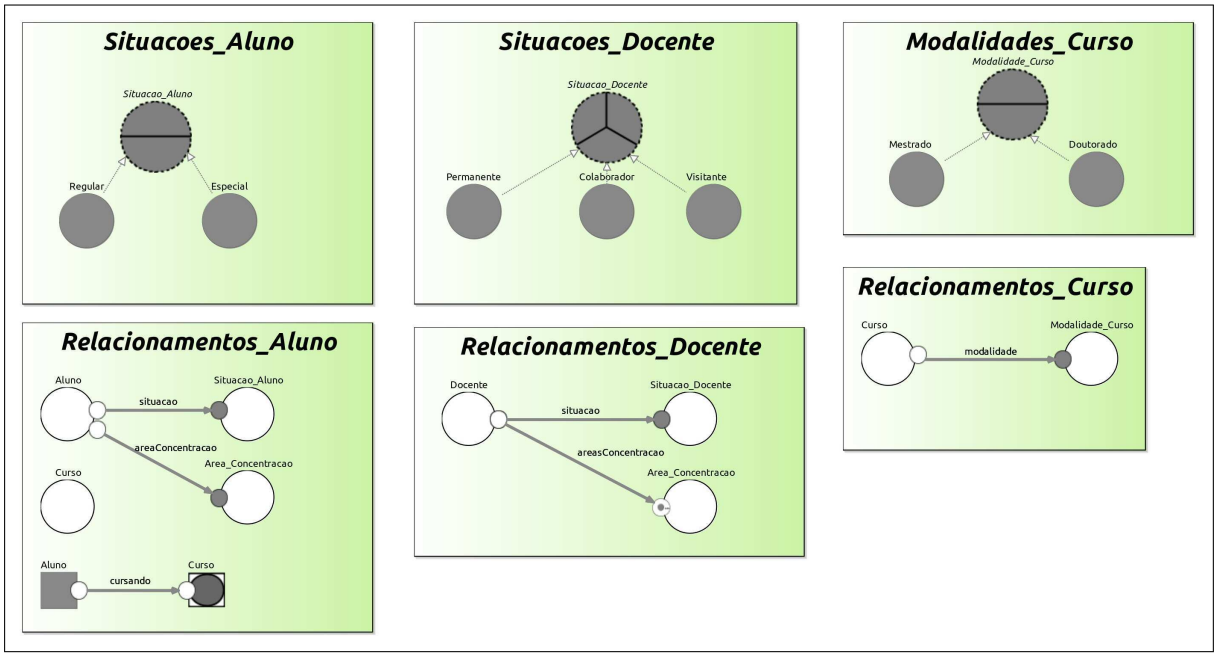

Figura 6.11: Participante 06 – Modelo Conjunto 3

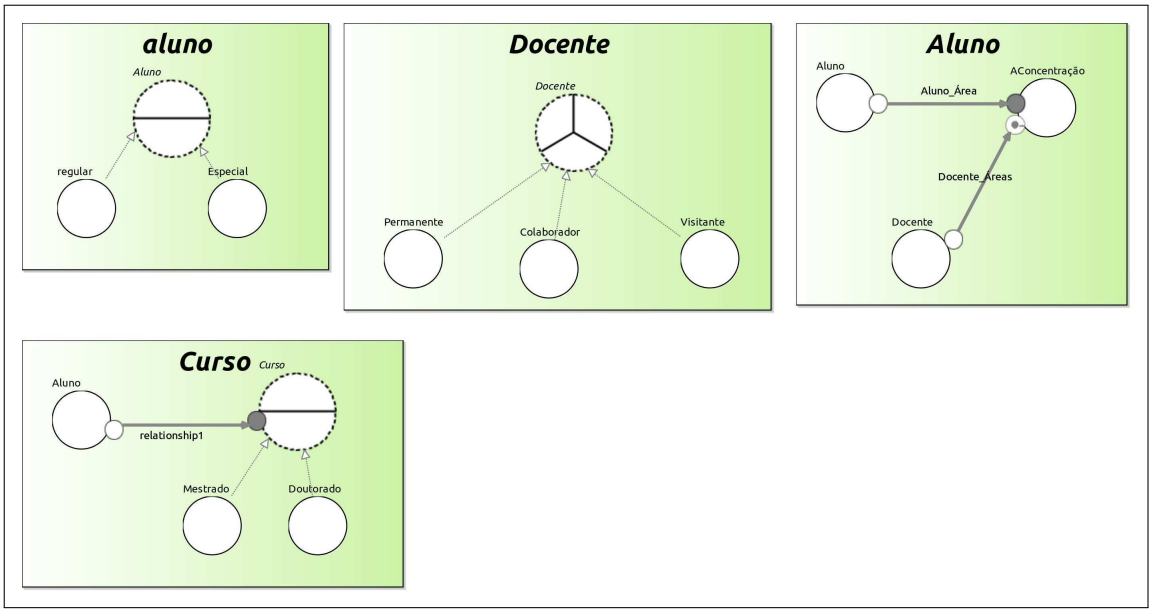

Figura 6.12: Participante 11 – Modelo Conjunto 3

6.2.4 Modelo do Conjunto de Requisito 4

Invariantes propostas para o Conjunto de Requisitos 4:

- Um Aluno pode ter até dois Docentes como orientadores.

- Se um Aluno cursa Doutorado, então ele deve ter um ou mais Docentes orientadores.

Para a representação do Conjunto de Requisitos 4, esperava-se que o participante utilizasse Entity, Quantification, Relationship, Multiplicity com expressão relacional (" <=2 ") e Implication na construção do modelo gráfico das invariantes em GIRL. A representação da expressão relacional na multiplicidade poderia gerar dificuldades por ser um conceito ainda não utilizado e não ter sido colocado nos recursos visuais de auxílio durante as sessões dos participantes (o Quadro e a Cola).

O grau de complexidade requerida para a segunda invariante aumentou, se comparado ao das invariantes modeladas até então. Alguns participantes detectaram a complexidade logo na primeira leitura como descrito na Tabela 6.4. O fato produziu maior frequência de consultas ao Tutorial, Quadro e Cola.

Tabela 6.4: Percepção de Dificuldades com a Implicação

*	Percepção de Dificuldades com a Implicação	*
Part.	**Percepção**	
P02:	"Peraí..."	
P03:	"Se o Aluno cursa Doutorado... aquele leriado de implicação!"	
P04:	"Bagulho da implicação! Posso colar do tutorial?"	
P06:	"E... peraí." Busca implicação na Cola. "Vou botar: Se o ... curso... peraí!" Voltou para ler o requisito.	
P07:	"Agora o negócio começa a pesar..."	
P08:	"Aqui vai ser mais complicado... " disse rindo.	
P10:	Inspirou profundamente indicando que estava diante de algo mais complexo e disse: "Aqui é implicação!"	
P11:	"Se Aluno cursa Doutorado... Entendi... Se isso então aquilo... Uma implicação!"	

Os modelos que os onze participantes criaram para as invariantes do Conjunto de Requisitos 4 estão dispostos nas figuras E.34 à E.44. Nas tabelas F.7 e F.8 estão registrados os passos de cada participante para a construção da sua solução, suas dúvidas e as intervenções do observador.

A seguinte fala do participante P02: "Parece Teoria dos Conjuntos: invariante, o Universo e as bolinhas, os conjuntos", expressa um achado interessante em relação à semelhança do conceito de conjuntos e entidades. No entanto, apesar do gráfico do conjunto Universo

assemelhar-se a uma invariante, o fato é que são todas as invariantes encerradas no arquivo .girl que representam o Universo do modelo. Não existe o conceito de importação de outro modelo em GIRL.

Análise das Representações da Primeira Invariante

Na primeira invariante, sete participantes (P01, P02, P05, P06, P07, P08 e P09) reconheceram a aplicação da expressão relacional na multiplicidade da entidade Docente (ou Orientador). Valeram-se de consultas do material ou de buscas na própria memória. P05, pensou alto mencionando o caminho da cardinalidade, mas, enquanto raciocinava, lembrou: "Vi um que a (era a) relação que colocava na bolinha?", querendo dizer que havia como colocar a expressão relacional (relação) na multiplicidade (bolinha). Para alguns, apenas lembrei a forma de digitar a expressão na multiplicidade. Com P07 também se deu algo parecido: inicialmente pensou, "Ver essa cardinalidade aqui...", leu o tutorial e afirmou, "Mas acho que era assim mesmo... Relacionamento...", leu sobre cardinalidade no tutorial impresso e definiu, "Mas acho que a função que você explicou lá... ", referenciando a explicação de multiplicidade com expressão relacional.

Em contrapartida, para os participantes P03 e P10 foi necessário a intervenção do observador lembrando da expressão relacional na multiplicidade. Ambos estavam buscando uma representação com cardinalidade sem cogitar a multiplicidade.

O participante P04 iniciou criando uma cardinalidade para Docente, um inteiro dois (2) e uma operação relacional menor ou igual (LESS_EQUALS) entre os dois primeiros. Não satisfeito com a representação, pensou alto: "Número de elementos...", depois leu o tutorial e percebeu: "Posso dizer na relação diretamente", o que seria diretamente na multiplicidade.

No momento da criação da operação LESS_EQUALS, um *bug* foi identificado no editor gráfico que representou a operação com "$>=$" (sinal de maior ou igual) quando deveria ser "$<=$". A falha foi resolvida de imediato. Tal incidente foi rapidamente contornado e o participante continuou suas atividades.

No caso do participante P09 ocorreu algo peculiar. Após usar a expressão relacional na multiplicidade, criou uma cardinalidade e tentou copiar a relação (a seta) para dentro da cardinalidade. Seria um caminho mais intuitivo, porém, apesar de Alloy aceitar a cardinalidade da relação, em GIRL não é possível. E, como o editor não permitiu a cópia da relação,

ele copiou Docente para a cardinalidade. Analisou o que fez e constatou que "Na relação já é suficiente", significando que a multiplicidade da relação já expressava o que ele estava querendo dizer com a cardinalidade.

Para o participante P11 pareceu óbvio criar a operação relacional LESS_EQUALS dentro da multiplicidade. Na prática, é isso o que acontece internamente após a digitação da expressão relacional dentro da multiplicidade, mas o editor só aceita os valores digitados. P11 apontou para um caminho mais intuitivo.

Ainda com o participante P11 aconteceu outra particularidade. Ao iniciar a representação da primeira invariante, enquanto dizia "Agora, quem são minhas entidades? Aluno mais 2 entidades, 2 entidades do requisito", criou duas entidades Orientador1 e Orientador2 para utilizar na relação de Aluno com Docente Orientador. Uma associação direta com a expressão "até dois Docentes como orientadores". Com isso, ficou sem saber como continuar a representação. O observador precisou intervir perguntando: "A relação de Aluno é com as entidades Orientador1 e Orientador2 ou com uma entidade maior?". Ao que respondeu "Com Docente".

Análise das Representações da Segunda Invariante

Nessa invariante a solução mais plausível seria utilizar uma implicação com relacionamentos entre quantificadores. O texto da invariante foi construído de tal forma que mostrasse com clareza a necessidade de uma implicação - "Se um Aluno... então...", para que o maior esforço de raciocínio recaísse na busca da representação da premissa e da conclusão da implicação. Isso ficou evidente quando o participante P01 criou logo a implicação e, ao ser questionado do porquê da sua escolha, respondeu: "Só porque tem um 'Se' e 'Então'."

O fato dessa invariante ser semelhante à utilizada no módulo que apresentou a implicação na ambientação, colaborou para que os participantes decidissem, por analogia, pelo uso de relacionamentos com quantificadores nos elementos da implicação. Com isso, ao consultar o Tutorial, o Quadro, ou a Cola, todos teriam igual possibilidade de fazer uma a escolha por analogia.

Algo fugiu do planejado com o participante P02 quando iniciou a representação da segunda invariante. Por uma falha no *setup* das primeiras sessões, o arquivo digital contendo as invariantes dos conjuntos de requisitos ficou desatualizado e continha a seguinte expres-

são para a segunda invariante: "Alunos do curso Doutorado devem ter um ou mais Docentes Orientadores" no lugar de "Se um Aluno cursa Doutorado, então ele deve ter um ou mais Docentes orientadores". P02 optou por ler as invariantes no documento digital e assim, iniciou trabalhando com essa expressão desatualizada que demandava mais esforço para se descobrir que se tratava de uma implicação. Por sua pouca prática na representação em GIRL (como os demais), ficou bem mais difícil para P02 perceber que a invariante tratava de uma implicação. Consequentemente, fez mais tentativas que os demais participantes, produzindo um aumento significativo do tempo de utilização para esse conjunto de requisitos. O problema não chegou a afetar os demais participantes.

Todos os participantes, exceto P03, apresentaram alguma dificuldade na construção dessa invariante como mostra a Tabela 6.5. A dificuldade mais significativa foi no uso de quantificadores. A restrição de só utilizar quantificadores ALL na premissa gerou dúvidas e críticas. A impressão é que a representação da implicação não ofereceu conceito gráfico intuitivo, como também sua construção. A cópia das relações na implicação também gerou dúvidas. Alguns caminhos inéditos de cópia de elementos foram experimentados pelos participantes P10 e P11 sinalizando a baixa intuitividade na construção dos elementos.

Tabela 6.5: Dificuldades na Implicação

* Dificuldades na Implicação *		
Dificuldade	**Part.**	**Descrição**
Uso de quantificadores em premissa e implicação	**P01**	Ia colocar relação entre as entidades. "Ah, tem quantificadores!"
	P02	Tentou colocar quantificadores direto na implicação. Dei dica sobre relationships permitidas na implicação e sobre como montar relationship com quantificadores.
	P05	Estava indo pelo caminho de uma relação sem quantificadores. Relembrei que relações para implicação a premissa tem que ter quantificadores. Consultou o tutorial
	P07	'A relação de quantificação tem multiplicidade?" Revendo o modelo do Módulo 6 - arquivo jpg via arquivo na pasta do Eclipse... Relendo requisito. Vendo implicação no Quadro. "Todo aqui fica a impressão que é todo Vencido... Fica ilógico." Todo significando quantificador ALL.
	P09	"Quero colocar uma implicação." Intervi lembrando que na implicação com relação, a premissa tem que ter quantificadores ALL.
	P11	Intervi informando que em uma implicação com relação deve ser quantificação na premissa.
Como copiar elementos	**P01**	Tentando copiar quantificadores para a implicação sem conseguir. Perguntou o que fazer. Expliquei que só podia copiar a relação.
	P02	Tentou colocar quantificadores direto na implicação.
	P08	Copiou e colou as duas relationships de uma vez na implicação.
	P09	"Se colocar a premissa errada, como fazer? Se eu errar a sequência, vai gerar resultados inconsistentes, né?"
	P10	"Copia como?" Querendo saber como copiar as relationships na implicação. Antes tinha tentado copiar uma entidade de outra invariante na quantificação.

Tabela 6.5: (continuação)

*	Dificuldades na Implicação	*
Dificuldade	**Part.**	**Descrição**
	P11	Queria copiar o nome da relação para a quantificação.
Sobrecarga de relações com o conceito de *enumeration*	**P04**	"Vou ter que fazer os dois já que botei lá tipos!" Percebendo que teria que colocar duas relações até chegar em Doutorado já que usou a a relação tipo_curso ligando Curso a Tipo_curso.
	P06	"Como expresso algo-ponto-algo?" Referenciando as duas relações aluno.cursando.modalidade para chegar na 'modalidade' Doutorado que o aluno está 'cursando'. E solicitou "Me dá dica aí!". Perguntei qual a relação mais preponderante, mais forte na implicação. Essa será a premissa e as demais conclusão. Respondeu "O Doutorado".
Usar entidade estendida Doutorado no lugar da super-entidade Curso	**P04**	Indiquei que Doutorado pode assumir o lugar de Tipo_Curso na relação Curso 'tem' Tipo_Curso
	P06	"Posso botar Doutorado direto?"
	P07	"Ele (a ferramenta) consegue entender Bloqueado no lugar de Situação_Cartão ?" Sobre a possibilidade de usar a entidade estendida Bloqueado no lugar da entidade pai Situação_Cartão na relação.
	P10	"Curso... ou copio a especialização que a gente sabe que é Doutorado?"Para decidir que entidade copiar, Curso ou Doutorado, para usar na relação.
Usar mesmo nome em relações iguais	**P05**	Indiquei que Doutorado pode assumir o lugar de Tipo_Curso na relação Curso 'tem' Tipo_Curso
	P07	Com dúvida sobre o nome do relacionamento. Achou que tinha que ser o mesmo nome da invariante 6.
Criar relações fora para colocar na implicação depois	**P04**	"Se o aluno cursa doutorado... Ah é por fora primeiro!" Sobre criar a relação antes para depois colocar na implicação.
	P07	"Faz tudo fora primeiro, depois bota dentro?" Sobre criar as relações fora da implicação primeiro, depois copiar para a implicação."
	P09	"Posso botar, criar premissa fora e colar, depois fazer a conclusão e colar?
Como diminuir a quantidade de instâncias	**P05**	"'Pode diminuir a quantidade?" Sobre a quantidade de instâncias visualizadas. "Posso só testar a implicação?" Prevendo que ia ter algumas instâncias não significativas para o objetivo da verificação - implicação - o que aumentaria a quantidade de objetos visualizados.

As soluções elaboradas se agruparam em três decisões distintas para a representação do **um ou mais** no trecho "...ele deve ter **um ou mais** Docentes orientadores". A Tabela 6.6 exibe as decisões e distribuição dos participantes em cada uma. O uso da multiplicidade SOME (um ou mais) em Docente (quantificado), como na figura E.39 na relação de orientação foi o mais bem cotado, dando a crer o *design* mais intuitivo nas escolhas dos participantes. Apesar disso, o quantificador SOME (um ou mais), na Figura E.37, é a escolha mais simples (disse tudo com poucas palavras - símbolos gráficos), sendo mais alinhada aos conceitos de lógica. Talvez por isso não tenha sido tão intuitiva. A decisão de usar uma expressão relacional na multiplicidade (vide Figura E.38) pode ter sido motivada pelo exemplo do vídeo na ambientação.

Tabela 6.6: Decisões de Representação na Implicação

* Decisões de Representação na Implicação *											
Decisão	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
Utilizar o **quantificador SOME** (um ou mais) para o Docente na relação de orientador sem multiplicidade	X						X				
Utilizar o quantificador ALL para o Docente na relação de orientador com a **expressão relacional " >=1 " na multiplicidade**					X			X			X
Utilizar o quantificador ALL para o Docente na relação de orientador com a **multiplicidade SOME** (um ou mais)		X	X	X		X			X	X	

Outras Análises

Nessa última etapa da utilização da linguagem GIRL, apenas o participante P01 não realizou verificação por esquecimento, mas durante a comparação entre os modelos ele retomou e verificou. Quase todos puderam verificar, com exceção dos participantes P02 e P04 que não conseguiram realizar a verificação a contento. Ainda assim o participante P02 demonstrou aceitação positiva com as verificações que conseguiu realizar.

Na verificação do participante P10, uma inconsistência associada às multiplicidades da esquerda das relações com quantificadores gerada por uma falha em GIRL impediu de seguir adiante. Foram eliminadas os nomes de relações iguais com relações distintas apontando para uma mesma entidade. Também retiramos as multiplicidades da esquerda que não constavam nos requisitos. Com isso, a verificação pode prosseguir. Nesses tipos de intervenções, o tempo gasto foi considerado mínimo e não invalidaria o tempo de utilização para o conjunto de requisito como um todo. A verificação do participante P04 não pode ser conduzida a contento devido a inconsistências nos nomes das relações e das multiplicidades na esquerda das relações. Como o tempo para solucionar invalidaria o tempo de utilização, optou-se por não prosseguir com a verificação. Por esses motivos, para P02 e P04 os tempos de utilização do Conjunto de Requisitos 4 não foram considerados nas análises descritivas dos tempos de utilização.

Uma limitação do questionário online quanto ao tempo de inatividade, fez com que a sessão expirasse na submissão desse último conjunto de requisitos na sessão dos participan-

tes P02 e P07. Os dados, arquivos e tempos de permanência nas páginas ficaram registrados até o Conjunto de Requisitos 3. A solução dada foi abrir novo questionário com mesmo participante, informar os graus de entendimento anteriores como 1 e carregar o último arquivo com o modelo finalizado. Os dados do participante foram resgatados e, para o tempo de permanência no Conjunto de Requisitos 4, foi considerada a hora de início da nova entrada no questionário e o tempo transcorrido no primeiro, somando o tempo de percorrer as páginas iniciais e de permanência na segunda passada pela última página da utilização. O participante P02 ainda gastou tempo finalizando sua modelagem. Já P07 havia concluído a modelagem no momento da queda da página.

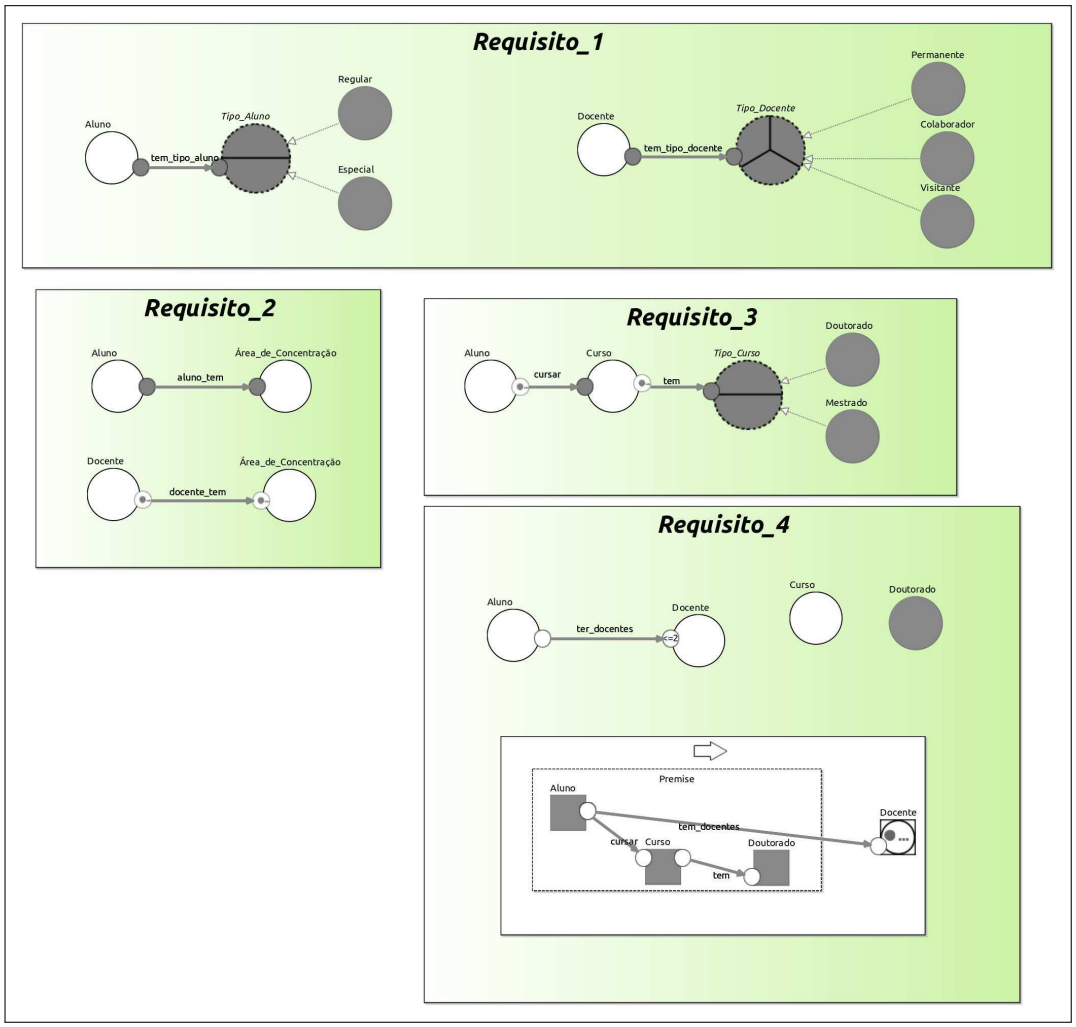

Figura 6.13: Participante 04 – Modelo Conjunto 4

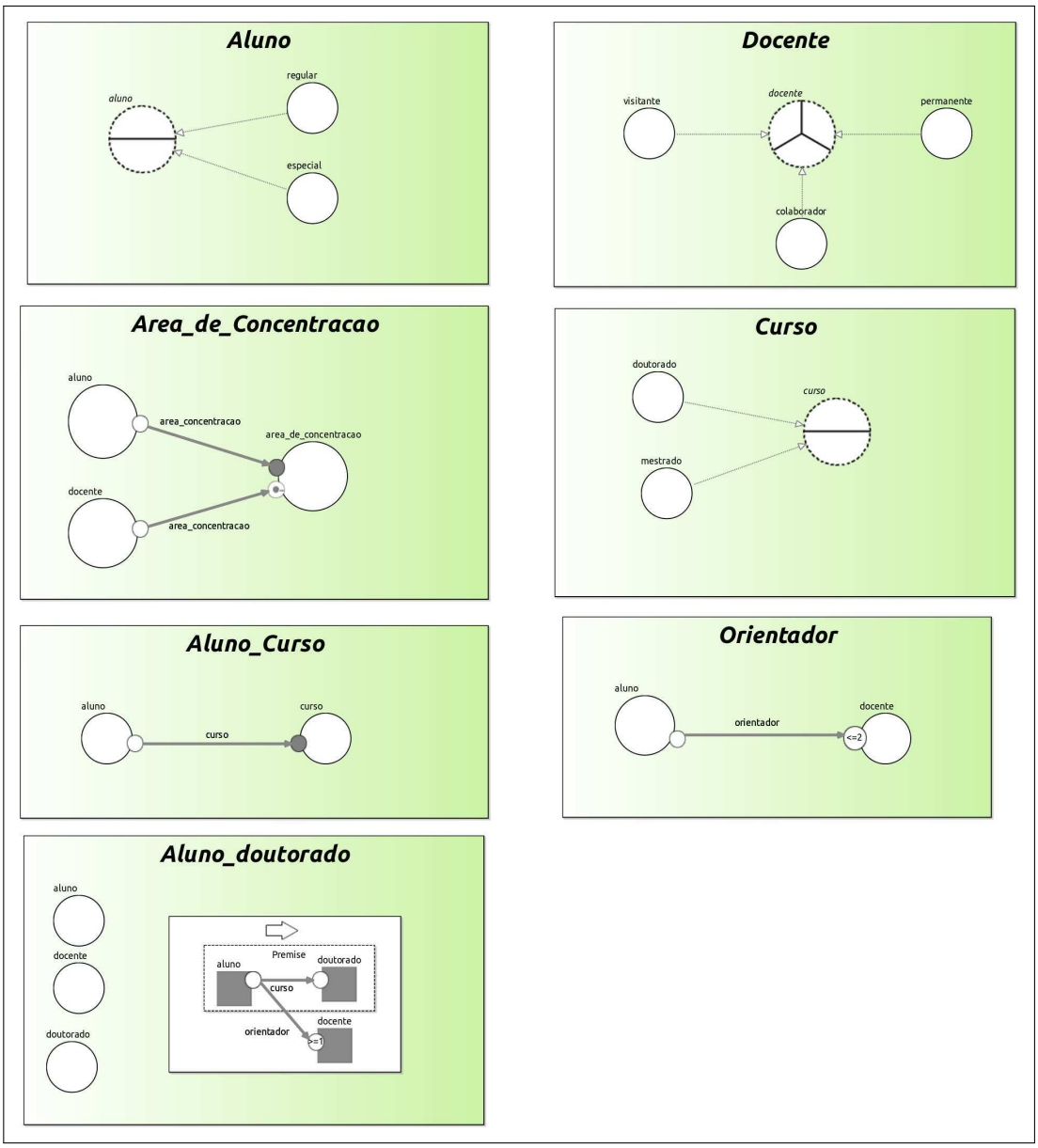

Figura 6.14: Participante 05 – Modelo Conjunto 4

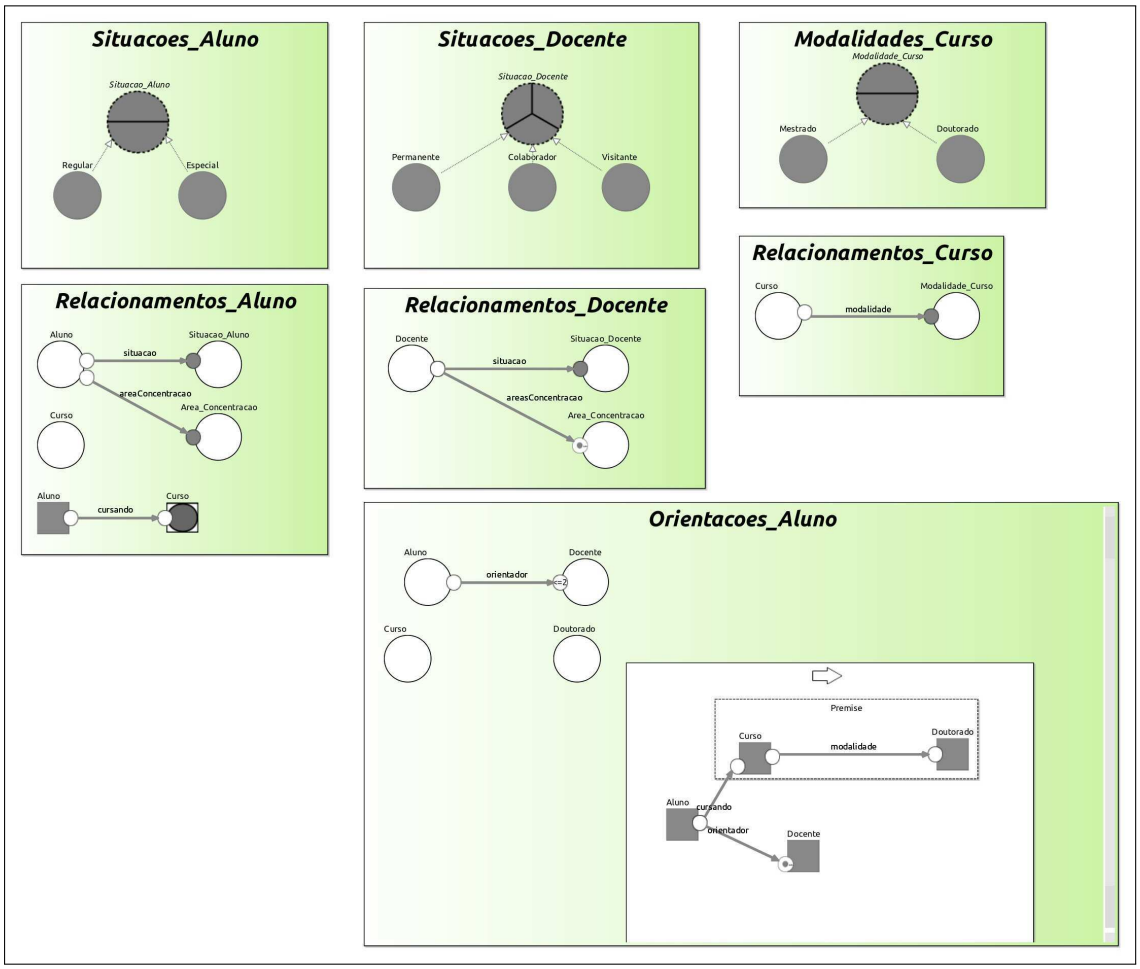

Figura 6.15: Participante 06 – Modelo Conjunto 4

6.2.5 Comparação entre Modelos

Apesar da diversidade de expressões nos modelos elaborados pelos participantes, as modelagens, em sua maioria, representaram as invariantes corretamente, algumas com um "texto visual" mais extenso, com o uso da figura das *enumerations*, outras mais reduzidas. Entretanto, foram observados três pontos de representação equivocadas. No primeiro, P04 inseriu multiplicidades na esquerda da relação modelando algo que não estava explicitado no texto dos requisitos. A falha persistiu porque o participante não pode realizar verificações adequadas ao longo da utilização. No momento da comparação entre os modelos, mencionou que sempre confunde o lado das multiplicidades como acontecia com diagramas em UML.

O mesmo fato também aconteceu com o participante P10, que adicionou multiplicidades na esquerda das relações das invariantes dos requisitos 2 e 3 enquanto modelava as invariantes do Conjunto de Requisitos 4. Ao ser perguntado se o motivo de colocar as multiplicidades na esquerda das relações estava nos requisitos propostos, respondeu não. No vídeo do módulo 3 foi apresentada a verificação com mudança posterior da multiplicidade da esquerda da relação. Essa pode ter sido a causa da adição dessas multiplicidades pelo participante.

Infelizmente, a decisão de utilizar multiplicidade na esquerda da relação fez com que esses dois participantes se deparassem com uma falha nas transformações de GIRL impossibilitando a verificação dessas invariantes. O lado positivo foi perceber que há uma possibilidade dos usuários tentarem aplicar mais restrições ao modelo, além do que fora especificado, seja por excesso de zelo, ou por uma tendência pessoal.

O terceiro ponto de divergência aconteceu quando o participante P08 optou por representar com containment o que se representaria com relationship. Isso aconteceu na modelagem da primeira invariante do Conjunto de Requisitos 2. A troca do conceito não preocupou, pois o participante utilizou a estrutura esperada na representação da próxima invariante que era muito semelhante à primeira. Configura-se numa confusão de conceitos que foi corrigido imediatamente em seguida. Como o participante não focalizou essas invariantes no momento da verificação, a falha permaneceu no modelo.

Cada participante realizou a comparação entre modelo da sua solução e o modelo da solução fornecida pelo observador e registrou as observações que constam na Tabela 6.7. Com base nos registros da comparação e nas expressões dos participantes, observamos: (1) a menção da dúvida do uso de quantificadores nas relações em implicação; (2) a percepção de

que as multiplicidades interagem entre si no modelo; (3) observações sobre maior clareza, objetividade, simplicidade no modelo fornecido; (4) dificuldade de diferenciar o containment do relationship, pela ambiguidade da linguagem textual (5) preocupação de um participante com a granularidade (excesso de símbolos visuais) da representação (6) ameaça não mitigada vista na constatação do participante P09: *Eu devia ter pensado em: "menos é mais". Quis complicar pra mostrar que entendi.* (vide Tabela G.1, Apêndice G).

Tabela 6.7: Comparações entre o Modelo Proposto e o Elaborado

*	Comparação Modelos *
Part.	**Relate o que você considera relevante quanto a diferença entre a representação que você elaborou e a representação proposta no vídeo e modelo acima.**
P01	No R6 b) Acredito que não entendi bem o uso dos **quantificadores junto com implicações**. E acabei usando o lugar errado pra representar o requisito "ter um ou mais".
P02	diferença do modo de modelar apenas
P03	'Na proposta do vídeo: As invariantes foram organizadas de forma mais clara e objetiva; A situação do curso foi representada de forma mais ampla, enquanto minha proposta restringiu as possibilidades de curso; A forma como as situações de Docente e Aluno foi realizada torna o reuso dos conceitos mais simples;
P04	Considerei um tipo_curso e tipo_docente que não foi considerado na representação. Isto teve consequência na representação da premissa no R4. Com relação à multiplicidade da esquerda, vejo que a forma da representação está correta e a minha está equivocada.
P05	Surgiu uma dúvida sobre a multiplicidade das relações no Requisito 4, pois achei que tinham que ser a mesma multiplicidade. Depois entendi que **pode haver multiplicidade diferente, desde que não quebre as outras**. Também, inicialmente, tive um conceito errado da **multiplicidade** da relação com a entidade.
P06	Pensei inicialmente em não utilizar singletons para modelar a parte de situações, mas acabei utilizando-os. Com isso, ficou mais complexo de modelar o requisito 04.
P07	Na representação do vídeo R3_curso_aluno foi coloca a representação com multiplicidade, talvez seja uma solução mais intuitiva do que a que eu abordei na minha solução. Apesar de ter dito que o relacionamento supria o que era especificado na quantificação, acredito que para entendimento do usuário, fica mais intuitivo se for representado as quantidades utilizando a quantificação, para manter a linha de raciocínio durante a validação do projeto. Na representação R4_Orientador_Aluno_Doutorado, a **multiplicidade seria desnecessária já que existe um quantificador** que já representa os valores solicitados.
P08	Achei complicado diferenciar o **Containment do o Relacionamento** comum.
P09	Os modelos são semelhantes, embora diferentes, mas optei por uma representação mais complexa, utilizando mais recursos de modelagem, que poderiam ser simplificados e gerar o mesmo resultado. Isso quer dizer que a ferramenta e linguagem GIRL oferece possibilidades de modelagem diversas para o mesmo problema, portanto, não mudando o resultado diante dessas possibilidades oferecidas, mesmo aumentando ou não o nível de detalhamento ou complexidade da modelagem.
P10	Creio que em minha representação houve uma diminuição da quantidade de entity para representação dos diversos cenários/requisitos elencados. Creio que isso vá impactar na fase seguinte (i.e. Instancias no alloy) devido a granularidade das representações.
P11	A proposta apresentada com a proposta padrão apresentam bastante semelhança, diferindo apenas na concentração dos requisitos.

6.3 Análise Quantitativa

Na análise quantitativa, buscou-se identificar possíveis efeitos de características dos participantes, do entendimento das estruturas da linguagem (GE) e da quantidade de dúvidas (DA e DU) sobre o tempo de utilização (TU) na representação dos requisitos propostos. Quanto menor o tempo para construir modelos corretos, maior a utilidade da linguagem para gerar requisitos precisos e consistentes.

6.3.1 Efeitos da Caracterização do Participante

Os participantes informaram suas características para: Experiência em Engenharia de Requisitos, Nível de Conhecimento em Lógica e Formação Acadêmica. A Tabela D.1 sumariza todos esses dados (e respectivas codificações) por participante. Os tempos de utilização dos participantes P02 e P04 foram suprimidos dos gráficos relativos ao Conjunto de Requisitos 4, por não serem representativos em virtude de falhas no verificador.

Buscamos as possíveis correlações entre Tempo de Utilização dos participantes em cada conjunto de requisito (1 a 4) com a Experiência em Engenharia de Requisitos, Nível de Conhecimento em Lógica e Formação Acadêmica. Quanto maior o nível das variáveis independentes, esperava-se uma diminuição no Tempo de Utilização.

Observamos que a Experiência em Engenharia de Requisitos e Formação Acadêmica não interferem significativamente na rapidez da representação das invariantes nos conjuntos de requisitos (gráficos nas figuras K.1, K.2 nos apêndices).

No entanto, a característica do Nível de Conhecimento em Lógica parece afetar positivamente na rapidez da representação das invariantes, como demonstra o gráfico na Figura 6.16. Percebemos que nos participantes que informaram possuir nível avançado de conhecimento em lógica (03A), a variação de tempo de utilização em cada conjunto de requisitos foi menor (*boxplot* mais achatado) e em níveis menores que os demais participantes. Essa tendência foi mais nítida no Conjunto de Requisitos 4 em que foram utilizadas estruturas puramente da lógica.

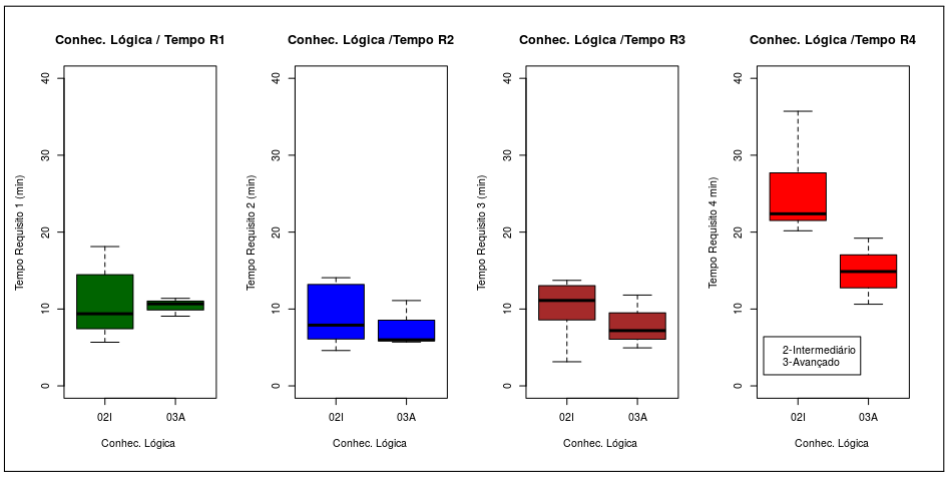

Figura 6.16: Grau de Conhecimento em Lógica / Tempo Utilização

6.3.2 Efeito do Grau de Entendimento

Ao tentar definir uma correlação entre os Graus de Entendimento nas estruturas abordadas e o Tempo de Utilização em cada conjunto de requisitos, tentaríamos negar a hipótese de que, quanto maior o grau de entendimento, menor seria o tempo de utilização. No entanto, a linha da regressão simples evidenciou uma função praticamente constante, demostrando baixa correlação entre as duas variáveis (gráficos dispostos nos apêndices - figuras K.3, K.4, K.5 e K.6).

6.3.3 Efeito da Quantidade de Dúvidas

É possível haver uma correlação entre a quantidade de dúvidas registradas para estruturas da linguagem e o Tempo de Utilização em cada conjunto de requisitos. A hipótese a ser negada seria: Quanto maior a quantidade de dúvidas maior seria o tempo de utilização. Entretanto, os gráficos dos pontos e da linha da regressão linear simples das quantidades de dúvidas na ambientação e utilização, por tempo de utilização dos quatro conjuntos de requisitos, não evidenciam comportamento padrão, tornando-se irrelevantes para o estudo (gráficos dispostos nos apêndices - figuras K.7 e K.8).

A quantidade de dúvidas não parece ser uma variável que afete no tempo de modelagem das invariantes. Quando uma dúvida é sanada, o tempo de modelagem tende a diminuir. No entanto, novas dúvidas podem surgir, a dificuldade de modelagem pode não estar relacionada às dúvidas, mas à dificuldade com a linguagem em si.

6.4 Discussões

Ao cabo da coleta, apresentação e análise dos dados gerados no estudo, podemos então elaborar as respostas das questões de pesquisa estabelecidas. Com as principais constatações que surgiram dessa análise, procedemos uma classificação de acordo com a questão pesquisa associada, para embasar as discussões que alicerçaram os resultados do presente estudo empírico. A Tabela 6.8, registra essas constatações que são discutidas nas próximas seções.

Tabela 6.8: Classificação das Constatações e Problemas na Análise dos Dados

*	Constatações e Problemas	*
Avaliação	**Questão de Pesquisa ou Outros**	**Descrição**
GIRL	GIRL	Avaliação geral positiva da linguagem GIRL.
GIRL	GIRL	Baixo poder estatístico pelo tamanho e escolha da amostra.
Estrutura	QP1	Maior dificuldade em Implication pela sua formatação e manuseio.
Análise Descritiva	QP1	Experiência em Engenharia de Requisitos e Formação Acadêmica não interferem significativamente na rapidez da representação das invariantes nos conjuntos de requisitos.
Análise Descritiva	QP1	Nível de Conhecimento em Lógica parece afetar positivamente na rapidez da representação das invariantes.
Editor	QP1	Dificuldades na construção de implicação.
Estrutura	QP1	Consenso na avaliação positiva de Entity e Relationship.
Estrutura	QP1	Razoável dificuldade em Cardinality e Quantification.
Estrutura	QP1	Baixa dificuldade em Containmet.
Estrutura	QP1	Limitações da escolha dos exemplos para a ambientação.
Requisito 2	QP1	Nomes iguais de Relationships apontando para uma mesma entidade geram inconsistências no verificador. As trocas de nomes eram sugeridas para que a verificação prosseguisse.
Requisito 4	QP1 e QP4	Artefatos de consulta mais utilizados em face da maior complexidade nas invariantes.
Requisito 4	QP1 QP3	Inconsistências na representação de P04 impediu a verificação.
Requisito 1	QP2	Tendência de sujeitos codificadores de aplicar recursos de codificação para a modelagem de conceitos na fase de requisitos.
Comparação dos Modelos	QP2	Todos os participantes representaram corretamente a maioria das invariantes dos quatro conjuntos de requisitos. Três participantes apresentaram representações divergentes. Apenas um deles registra equívoco em uma das oito invariantes. Os outros dois apresentam excesso de restrição, não equívocos.
Requisito 1	QP3	Baixa adesão à verificação nas primeiras invariantes.
Estrutura	QP3	Sugestão de geração de código.
Requisito 2	QP3	Aumento da adesão à verificação (de duas passou para quatro).
Requisito 3	QP3	Aumento da utilização do mecanismo de verificação.
Requisito 3	QP3	Um participante declaradamente prefere a verificação não incremental.
Requisito 3	QP3	Problema operacional do Analyser - Participante P04 não conseguiu verificar.
Requisito 4	QP3	Requisito desatualizado para P02 impediu a verificação.

Tabela 6.8: (continuação)

* Constatações e Problemas *		
Avaliação	**Questão de Pesquisa**	**Descrição**
Verificação	QP3	Avaliação positiva do mecanismo de verificação nos dados registrados pelos participantes.
Verificação	QP3	Avaliação positiva detectada pelas reações positivas dos participantes nos registros do observador.
Verificação	QP3	Limitações do mecanismo em face de inconsistências sintáticas de alguns modelos.
Verificação	QP3	Sugestão de uso das instâncias como casos de uso ou teste.
Comparação dos Modelos	QP4	Constatações de possibilidade de imprimir maior clareza, objetividade, simplicidade nas modelagens.
Comparação dos Modelos	QP4	Preocupação de um participante com o excesso de símbolos visuais numa representação gráfica.
Editor	QP4	Indicativos de efetividade na representação gráfica de entidade, multiplicidade, relações.
Editor	QP4	Indicativos de efetividade na representação das instâncias na verificação.
Requisito 2	QP4	O conceito de Cardinality gera razoável associação ao conceito de Multiplicity.
Requisito 2	QP4	O conceito de Containment levou a uma associação ao conceito de Relationship, com provável confusão com o termo 'ter' da invariante.
Requisito 2	QP4	Conceitos de Multiplicity nas relações foram bem entendidos e aplicados.
Requisito 2	QP4	A representação gráfica, por si, auxiliou a compreensão e a verificação do modelo, por meio da "leitura" dos símbolos visuais.
Requisito 3	QP4	Susceptibilidade da linguagem natural a ambiguidades.
Requisito 3	QP4	Uso de quantificadores no lugar da multiplicidade.
Requisito 4	QP4	Conflito entre Cardinalidade versus Multiplicidade ainda foi observado, mas com menor intensidade. O entendimento do conceito de multiplicidade estava mais sedimentado para a representação do número de elementos na relação ("Posso dizer na relação diretamente.", por P04).
Requisito 4	QP4	Analogia visual da representação em GIRL com a Teoria dos Conjuntos.
Requisito 4	QP4	Tentativa de representar expressão relacional da multiplicidade por meio da estrutura Operação Relacional.
Requisito 4	QP4	O sujeito com pouco ou nenhum conhecimento em Engenharia de Software foi mais propenso a 'inovar' na forma de construir sua representação.
Comparação dos Modelos	Validade	Tendência de aplicar outras restrições além do especificado para experimentar outros cenários. Talvez porque os participantes se encontravam em processo de aprendizagem ou de sedimentação da aprendizagem recém adquirida.
Requisito 3	Validade	Ocorrência de uma das ameaças à validade de construto (Interação do Teste e Tratamento e Apreensão na Avaliação) em um participante.
Requisito 4	Validade	Queda do questionário online por tempo de inatividade excedido.

6.4.1 QP1 - Qual o nível de dificuldade percebida no aprendizado e utilização da linguagem gráfica GIRL?

O nível de dificuldade foi considerado baixo para estruturas mais simples (Entity, Relationship, Invariant) e com forte associatividade com conceitos comuns na Engenharia de Software e a Teoria dos Conjuntos. Em estruturas mais complexas e de maior associação com conceitos da lógica do que com os da engenharia de software (Quantification, Implication, Cardinality e Containment), a dificuldade foi maior. Nesses últimos, a maioria dos participantes construiu o modelo com pouca intervenção do observador. Apesar disso, os participantes registraram um alto grau de utilidade da linguagem, alguns deles sugerindo agregar a geração de código dos modelos confeccionados em GIRL.

Uma grande parcela de dúvidas e pontos negativos na Implication e Quantification, residiam em questões de edição e manipulação dos elementos. Isso conduz à necessidade de revisão e melhorias na representação gráfica do conceito desses elementos.

Os conceitos de Cardinality e Containmet foram confundidos com a Multiplicity no momento da escolha das estruturas. Uma possível razão para esse comportamento está em alguns exemplos utilizados na ambientação, que eram expressões redundantes se comparadas a outras expressões de exemplos anteriores ou posteriores, quais sejam:

- Módulo 1, exemplo de containment - *Banco Central está contido em Banco* - gerou dificuldades de associação com conhecimento prévio do sistema bancário no Brasil. Muitos interpelaram se não seria *Bancos estão contidos em Banco Central.*

- Módulo 2, exemplo de cardinality - *A quantidade de Titulares é maior ou igual a um* - ao restringir a quantidade de titulares de conta por meio da cardinalidade da entidade Titular, e, no próximo módulo, a restrição foi expressa por meio da relação 'titular' entre Conta e Cliente com a multiplicidade no Cliente. Uma limitação no design do estudo na escolha dos exemplos para a ambientação. Uma explicação para esse comportamento poderá ser investigada em uma nova rodada do estudo, com outros sujeitos, com a definição de exemplos mais significativos e que não gerem duplicidade de conceitos.

- Módulo 4, exemplo de Logical Operation - *Uma Conta Corrente está contida em*

Conta E uma Conta Poupança está contida em Conta - essa expressão está implícita no conceito de entidades que estendem uma entidade abstrata. Alguns participantes detectaram essa redundância.

- Módulo 5, exemplo de Quantification - *Toda Conta tem apenas uma Situação* - Essa invariante também poderia ser expressa em termos de multiplicidade em Situação na direita da relação. Dois participantes utilizaram essa variação de expressão na utilização.

6.4.2 QP2 - Qual a corretude dos requisitos representados durante a utilização da linguagem GIRL?

Praticamente todos os 11 participantes representaram corretamente as oito invariantes propostas, com pouca intervenção do observador. Apenas um participante fez uma representação equivocada com Containment para uma invariante, mas, logo em seguida, representou uma invariante semelhante por meio de relationship, o que era o esperado. Outros dois incluíram restrições além das propostas nas invariantes.

Apesar da dificuldade relatada na utilização da implicação, praticamente todos perceberam que era o caso de uso da implicação. Para muitos, foi mais difícil perceber a necessidade do uso de quantificadores nas relações que seriam os elementos da implicação. No exemplo de implicação apresentado na ambientação havia relações com quantificadores. Para esses, não foi rápido o resgate da lembrança desse exemplo. Isso pode levar a duas explicações, ou uma combinação delas. A primeira, a falta de associação da representação gráfica da implicação com o conceito. A segunda, a baixa familiaridade com o conceito de implicação em lógica matemática.

6.4.3 QP3 - Qual a utilidade percebida na verificação de requisitos representados durante a utilização da linguagem GIRL?

A resposta a essa questão foi bem clara. Todos mencionaram satisfação ao ver as instâncias sendo exibidas e podendo verificar se o conceito modelado estava 'acontecendo' como previsto. As expressões de expectativa (*Bom, vou poder rodar pra ver se tirei zero!*) e de se

deparar com uma nova maneira de validar, verificar, executar, debugar, descobrir inconsistências ou simular, pareceu agradável a todos. Foi uma novidade poder testar um software ainda na fase de requisitos (*Gostei dessa ferramenta. Nunca tinha usado uma ferramenta para requisitos.*).

Infelizmente, a expectativa foi frustrada para um participante que não conseguiu prosseguir na verificação por conta das falhas nas transformações entre as linguagens. Uma limitação na ferramenta que comprometeu o resultado de um participante.

6.4.4 QP4 - Qual a efetividade da representação gráfica durante a utilização da linguagem GIRL?

Essa questão está intimamente relacionada à primeira questão sobre as dificuldades no aprendizado e na utilização da linguagem GIRL. Dois aspectos foram avaliados: a efetividade da informação visual (como preconiza Moody [23]) e a facilidade de utilizar o editor gráfico.

Quanto ao editor gráfico em si, a avaliação dos participantes foi positiva para a maioria das estruturas. A exceção ficou na dificuldade de "desenhar" a implicação, o redimensionamento de alguns componentes (a implicação é um caso) e o pouco uso de cores. Os dois primeiros pontos negativos se referem a usabilidade da ferramenta.

De fato, esse é um ponto negativo da linguagem GIRL, prover pouca variação em cores. A cor é uma das oito variáveis visuais fundamentais para o design gráfico, as outras sete são: posição horizontal e vertical, forma, brilho, tamanho, orientação e textura. Moody [23] afirma que, ao variar os valores a essas variáveis, informações são transmitidas. Essas variações geram fortes interpretações perceptivas (pela visão) difíceis de serem derrubadas conscientemente.

Apesar desse ponto explicitamente mencionado para a não efetividade da notação, outras observações elevam a efetividade da notação visual de GIRL, como: (1) associação do "fatiamento" de entidades abstratas (transparência semântica); (2) a cor distinta para entidades *singleton* (expressividade visual); (3) associação do significado dos símbolos gráficos com a teoria dos conjuntos (integração cognitiva); (4) percepção visual de diferença de significados das multiplicidades expressas com identificação de falha (discriminação perceptiva); (5) a satisfação de lidar com o *feedback* gráfico na verificação dos modelos; e (6) a facilidade de

desenhar um modelo à mão, mesmo para os que tem poucas habilidades.

Além das observações positivas de reações ou menções dos participantes, pudemos identificar outros princípios PoN [23] em prática, na linguagem GIRL, em algum grau de intensidade[1], que são: clareza semiótica, codificação dual e economia gráfica.

6.5 Ameaças à Validade do Estudo

Para que o estudo empírico seja considerado válido em seus resultados, é necessário checar as possíveis ameaças à sua validade. Wohlin et al. [40] classifica as possíveis ameaças à validade em quatro grupos: validade de conclusão, validade interna, validade de construto e validade externa. Discutimos mais detalhadamente as possíveis ameaças de cada grupo por tipo de ameaça, avaliando seu potencial causador, seu grau de ameaça e que ações foram estabelecidas para evitá-la.

Além dessas ações, Runeson et al. [27] indica meios de incrementar a validade em estudos de caso, como a triangulação, onde dados são coletados de várias fontes para identificar possíveis contradições. No estudo em questão, é possível avaliar minimamente se o grau de entendimento fornecido na ambientação corresponde ao nível de dúvidas observadas durante a utilização.

6.5.1 Ameaças à Validade de Conclusão

Ameaças à validade de conclusão refere-se a questões que afetam a capacidade de tirar conclusões corretas sobre a relação entre o tratamento e as saídas do experimento [40]. Quanto ao estudo avaliamos as seguintes ameaças, com potencial causador e as ações para evitá-las:

- **Baixo Poder Estatístico**: O tamanho da amostra (11 participantes) pode afetar as conclusões, no entanto, por limitações de tempo e recursos, a amostra não pode ser alargada. Esforços foram direcionados para assegurar a replicabilidade do estudo.

- **Suposição Violada dos Testes Estatísticos**: testes estatísticos não serão aplicados devido ao teor qualitativo e descritivo do estudo.

[1] conforme descrito na Tabela 7.1 da Seção 7.3 do capítulo Trabalhos Relacionados

- ***Fishing* e Taxa de Erro** : Não focalizar o máximo de aspectos relevantes da observação ou coleta de dados pode ser uma ameaça. Com isso em vista, realizamos uma cobertura extensa na análise dos dados.

- **Confiabilidade de Medição**: Sobre o questionário, pode haver falta de clareza nas questões, dimensionamento inadequado das questões, questões prioritárias não respondidas. Para evitar esse efeito, um estudo piloto foi realizado e mantivemos a presença do observador nas sessões de aplicação do estudo.

  Há a possibilidade do instrumento de coleta de dados ficar indisponível ou susceptível a queda por inatividade. Providenciamos a quebra do questionário em unidades menores para distribuir o tempo de ambientação e utilização.

  Quanto ao instrumento de anotações das observações, ele pode ser indevido ou pode ocorrer falha no registro de observações importantes. A solução é empregar codificação reduzida nas anotações e pedir esclarecimentos em situações mal compreendidas. Como o observador também pode ser requisitado para sanar dúvidas ou resolver problemas no manuseio da ferramenta, o método das anotações deve ser bem aplicado. A digitação dos dados anotados também deve ser feita num momento o mais próximo possível da sessão. A decisão de marcar apenas um participante por turno da manhã ou tarde, para evitar fadiga do observador, também mitiga essa ameaça.

- **Irrelevância Randômica na Configuração Experimental**: Ruídos ou interrupções no momento da sessão podem deturpar os resultados. Para evitar isso, reservamos uma sala para ser utilizada apenas nas sessões com os participantes. Mantemos a sala trancada no momento da sessão e o ar-condicionado no mesmo nível de ruído em todas as sessões.

- **Heterogeneidade Randômica dos Sujeitos** : Grupos muito heterogêneo podem afetar os resultados. Então, no convite para a pesquisa, ressaltamos o perfil desejado do participante. Já um grupo homogêneo demais pode afetar a validade externa. Para minimizar esse efeito, convidamos profissionais de vários setores da instituição e um especialista em matemática.

6.5.2 Ameaças à Validade Interna

Ameaças à validade interna são influências que afetam as variáveis independentes sem o conhecimento do pesquisador [40]. Como no nosso estudo não houve grupo de controle, analisamos apenas as ameaças pertinentes a esse tipo de configuração. Quanto ao estudo avaliamos as seguintes ameaças, com potencial causador e as ações para evitá-las:

- **História**: A interação entre os participantes poderia fazer com que alguns tenham mais conhecimento do objeto do estudo, ficando em condições desiguais. Buscamos fugir desse efeito realizando sessões com um indivíduo por vez e solicitando a cada participante que não troque informações com outros participantes, antes que todos tenham sido submetidos à sessão de estudo.

- **Maturação**: Surgimento de tédio e/ou sonolência na exibição dos vídeos é uma possível ameaça. Para minimizar essa ameaça, a sessão foi aplicada uma única vez para cada participante, sendo observado os níveis de cansaço e de atenção de cada um. Disponibilizamos também, água e alimentos para evitar sonolência durante a exibição dos vídeos.

- **Conhecimento do Teste**: Como não há repetição das sessões, não precisamos nos preocupar com esse efeito.

- **Instrumentação**: Dificuldades de entendimento e preenchimento de questionário e vídeos mal elaborados, são possíveis ameaças desse tipo. Para detê-las, avaliamos os instrumentos no estudo piloto com dois sujeitos.

- **Regressão Estatística**: Não há classificação dos sujeitos com base em estudo prévio, portanto o estudo não está passível dessa ameaça.

- **Seleção**: Apesar da participação ser voluntária, os sujeitos são profissionais, e não há recompensas para o melhor desempenho.

- **Mortalidade**: Desistências podem afetar negativamente o estudo. Para isso não ocorrer, estimulamos a participação e flexibilizamos mudanças de horário de acordo com a conveniência do participante.

- **Ambiguidade sobre a Direção da Influência Causal**: As variáveis independentes experiência, grau de conhecimento em lógica e nível de formação dos participantes numa amostra limitada a 11 não representam uma tendência geral. Assim, é melhor avaliar e registrar com cautela as evidências de efeito causal das variáveis que caracterizam o participante.

6.5.3 Ameaças à Validade de Construto

Ameaças à validade de construto são concernentes a generalizações de resultados do experimento para o conceito ou teoria por trás do experimento. Algumas ameças são ligadas ao projeto do experimento, outras à fatores sociais [40]. Quanto ao estudo avaliamos as seguintes ameaças, com potencial causadore as ações para evitá-las:

- **Explicação Pré-operacional Inadequada dos Construtos**: Uma possível ameaça está nas métricas ou tratamentos inadequados para as variáveis: quantidade de dúvidas, tempo de utilização, grau de entendimento, grau de utilidade. Para que isso não ocorra, vamos analisar a quantidade de dúvidas que persistiram, considerando possível falha no artefato ou na resposta às dúvidas. Também registrar qualquer perturbação ou interação com o observador que possa afetar o tempo de utilização. Comparar grau de utilidade registrado com a corretude dos modelos gerados. Considerar o baixo poder de conclusão com base nos dados das variáveis.

- **Viés Mono-operação**: Os sujeitos foram submetidos a quatro conjuntos de requisitos que demandavam a utilização das principais estruturas da linguagem com diferentes níveis de complexidade.

- **Viés Mono-método**: Com critérios para julgamento da acurácia dos modelos elaborados foram bem estabelecidos, sendo possível variação de representação para uma mesma invariante, esta ameaça está tratada.

- **Confusão nos Construtos e Níveis de Construtos**: Como os níveis das variáveis foram definidos com granularidade adequada - escala de 1 a 10 do grau de entendimento e mais de quatro níveis nas variáveis de caracterização do sujeito, consideramos que essa ameaça está afastada.

- **Interação de Diferentes Tratamentos**: Sujeitos participando apenas uma vez da aplicação do estudo descarta essa ameaça.

- **Interação do Teste e Tratamento**: O risco do participante tentar ser bem-sucedido no teste está sendo mitigado com a explicação de que, o objetivo do estudo não é atribuir conceito de certo ou errado, mas conhecer como se deu o raciocínio participante para elaborar a solução.

- **Generalização Restringida por entre os Construtos**: É possível que a utilidade da linguagem fornecida não seja demonstrada na utilização. Para que isso não ocorra, vamos checar se a utilidade foi confirmada pelas declarações durante a utilização.

- **Adivinhação da Hipótese**: Caso o participante queira demonstrar que a linguagem é útil, vamos observar o nível de atenção e interesse do participante durante a sessão.

- **Apreensão na Avaliação**: A possibilidade do participante demorar por excesso de conferências para ter êxito na modelagem, por temer o insucesso, também está sendo tratada com a explicação de que o objetivo do estudo não é atribuir conceito de certo ou errado, mas conhecer como se deu o raciocínio para elaborar a solução, e assim, avaliar a linguagem e não o participante.

- **Expectativas do *Experimenter***: Apesar da amostra ser de conveniência, vamos aplicar o estudo conforme o protocolo com o máximo de impessoalidade. Ao final, vamos enviar o estudo para que os participantes tomem conhecimento dos resultados.

6.5.4 Ameaças à Validade Externa

Ameaças à validade de externa são condições que limitam a capacidade de generalizar os resultados do experimento para a prática industrial [40]. Quanto ao estudo, avaliamos as seguintes ameaças, com potencial causador e as ações para evitá-las:

- **Interação da Seleção e Tratamento**: Os sujeitos da pesquisa foram desenvolvedores com níveis distintos de experiência em requisitos, alguns sem nenhuma. No entanto, a amostra não é significativa para generalizar os resultados, é necessário mais estudos com amostra maior.

- **Interação de Configuração e Tratamento**: O estudo foi conduzido com requisitos reais mas em ambiente simulado. Há certa garantia de representatividade nos tratamentos, apesar da baixa complexidade dos requisitos para que o tempo da sessão fosse exequível.

- **Interação da História e Tratamento**: Um dia ou horário escolhido para a sessão pode ser particularmente significativo para o participante, por causa de um acontecimento próximo relacionado ao objeto do estudo, e que interfere no seu julgamento ou comportamento. Como o estudo ocorreu presencialmente, uma maneira de impedir essa ameaça é sondar acontecimentos relacionados ao estudo e, caso se confirme, sinalizar a ocorrência para o participante.

Capítulo 7

Trabalhos Relacionados

Este estudo abrange a propositura e a avaliação da linguagem GIRL, que fornece mecanismos para modelagem gráfica formal e para verificação automática de requisitos invariantes. Nesse contexto, é importante situar e comparar o trabalho realizado com outros trabalhos correlatos.

O problema em questão denota a necessidade de utilização de modelagem que: (1) forneça o formalismo necessário para viabilizar verificações automáticas, (2) permita representação formal simples e acessível aos que não estão habituados a usar notações matemáticas e, por último, (3) seja um artefato claro e preciso para a comunicação com os diversos *stakeholders*, e também com os desenvolvedores de software.

Com isso em mente, não seria relevante analisar trabalhos que abordassem formalismos matemáticos textuais e simbólicos, ou simples transformações entre modelos e linguagens sem verificação automática. Buscamos, então, trabalhos afins, ou estudos comparativos, nas temáticas modelagem gráfica de invariantes, verificação automática de requisitos e efetividade de apresentações gráficas. Adicionamos, estudos sobre a efetividade e avaliação de notações gráficas para endereçar a necessidade (3), mencionada anteriormente para o problema em questão.

Encontramos oito trabalhos para formalismos com notação gráfica e, opcionalmente, com verificação automática de requisitos. Apesar do número reduzido de trabalhos, consideramos representativo, pois, os demais trabalhos abordavam formalismos textuais ou simbólicos, como o trabalho recente em [**?**], o que foi intencionalmente evitado no estudo. As seções deste capítulo descrevem os trabalhos relacionados e respectivas associações, deficiências, lacunas e motivações que justificam o foco e a relevância do presente trabalho.

7.1 Predominância de UML/OCL na Modelagem Gráfica e no Formalismo

Na revisão sistemática de Gonzalez *et al.* [11], são avaliadas abordagens de modelagem estática e de verificação de modelo, caracterizando o formalismo usado e o *feedback* da verificação, dentre outros aspectos. Em praticamente todas as abordagens encontradas (dezoito), apenas uma não utilizava modelo de diagrama de classes de UML, utilizava esquemas de banco de dados orientado a objetos (*Object Oriented Data Base Schema*). Uma segunda utilizava, adicionalmente ao diagrama de classes, o modelo EMF (Eclipse Modelling Framework). Quanto à utilização de OCL nos diagramas, apenas quatro não representavam as restrições do modelo por meio de OCL. Já em relação à presença de *feedback* da verificação, caiu para a metade definir (oito) a quantidade das que forneciam esse *feedback*.

Nas soluções propostas em UML2Alloy [1], CD2Alloy [21] , EMFToCSP [10], CD2Formula [26] , e UML-B [30], os esforços se concentram no problema da falta de precisão formal nos modelos UML agregando um mecanismo de verificação automática do modelo. O mesmo se dá nos modelos propostos para *Constraint Diagrams* em UML [16], *Constraint Trees* [17] (uma melhoria de *Constraint Diagrams*) e *Business Rule Diagram* [18]. Entretanto, para estes três estudos, a solução provê uma notação gráfica para modelar as invariantes, enquanto que para os primeiros, utilizou-se o formalismo da linguagem textual OCL dentro da linguagem gráfica UML.

Comparado a notação visual de GIRL com a notação proposta em [18], verificamos que a notação visual não abarca algumas das estruturas para expressar invariantes no *Business Rule Diagram*, como a implicação com quantificação. O diagrama modela regras de negócio associadas a objetos de classes, como exemplificado na Figura 7.1.

Nesse diagrama, a notação gráfica é usada apenas para objetos (retângulos tracejados), relacionamentos (retas entre objetos) e no contorno e compartimentos das outras estruturas (retângulos com rótulo vazando a linha e simples linhas, respectivamente). A identificação e distinção entre os tipos destas últimas estruturas é feita com uma notação textual, onde, em alguns casos, o texto entre chaves ({ }) também distingue os subtipos, a exemplo das operações lógicas $\{and\}$ e $\{not\}$ no item (b) da Figura 7.1. Além disso, restrições booleanas sobre os atributos dos objetos também são expressas em texto, como se vê no compartimento

inferior do item (a) da Figura 7.1, onde $o1 : i$ é o atributo i do objeto $o1$ da classe A. Consideramos, as estruturas de GIRL com correspondente em [18], como *logicalOperation*, *relationalOperation*, *quantification*, apresentam notação visual mais rica, explorando mais as variáveis visuais.

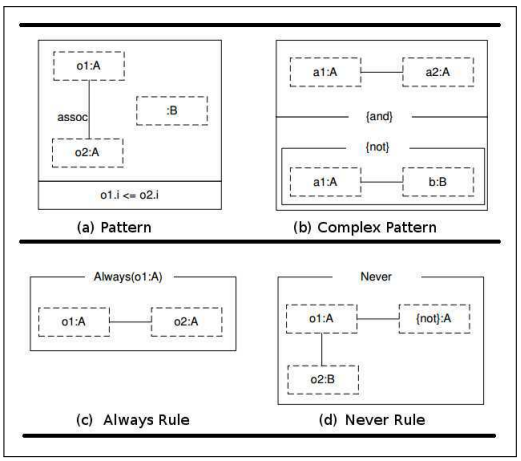

Figura 7.1: Exemplos de Estruturas Business Rule Diagram - Fonte: [18]

Uma melhoria em *Constraint Diagrams* [16] é apresentada em *Constraint Trees* [17], que apresenta uma solução unificada para misturar notações de restrições textuais (OCL) e gráficas, além de permitir organizar e visualizar o espaço de restrição de um modelo como mecanismo para gerenciar a complexidade [23]. A Figura 7.2 exibe um exemplo de *Constraint Trees*, onde os nós da árvore podem ser: (1) uma definição de uma variável em sentenças *let* ou *quantification*; (2) uma representação de uma fórmula booleana textual ou gráfica (OCL, *Constraint Diagrams*) ou; (3) um conectivo lógico (*AND, OR, XOR, NOT*). Nós do tipo (1) ou (2) devem ter apenas um filho. Não existe o conceito de implicação. A sintaxe de *Constraint Diagrams* está baseada na notação dos diagramas de Venn / Euler / Pierce e mostra as relações entre conjuntos e elementos. As setas representam a navegação. As aranhas (bolhas e estrelas) representam elementos. Uma estrela significa um elemento universalmente quantificado. Contornos (caixas e elipses) representam conjuntos.

Encontramos algumas semelhanças entre os conceitos de GIRL e de *Constraint Trees*, como, por exemplo, conjuntos singleton (apenas um elemento), cardinalidade, conjuntos com zero ou um elemento, conjunto vazio (vide Figura 7.3), mas com algumas diferenças nas notações e na localização das notações. Conjuntos singleton em GIRL são independentes, já em *Constraint Trees* precisam estar dentro de um conjunto. As notações com círculos

reduzidos para conjuntos vazios, e com zero ou um elementos, são usadas como multiplicidade em GIRL.

Consideramos que a modelagem estrutural é explícita nas entidades e relacionamentos de GIRL. Já nas *Constraint Trees* a estrutura fica subentendida, e só pode ser representada no diagrama de classes. E, além disso, o modelo não foi instanciado em ferramental que permite verificação automática [17].

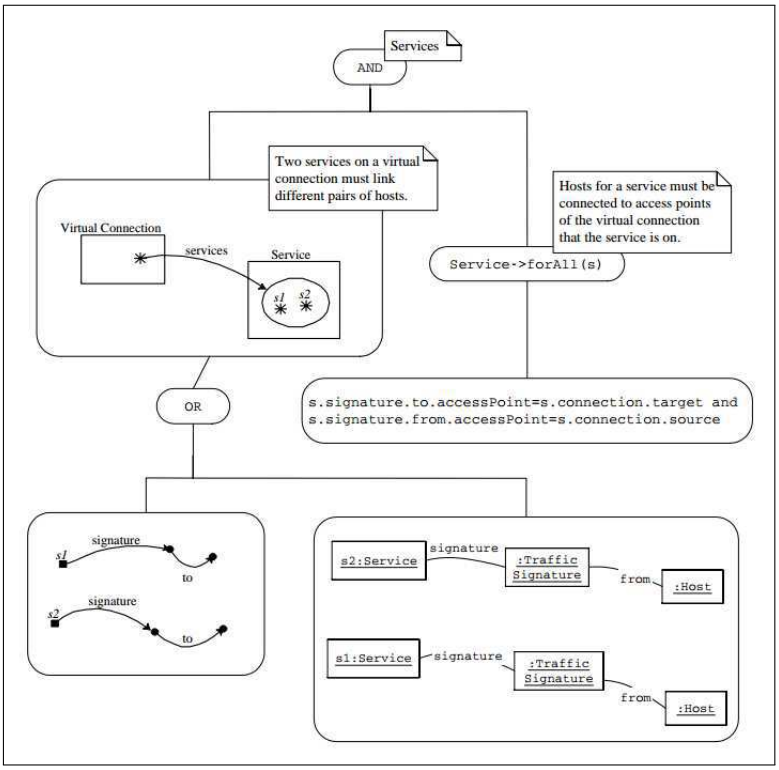

Figura 7.2: Exemplo de *Constraint Trees* e notação textual - Fonte: [17]

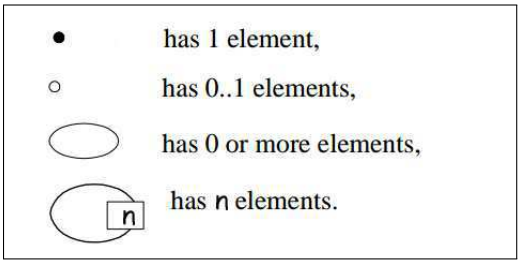

Figura 7.3: Notação de Conjuntos em *Constraint Trees* - Fonte: [17]

Verificamos a predominância da notação UML nos esforços de prover uma modelagem gráfica do software. No entanto, apesar de largamente disseminada, a linguagem UML é

mais utilizada para modelar artefatos voltados para a solução do software, na fase de projeto, do que para sua concepção, na fase de requisitos. Por esse motivo, tais artefatos são mais úteis para os desenvolvedores do que para os demais envolvidos. Isso reforça a necessidade de abordagens mais adequadas para a modelagem e verificação do software na fase de requisitos. UML não provê notação gráfica para invariantes. Para preencher essa lacuna, as modelagens em *Constraint Trees*[17] e *Business Rule Diagram* [18] disponibilizam meios de representar invariantes graficamente. Entretanto, carecem respectivamente de: ferramental para avaliação da utilidade da notação e de uma notação visual mais aderente aos princípios de uma notação efetiva [23].

7.2 Verificação Automática de Requisitos com Feedback ao Usuário

Em grande parte dos estudos mencionados por Gonzalez *et al.* [11], os modelos são verificados automaticamente com base em diversas abordagens disponíveis, como: SAT (*Boolean Satisfiability*), CSP (*Constraint Satisfaction Problem*) e SMT (*Satisfiability Modulo Theories*). Todavia, nas ferramentas associadas, não é frequente a presença de *feedback* ao usuário apresentando resultados compreensíveis e úteis para a validação dos requisitos.

Os trabalhos em [1], [21], [10], [26] e [30] agregam um mecanismo de verificação automática do modelo com *feedback* ou simulação de instâncias do modelo. Em [18] encontramos a verificação pela abordagem BDD-based (algorítimos baseados em *Binary Decision Diagram*), não deixando claro se apresenta *feedback*. Já os trabalhos [16] e [17] não apresentaram ferramental para verificação automática. Em GIRL, o mecanismo de verificação disponibilizado pelo Alloy provê *feedback* aceitável, apesar de ser uma notação gráfica distinta.

7.3 Efetividade e Avaliação de Notações Gráficas

Estudos de Moody [23] sugerem a utilização de princípios que estabeleçam notações visuais cognitivamente efetivas com base em disciplinas de outros campos do conhecimento como: comunicação, semiótica, design gráfico, percepção visual e cognição. São apresentados nove

princípios denominados *physics of notation* (PoN), que estão descritos na Tabela 7.1.

Vários estudos utilizaram os princípios de *physics of notation*. Alguns focaram na aplicação dos princípios [29] [5], outros, na sistematização da operacionalização dos princípios [34], e outros na avaliação da importância dos princípios por profissionais [35]. Este último indicou que, no projeto da notação visual, deve-se considerar: o usuário que vai modelar, a capacidade de desenho de modelos à mão, e o suporte ferramental. Também evidenciou que, as notações visuais mais usadas, na prática, são aquelas reguladas por órgãos de padronização (UML, BPMN, SysML), ainda que a literatura aponte evidências da falta de aderência dessas notações aos princípios PoN. Novamente, UML e SysML (uma extensão de um subconjunto de UML), estão no topo da lista dos mais utilizados, mesmo com deficiências.

A Seção 6.4.4 deste documento descreve a avaliação da linguagem GIRL, quanto à presença dos princípios PoN (vide Tabela 7.1). Percebemos indícios de adequação da notação aos princípios de: clareza semiótica, discriminação perceptiva, transparência semântica, expressividade visual, codificação dual e economia gráfica. Por outro lado, a notação de GIRL não está aderente aos princípios de: gerenciamento de complexidade, integração cognitiva e ajuste cognitivo. Na seção final deste trabalho, citamos trabalhos futuros para propor melhorias que comportem os princípios PoN desconsiderados ou menos considerados na linguagem GIRL. Em [29], notações da linguagem de modelagem i* são revisadas e melhoradas à luz dos princípios PoN.

Tabela 7.1: Princípios em Physics Of Notation

* Princípios PoN (Adaptado de [29]) *		
Princípio	**Descrição**	**GIRL suporta?**
Clareza semiótica (*semiotic clarity*)	Deve haver uma correspondência de um para um entre os elementos da linguagem e os símbolos gráficos	Sim
Discriminação perceptiva (*perceptual discriminability*)	Símbolos diferentes devem ser claramente distinguíveis uns dos outros	Sim
Transparência semântica (*semantic transparency*)	O uso de representações visuais que sugerem seu significado	Sim
Gerenciamento de complexidade (*complexity management*)	A notação inclui mecanismos explícitos para lidar com a complexidade	Não
Integração cognitiva (*cognitive integration*)	A notação inclui mecanismos explícitos para suportar a integração de informações de diferentes diagramas	Não

Tabela 7.1: (continuação)

* Princípios (Adaptado de [29]) *		
Princípio	**Descrição**	**GIRL suporta?**
Expressividade visual (*visual expressiveness*)	O uso de toda a gama e capacidades das variáveis visuais (formas, cores, texturas, tamanho, brilho, orientação, posição vertical e posição horizontal).	Sim
Codificação dual (*dual coding*)	Uso de texto para complementar gráficos	Sim
Economia gráfica (*graphic economy*)	O número de diferentes símbolos gráficos deve ser cognitivamente administrável	Sim
Ajuste cognitivo (*cognitive fit*)	Uso de diferentes dialetos visuais para diferentes tarefas e audiências	Não

Capítulo 8

Considerações Finais

Neste trabalho, apresentamos um meio para que desenvolvedores de software obtenham requisitos de qualidade, sem ambiguidades ou inconsistências. Com isso, a descoberta de falhas pode ser antecipada, o que afeta positivamente o custo do desenvolvimento do software. Propomos então a linguagem GIRL, uma linguagem gráfica para representar formalmente requisitos, permitindo a verificação automática da consistência dos requisitos modelados.

A linguagem GIRL é uma DSL (linguagem específica de domínio) para representação de invariantes, ou seja, de requisitos estáticos ou estruturais. É uma linguagem relativamente reduzida, se compararmos com a linguagem de propósito geral Alloy [13], utilizada para a verificação de consistência dos modelos GIRL. A linguagem contém doze estruturas classificadas em primárias e compostas, e mais uma estrutura de organização.

No estudo empírico realizado, investigamos se uma linguagem gráfica de modelagem de requisitos, com formalismo inerente à própria linguagem, seria um instrumento útil para detecção antecipada de inconsistências em requisitos de software e de fácil utilização, na percepção de analistas de requisitos.

A estratégia de pesquisa adotada foi do tipo *judgment task* [32]. Dez profissionais desenvolvedores de software, que lidavam com atividades de levantamento de requisitos, utilizaram e avaliaram a linguagem GIRL. Além desses, um outro sujeito tomou parte na pesquisa, na qualidade de *expert* em matemática e cliente de projetos de software, não sendo profissional de desenvolvimento de software.

A execução do estudo se deu em sessões individuais divididas em duas fases: a primeira, um treinamento na linguagem, dividido em módulos sucessivos em que o sujeito assistia a

um vídeo e registrava uma autoavaliação de entendimento das estruturas abordadas.

Na segunda fase, o sujeito utilizava a linguagem GIRL para a modelagem e verificação de quatro conjuntos de requisitos invariantes, sucessivamente. Ao final da modelagem, o sujeito registrava o resultado da comparação da sua solução com uma solução fornecida. Em seguida, relatava uma avaliação dos pontos positivos e negativos das estruturas da linguagem, do editor e do mecanismo de verificação, além de registrar o grau de utilidade da linguagem GIRL para representar formalmente e verificar requisitos.

Em toda a sessão, eram registrados os tempos de permanência nas páginas do instrumento de coleta dos dados (questionário online), que correspondiam às divisões das fases do treinamento e da utilização. O questionário online utilizado, determinava o cumprimento sucessivo de cada etapa dentro das duas fases da sessão. Nesse ínterim, um observador tomava notas de dúvidas e expressões do sujeito, como também os passos para a construção do modelo.

8.1 Resultados

Procedemos duas análises sobre os dados levantados. Uma análise de caráter qualitativo sobre a exatidão dos modelos elaborados, registros de pontos positivos e negativos da linguagem e aspectos relevantes registrados nas observações de cada participante. E a outra, uma análise quantitativa, sendo as variáveis independentes: o conhecimento em lógica, a experiência com requisitos e o nível de formação acadêmica e; as variáveis dependentes: os tempos registrados, a quantidade de dúvidas e o grau de entendimento das estruturas da linguagem.

A análise dos dados coletados revelou que a linguagem GIRL se constitui num instrumento viável para alcançar requisitos de qualidade. Os profissionais envolvidos no estudo não apresentaram resistência ou tédio durante o treinamento e a utilização da linguagem GIRL, conforme observações de expressões verbais e comportamentais registradas para cada profissional. Isso reforça a evidência de que profissionais utilizam bem os formalismos embutidos na automação e abstração de linguagens de modelagem [14].

Todos os participantes registraram um grau elevado de utilidade da linguagem para a representação formal e verificação de requisitos. Para o registro do julgamento do grau de

utilidade, usando uma escala de um a dez, encontramos a mediana 9.

Essa constatação da percepção de utilidade da linguagem é reforçada nas entrelinhas de sugestões de extensão, como a geração de código do modelo e a geração de casos de testes com as instâncias da verificação. Isso denota que os participantes vislumbraram a economia de tempo ao transferir bons modelos de requisitos para as etapas seguintes do desenvolvimento do software — codificação e testes. Foi identificado um potencial para diminuir o retrabalho de programação e compensar o custo do processo de análise de requisitos na ferramenta.

Além disso, as expressões de satisfação da maioria dos participantes com a possibilidade de verificar, ou testar, ou simular, requisitos modelados, demonstra a relevância da linguagem para os profissionais que participaram do estudo.

Os requisitos foram modelados corretamente em praticamente todos os participantes. Apenas um apresentou uma modelagem equivocada em uma das oito invariantes, mas em seguida representou corretamente outra invariante similar. Um exemplo de interpretação ambígua: o termo "tem" foi entendido como "contém" e representado com a estrutura *Containment*, quando deveria ser uma relação (*Relationship*).

A notação da linguagem GIRL apresenta transparência semântica [23] nos símbolos adotados que têm correspondência direta de significado com a teoria dos conjuntos. Já nas estruturas que não fazem parte dessa teoria, o significado da notação foi menos intuitivo. Isso aponta para a necessidade de melhorias na representação gráfica.

A análise quantitativa não se mostrou conclusiva pelo tamanho da amostra. No entanto, evidenciamos que pessoas com maior conhecimento em lógica representaram mais rapidamente requisitos contendo implicação e quantificação, que são estruturas da linguagem com pouca associação a conceitos da Engenharia de Software. Para as demais estruturas não identificamos correlação entre tempo gasto para representar e bagagem em lógica. Nas demais variáveis independentes - experiência com requisitos e nível de formação acadêmica, não identificamos influência nos tempos de utilização e na quantidade de dúvidas.

A seguir apresentamos os principais resultados evidenciados no estudo, divididos por questão de pesquisa estabelecida, quais sejam:

- **QP1**: *Nível de dificuldade percebida no aprendizado e utilização da linguagem GIRL* - O nível de dificuldade foi considerado baixo para as estruturas mais simples e com

conceitos e notações mais semelhantes aos conceitos da Engenharia de Software e da teoria de conjuntos. Em estruturas mais complexas e com conceitos mais próximos à lógica, o nível de dificuldade foi maior. Na análise quantitativa observamos que quanto maior o nível de conhecimento em lógica do sujeito, menor o tempo de modelagem em requisitos com estas estruturas mais complexas.

- **QP2**: *Exatidão dos requisitos representados durante a utilização da linguagem GIRL* - A representação foi correta em praticamente todos os requisitos, apesar das dificuldades relatadas nas estruturas mais complexas. Apenas um participante representou equivocadamente uma, dentre as oito invariantes, mas, em seguida representou corretamente uma invariante similar.

- **QP3**: *Utilidade percebida na verificação automática de requisitos em GIRL* - para todos os participantes a verificação foi considerada útil, com ocorrência de expressões de satisfação com a possibilidade de testar requisitos.

- **QP4**: *Efetividade da representação gráfica da linguagem GIRL* - evidências da efetividade da notação por meio de observações de reações dos participantes, denotando a presença de princípios recomendados para uma notação visual efetiva [23]. No entanto, houve um registro negativo na avaliação do participante quanto a uma variável de expressividade visual, o uso de cores.

8.2 Limitações

Podemos dividir as limitações do estudo em: limitações no objeto do estudo e limitações no método do estudo empírico. O objeto do estudo, a linguagem GIRL — enquanto meio para viabilizar a representação formal e verificação de requisitos, apresenta as seguintes limitações:

- A linguagem representa apenas requisitos estáticos ou estruturais, carecendo de meios para representar comportamento do software. Podemos representar em GIRL a invariante: "Uma Conta pode ter um ou mais Clientes como Titular". Mas não podemos representar o procedimento para um cliente abrir uma conta.

- Falha nas transformações de algumas inconsistências nos modelos, produziram falhas na verificação. Um exemplo disso foi a representação de relações com mesmo nome para *targets* diferentes numa mesma entidade.

- Falta de automatização na verificação do modelo. Transformações e a verificação do código Alloy gerado eram chamadas manualmente.

- Diferenças das notações na modelagem (GIRL) e no retorno da verificação (Alloy).

- O conjunto de instâncias da verificação pode conter entidades e relacionamentos irrelevantes aumentando o espaço de pesquisa na avaliação das instanciações.

- Dificuldade de exibição de conteúdo de algumas estruturas compostas no momento da criação dessas estruturas por limitação no *framework* de desenvolvimento de editores gráficos *Sirius*[1].

Quanto ao método do estudo empírico identificamos as seguintes limitações:

- Tamanho da amostra pequeno, com 11 participantes, o que compromete generalizações.

- Por limitações de recursos e tempo, nem todas as estruturas da linguagem foram utilizadas na modelagem dos requisitos durante o estudo. Faltam indícios da efetividade das seguintes estruturas que foram abordadas no treinamento, mas não foram necessárias na utilização: *Containment*, *Cardinality*, *SetOperation*, *LogicalOperation*. Além dessas, falta a estrutura *RelationshipOperation* que não entrou no escopo do estudo empírico.

- Exemplos de aplicação da linguagem, que foram exibidos nos vídeos, trouxeram confusão entre conceitos: *containment vs* extensão, *quantification vs multiplicity*, *multiplicity vs cardinality*, *enumeration*. No exemplo do conceito de *logicalOperation*, empregamos as estruturas *containment* para expressar as invariantes: "Uma Conta Corrente está contida em Conta" e "Uma Conta Poupança está contida em Conta". Mas essas invariantes estão implícitas no conceito de extensão das entidades Conta Corrente e Conta Poupança estendem Conta, que foi apresentado anteriormente.

[1] Sirius — informações disponíveis em: https://www.eclipse.org/sirius/

- A presença e mediação do observador, apesar de facilitar no caso de dúvidas, insere um risco de desconforto com a tomada de notas, como expressou um dos participantes: *"Tô me sentindo num psicólogo. Tudo que eu faço ela anota!" (risos)*. Uma alternativa é eliminar a tomada de notas e gravar em vídeo as ações do sujeito em todo a sessão do estudo.

Identificamos também duas divergências de conceitos nas estruturas da linguagem GIRL. A primeira, está na denominação *Complement* da operação de conjuntos. A notação gráfica exibe o identificador como uma diferença de conjuntos (ex. $A - B$), porém, o termo usado na interface de criação e nas definições é o complemento, cuja notação seria B', B^{-} ou B^{C}. Makinson [20] esclarece que, a diferença entre conjuntos é um complemento relativo e que, em computação, não importa essa divergência. Ainda assim, é melhor ajustar o termo para 'diferença' e corrigir a denominação da operação na linguagem GIRL.

A outra possível diferença conceitual diz respeito à notação gráfica de GIRL para as operações semelhantes aos diagramas de Venn. A teoria acadêmica da notação de Venn para expressar declarações existenciais, estabelece que as partes sombreadas ou hachuradas verticalmente representam a ausência, como descrito em [22]. No entanto, utilizamos em GIRL a representação dos diagramas de Venn para operações de conjuntos e não para declarações [20]. Utilizamos esse último conceito em GIRL, que é o mais disseminado para os diagramas de Venn.

8.3 Trabalhos Futuros

Apesar dos resultados animadores, lacunas e falhas encontradas podem ser resolvidas na continuação do trabalho. A seguir, elencamos os trabalhos futuros em termos de: melhorias do editor gráfico, melhorias na notação visual, expansão da linguagem e melhorias no método do estudo.

Quanto ao editor gráfico podemos agregar recursos de auto-completar para os identificadores de estruturas (*entity* e *relationship*) evitando inconsistências sintáticas do modelo. Outra melhoria, diz respeito aos arranjos iniciais do conteúdo de estruturas compostas, como cardinalidade e implicação, que escondem os elementos durante a criação, requerendo ação do usuário exibir corretamente o conteúdo.

Também podemos automatizar a verificação dos modelos em GIRL, com delimitação de invariantes para o escopo da verificação, diminuindo a quantidade de elementos nas instâncias verificadas, tornando mais rápida a verificação. Nesse tema, temos um trabalho em conclusão para apresentar as instâncias da verificação na mesma notação GIRL. Para garantir a consistência nas melhorias propostas para a ferramenta, o procedimento de testes deve ser aprimorado para evitar falhas, como as encontradas durante o estudo, e prover maior robustez à ferramenta.

A notação visual de GIRL pode ser melhorada, com uma análise mais aprofundada da adequação da notação gráfica aos princípios PoN [23], que apontaria melhorias nas estruturas com menor grau de entendimento, como implicação e quantificação. Quanto à implicação, a estrutura considerada mais complexa e de difícil aplicação, sua notação gráfica e sua usabilidade podem ser melhoradas, com uma delimitação mais clara dos compartimentos da premissa e da conclusão, permitindo a inclusão de componentes diretamente nesses compartimentos. Também se faz necessário um trabalho interdisciplinar com profissionais de *design* gráfico para definir a melhor notação para cada estrutura.

Trabalhos futuros para fornecer meios de modelagem e verificação do comportamento do software em GIRL, elevariam significativamente os benefícios da linguagem, para o alcance de requisitos precisos e consistentes.

Outra melhoria da linguagem GIRL que afeta o processo de desenvolvimento como um todo, seria a adição de meios para promover a rastreabilidade e manutenibilidade dos requisitos modelados. Essas características relevantes para a qualidade de um conjunto de requisitos não estão presentes em GIRL. Sendo assim, constitui-se uma meta de melhoria de GIRL, implementar mecanismos para explicitar as dependências entre as invariantes e delinear o encadeamento entre elas.

Considerando o estudo empírico realizado, identificamos os seguintes pontos para melhoria em novos estudos: uso de todas as estruturas da linguagem; selecionar amostra com tamanho mais representativo; utilizar recursos de gravação de vídeo e voz nas sessões do estudo, para substituir anotações; reestruturação dos exemplos do treinamento e dos vídeos e; avaliar o uso da linguagem GIRL com quantidade maior de requisitos em contexto real.

Bibliografia

[1] Kyriakos Anastasakis, Behzad Bordbar, Geri Georg, and Indrakshi Ray. Uml2alloy: A challenging model transformation. In *Proceedings of the 10th International Conference on Model Driven Engineering Languages and Systems*, MODELS'07, pages 436–450, Berlin, Heidelberg, 2007. Springer-Verlag.

[2] Ralph-Johan Back. Invariant based programming: basic approach and teaching experiences. *Formal Aspects of Computing*, 21(3):227–244, May 2009.

[3] Banco do Nordeste do Brasil S.A, Fortaliza - CE. *Request for Proposal - Aquisição e Implantação de uma Solução de Automação Bancária Multicanal - Anexo II - Requisitos Funcionais*, 2014. Disponível em: https://www.bnb.gov.br/documents/45078/47401/anexo_ii_requisitos_funcionais...

[4] Jordi Cabot and Martin Gogolla. Object constraint language (ocl): A definitive guide. In *Proceedings of the 12th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Formal Methods for Model-driven Engineering*, SFM'12, pages 58–90, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] P. Caire, N. Genon, P. Heymans, and D. L. Moody. Visual notation design 2.0: Towards user comprehensible requirements engineering notations. In *2013 21st IEEE International Requirements Engineering Conference (RE)*, pages 115–124, July 2013.

[6] D. Callele, K. Wnuk, and B. Penzenstadler. New frontiers for requirements engineering. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 184–193, Sept 2017.

[7] B. H. C. Cheng and J. M. Atlee. Research directions in requirements engineering. In *Future of Software Engineering (FOSE '07)*, pages 285–303, May 2007.

[8] Ivo Damyanov and Mila Sukalinska. Article: Domain specific languages in practice. *International Journal of Computer Applications*, 115(2):42–45, April 2015. Full text available.

[9] D. Méndez Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M.-T. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R. Spinola, A. Tuzcu, J. L. de la Vara, and R. Wieringa. Naming the pain in requirements engineering. *Empirical Software Engineering*, 22(5):2298–2338, Oct 2017.

[10] C. A. Gonzalez, F. Büttner, R. Clariso, and J. Cabot. Emftocsp: A tool for the lightweight verification of emf models. In *2012 First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA)*, pages 44–50, June 2012.

[11] Carlos A. González and Jordi Cabot. Formal verification of static software models in mde: A systematic review. *Information and Software Technology*, 56(8):821 – 838, 2014.

[12] Tony Hoare. Science and engineering: A collusion of cultures. In *The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25-28 June 2007, Edinburgh, UK, Proceedings*, pages 2–9. IEEE Computer Society, 2007.

[13] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.

[14] C. Jaspan, M. Keeling, L. Maccherone, G. L. Zenarosa, and M. Shaw. Software mythbusters explore formal methods. *IEEE Software*, 26(6):60–63, Nov 2009.

[15] Ralph D. Jeffords and Constance L. Heitmeyer. A strategy for efficiently verifying requirements. *SIGSOFT Softw. Eng. Notes*, 28(5):28–37, September 2003.

[16] Stuart Kent. Constraint diagrams: Visualizing invariants in object-oriented models. *SIGPLAN Not.*, 32(10):327–341, October 1997.

[17] Stuart Kent and John Howse. Constraint trees. In *Object Modeling with the OCL, The Rationale Behind the Object Constraint Language*, pages 228–249, London, UK, UK, 2002. Springer-Verlag.

[18] Deepali Kholkar, G. Murali Krishna, Ulka Shrotri, and R. Venkatesh. Visual specification and analysis of use cases. In *Proceedings of the 2005 ACM Symposium on Software Visualization*, SoftVis '05, pages 77–85, New York, NY, USA, 2005. ACM.

[19] K. Rustan M. Leino and Angela Wallenburg. Class-local object invariants. In *Proceedings of the 1st India Software Engineering Conference*, ISEC '08, pages 57–66, New York, NY, USA, 2008. ACM.

[20] David Makinson. *Sets, Logic and Maths for Computing*. Springer Publishing Company, Incorporated, 2nd edition, 2012.

[21] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Cd2alloy: Class diagrams analysis using alloy revisited. In *Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems*, MODELS'11, pages 592–607, Berlin, Heidelberg, 2011. Springer-Verlag.

[22] Amirouche Moktefi and Ahti-Veikko Pietarinen. On the diagrammatic representation of existential statements with venn diagrams. *Journal of Logic, Language and Information*, 24(4):361–374, Dec 2015.

[23] D. Moody. The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, Nov 2009.

[24] Object Management Group® (OMG®). *Object Constraint Language - Specification Document - Version 2.4*, 2014. Disponível em: https://www.omg.org/spec/OCL/2.4/PDF.

[25] David Lorge Parnas. The use of mathematics in software quality assurance. *Frontiers of Computer Science*, 6(1):3–16, Feb 2012.

[26] Beatriz Pérez and Ivan Porres. Reasoning about uml/ocl class diagrams using constraint logic programming and formula. *Information Systems*, 2018.

[27] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley Publishing, 1st edition, 2012.

[28] Arianna Zoila Olivera Salmon. *Modelagem E Análise De Requisitos De Sistemas Automatizados Usando UML E Redes De Petri*. PhD thesis, Escola Politécnica, University of São Paulo, 2017. São Paulo, retrieved 2018-10-09 from www.teses.usp.br.

[29] M. Santos, C. Gralha, M. Goulão, J. Araújo, and A. Moreira. On the impact of semantic transparency on understanding and reviewing social goal models. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 228–239, Aug 2018.

[30] Colin Snook and Michael Butler. Uml-b: Formal modeling and design aided by uml. *ACM Trans. Softw. Eng. Methodol.*, 15(1):92–122, January 2006.

[31] Ian Sommerville. *Software Engineering*. Addison-Wesley, Harlow, England, 9 edition, 2010.

[32] K. Stol and B. Fitzgerald. A holistic overview of software engineering research strategies. In *2015 IEEE/ACM 3rd International Workshop on Conducting Empirical Studies in Industry*, pages 47–54, May 2015.

[33] Ashish Tiwari, Harald Rueß, Hassen Saïdi, and Natarajan Shankar. A technique for invariant generation. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2001, pages 113–127, London, UK, UK, 2001. Springer-Verlag.

[34] D. v. d. Linden, A. Zamansky, and I. Hadar. A framework for improving the verifiability of visual notation design grounded in the physics of notations. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 41–50, Sept 2017.

[35] Dirk van der Linden, Irit Hadar, and Anna Zamansky. What practitioners really want: requirements for visual notations in conceptual modeling. *Software & Systems Modeling*, Feb 2018.

[36] Rini Van Solingen and Egon Berghout. *The goal/question/metric method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[37] Markus Völter, Sebastian Benz, Christian Dietrich, Birgit Engelmann, Mats Helander, Lennart CL Kats, Eelco Visser, and Guido Wachsmuth. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013.

[38] Karl E Wiegers and Joy Beatty. *Software Requirements 3*. Microsoft Press, Redmond, WA, USA, 2013.

[39] Karl Eugene Wiegers. *Software Requirements*. Microsoft Press, Redmond, WA, USA, 2 edition, 2003.

[40] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Bjöorn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

Apêndice A

E-mail Convite

Olá,

"Se você atua (ou atuou nos últimos 18 meses) em projeto de desenvolvimento de software, executando atividades de elicitação de requisitos, convidamos você para participar de nossa pesquisa e assim, contribuir para a melhoria de práticas na Engenharia de Requisitos.

Na pesquisa vamos analisar a adequação de uma ferramenta gráfica de representação e verificação automática de requisitos para facilitar a descoberta antecipada de falhas.

Para obter maiores detalhes sobre a pesquisa e, caso deseje, confirmar e agendar a sua participação, acessar o link:

Chamada para Participação em Pesquisa (Engenharia de Requisitos)

Caso sua resposta seja positiva, entraremos em contato para confirmar detalhes da sua participação."

Atenciosamente,

Marzina Vidal Negreiros Bezerra Analista de TI - Serviços de Tecnologia da Informação - STI/SEPLAN/UFCG Mestranda em Ciências da Computação - COPIN/UFCG Universidade Federal de Campina Grande - UFCG

Apêndice B

Questionário Convite

O questionário respondido na hora da aceitação do convite tem a seguinte sequência de páginas:

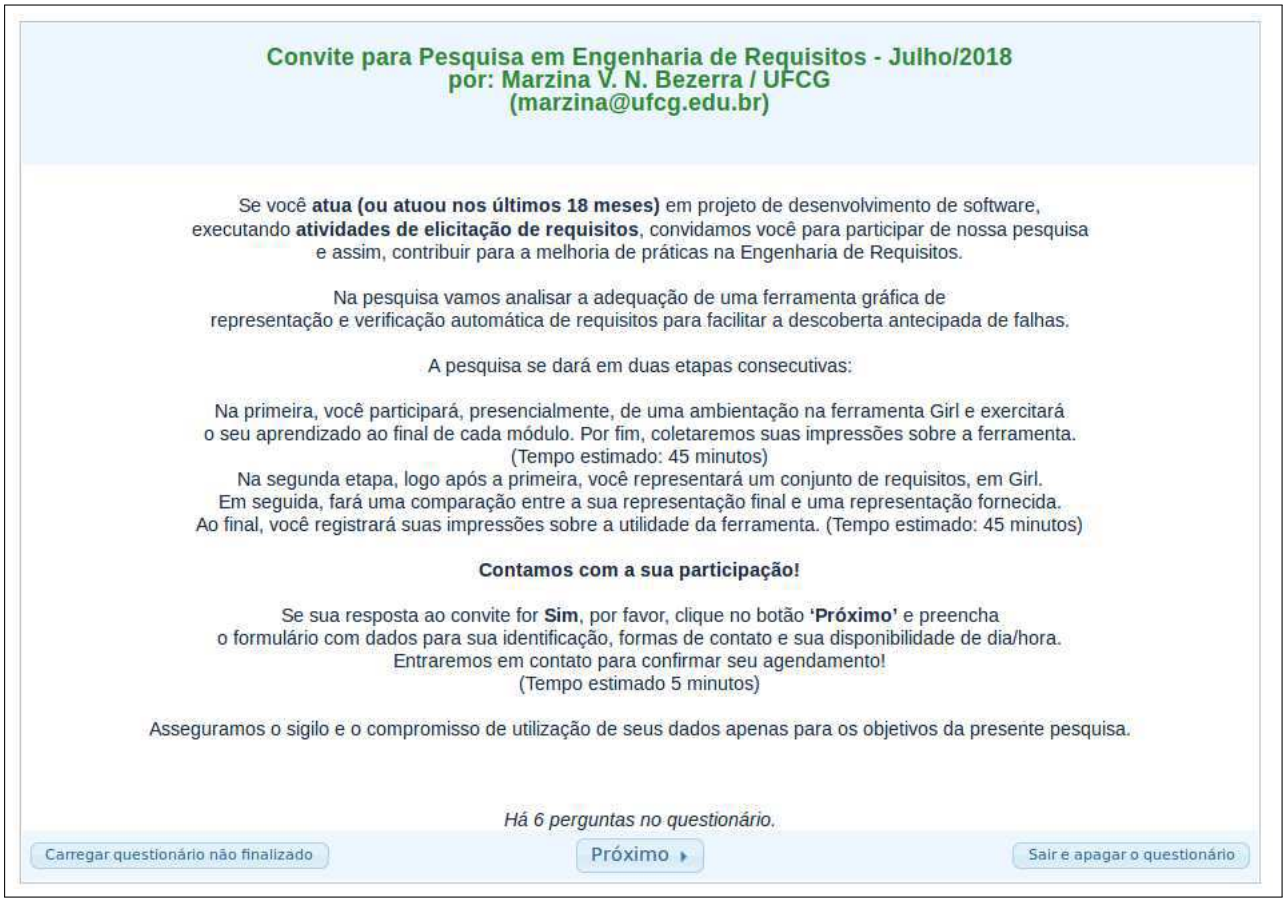

Figura B.1: Convite - Apresentação

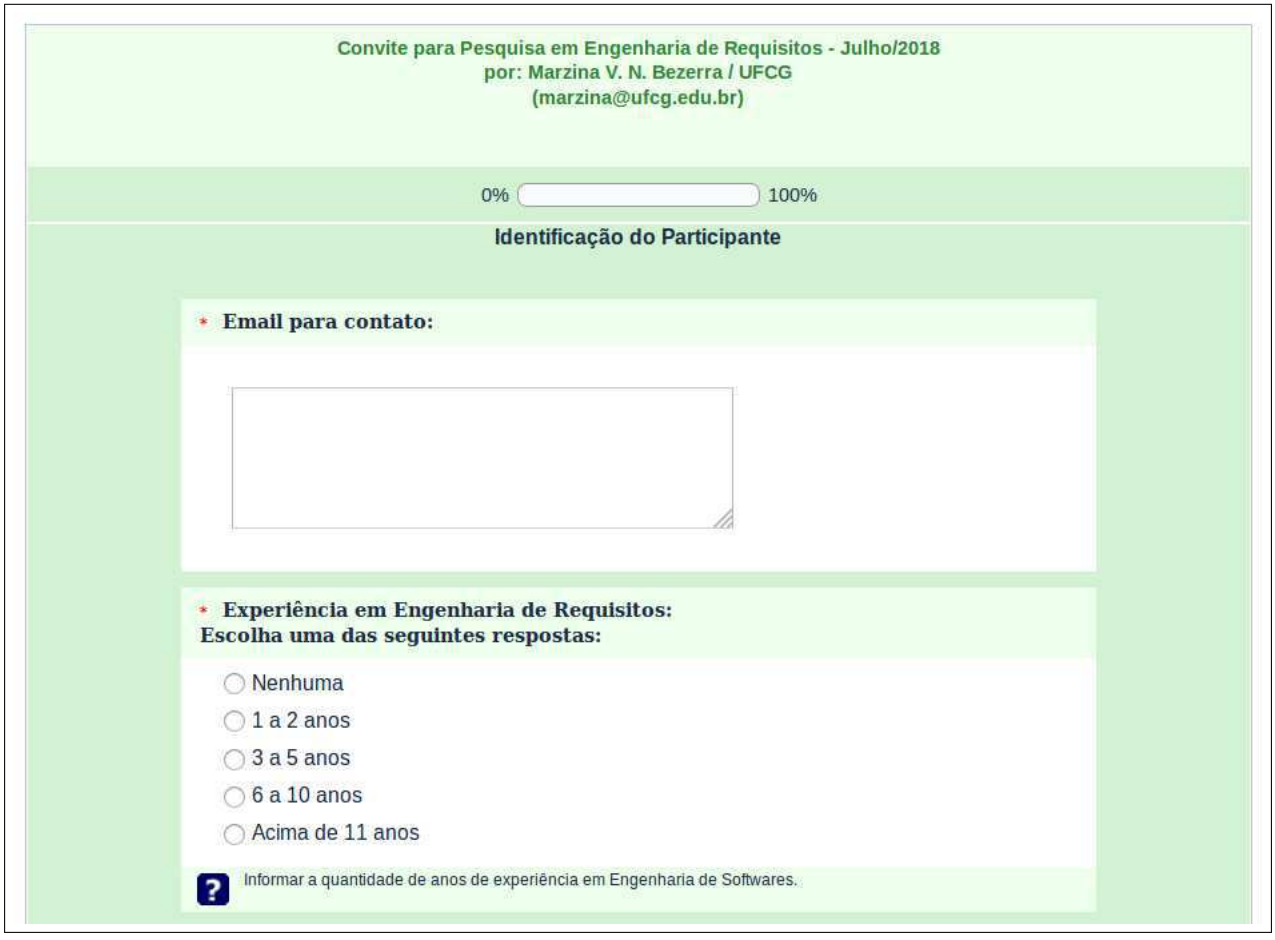

Figura B.2: Convite - Caracterização do Participante - parte 1

Figura B.3: Convite - Caracterização do Participante - parte 2

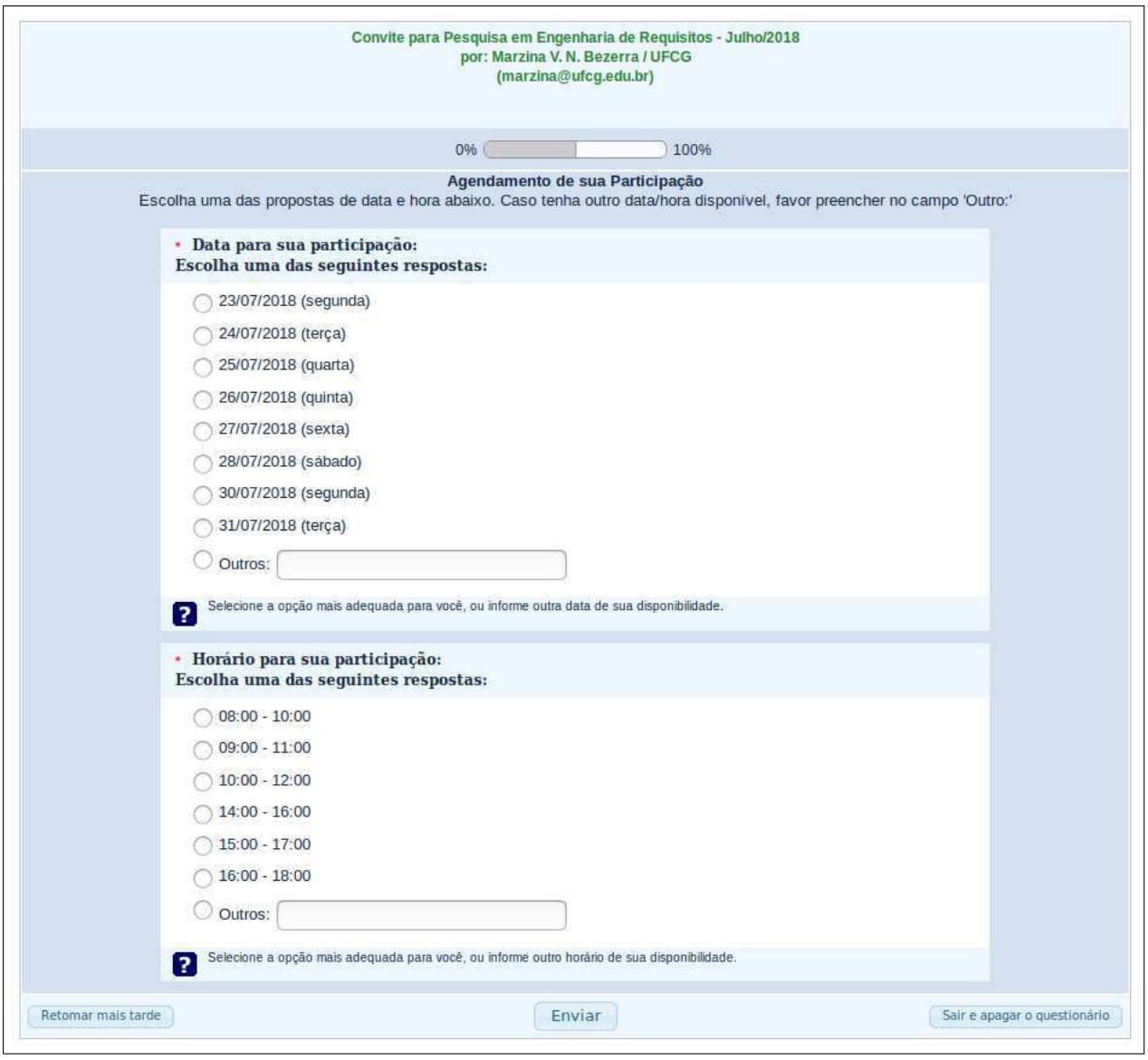

Figura B.4: Convite - Sugestão de Agendamento

Apêndice C

Questionário Online

Cada sessão do estudo foi guiada pelo Questinário Online, cuja sequência de páginas está apresentada nas figuras a seguir. O Questionário foi constrído na ferramenta LimeSurvey (Version 2.05+ Build 141210) e ficou disponível em:

Questionário

Os vídeos de cada módulo e da solução proposta estão disponíveis em:

Módulo 1

Módulo 2

Módulo - 3 - 1

Módulo - 3 - 2

Módulo - 4

Módulo - 5

Módulo - 6

Solução Proposta

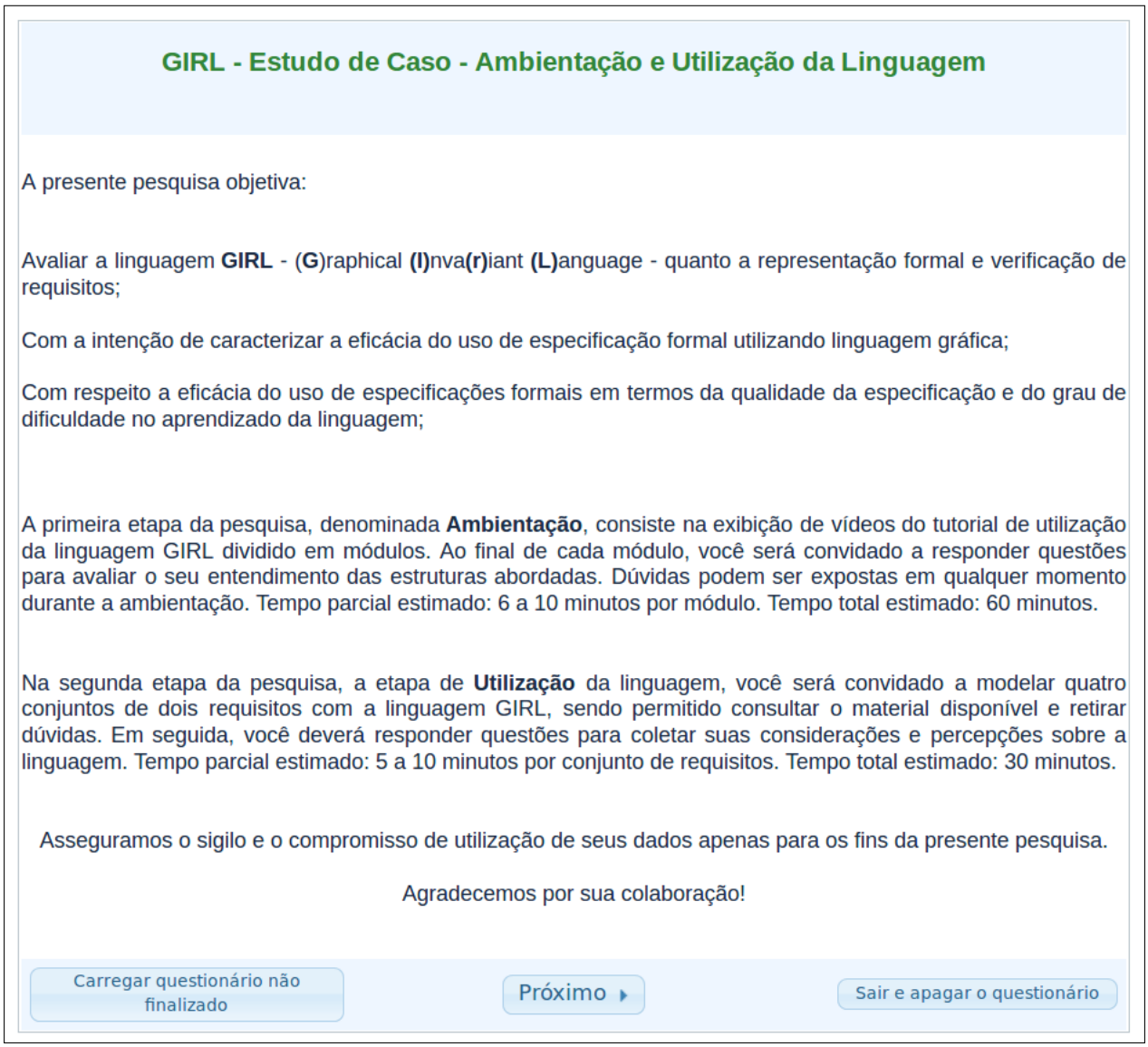

Figura C.1: Questionário - Apresentação

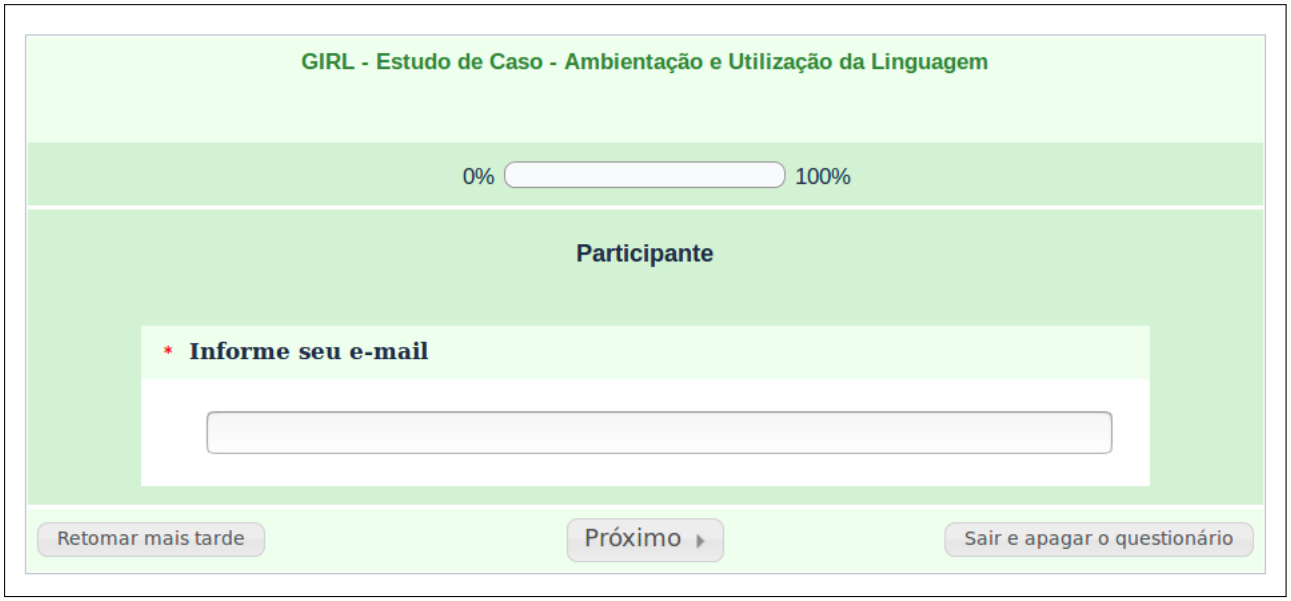

Figura C.2: Questionário - Identificação

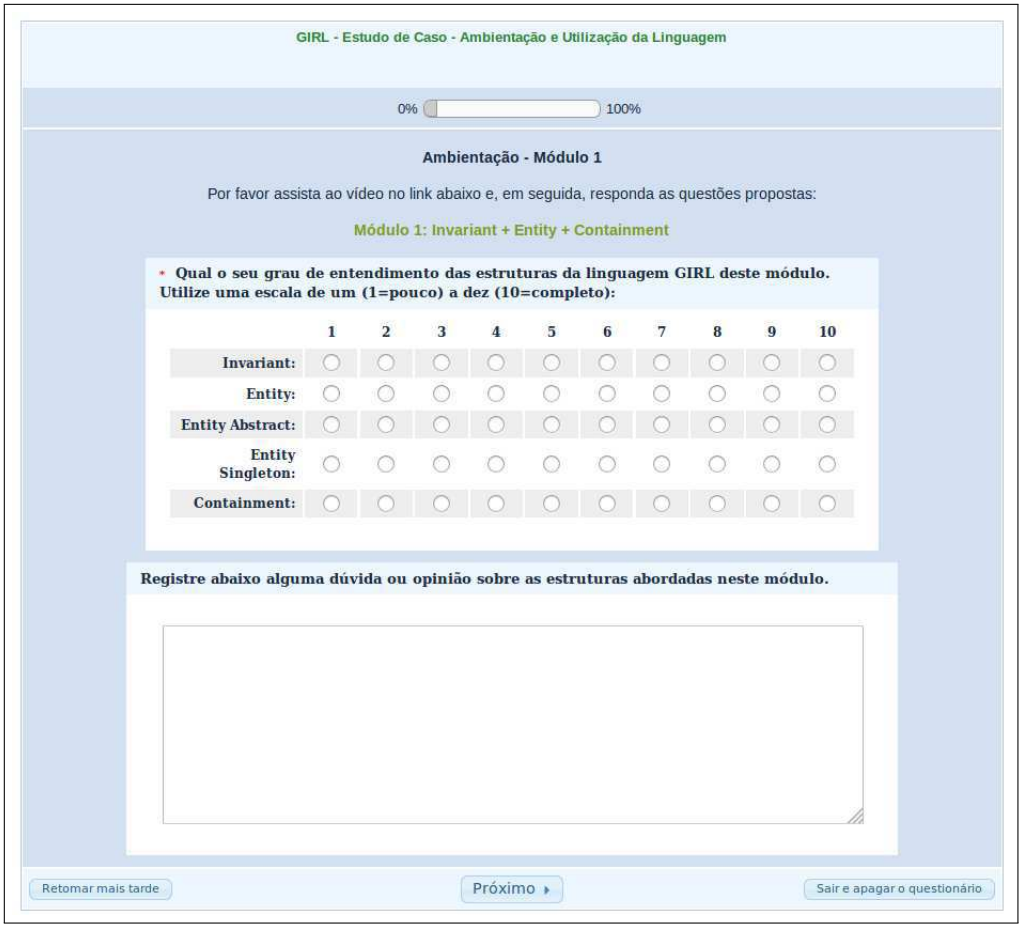

Figura C.3: Questionário - Módulo 1

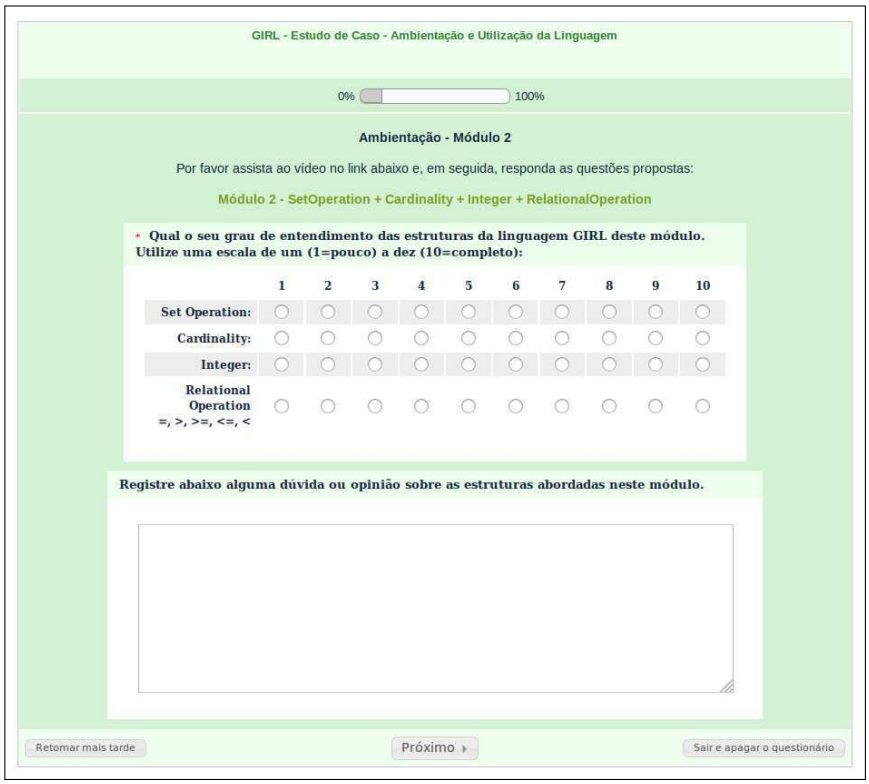

Figura C.4: Questionário - Módulo 2

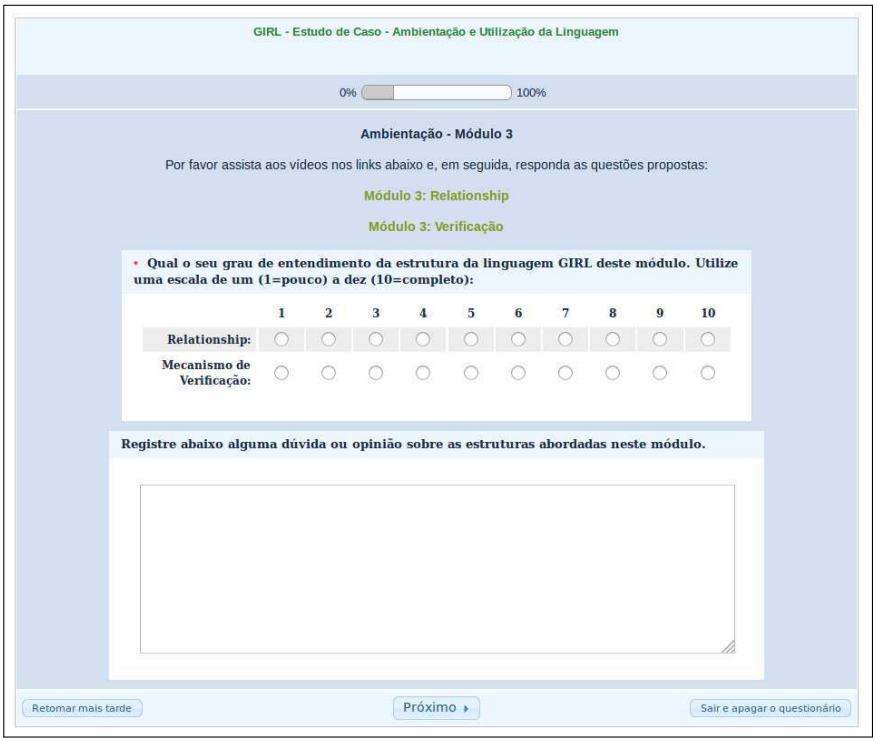

Figura C.5: Questionário - Módulo 3

Figura C.6: Questionário - Módulo 4

Figura C.7: Questionário - Módulo 5

Figura C.8: Questionário - Módulo 6

Figura C.9: Questionário - Conjunto de Requisitos 1

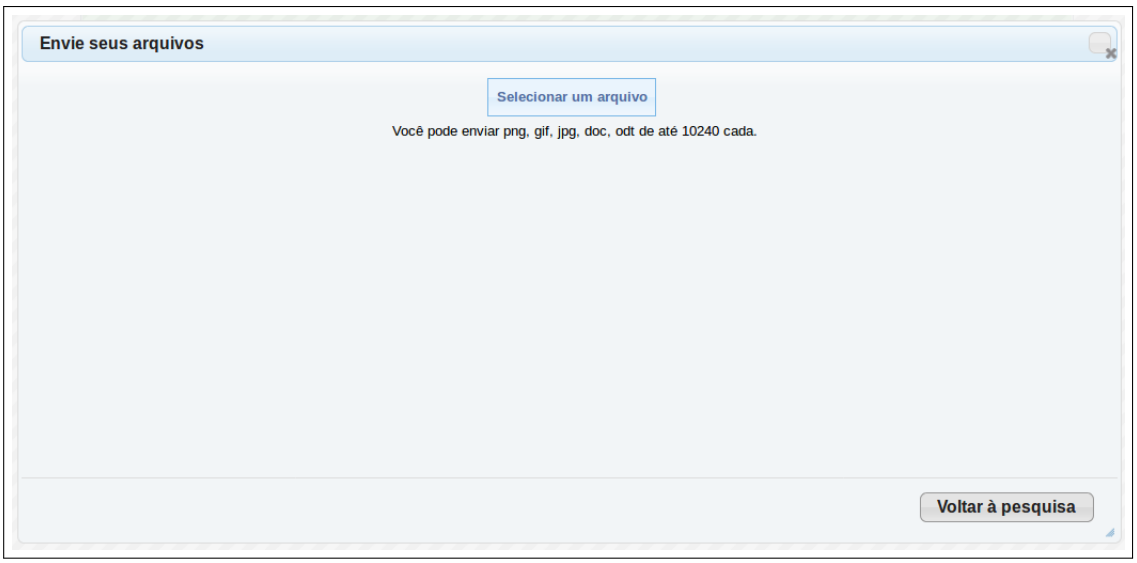

Figura C.10: Questionário - Envio do Arquivo com Modelo Elaborado pelo Participante

Figura C.11: Questionário - Conjunto de Requisitos 2

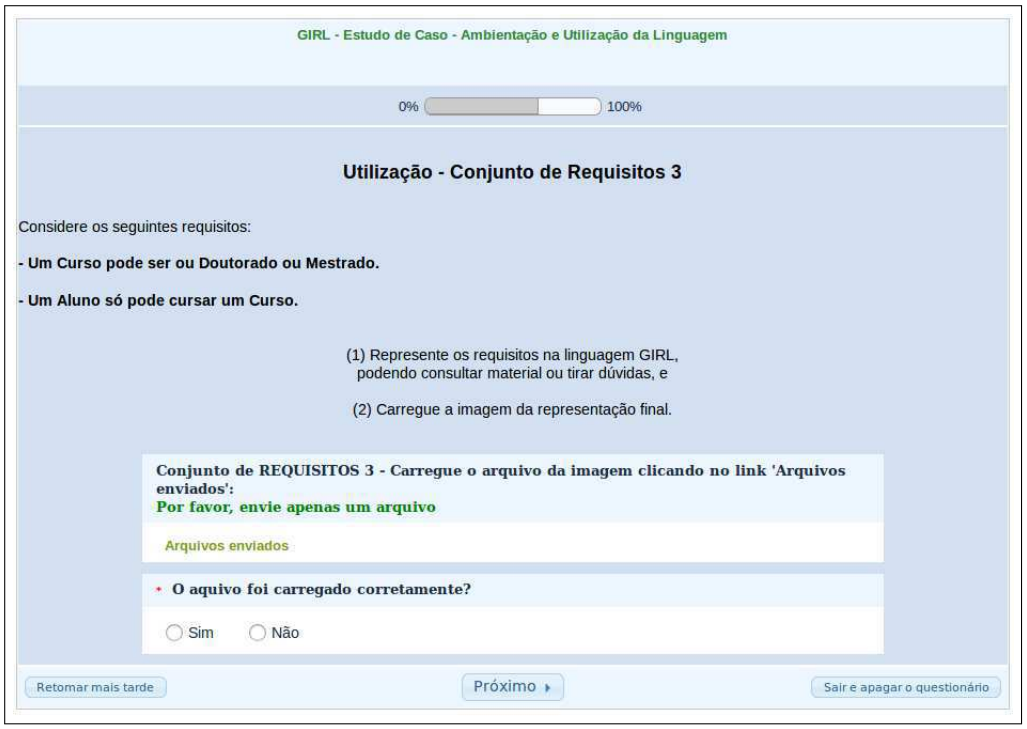

Figura C.12: Questionário - Conjunto de Requisitos 3

Figura C.13: Questionário - Conjunto de Requisitos 4

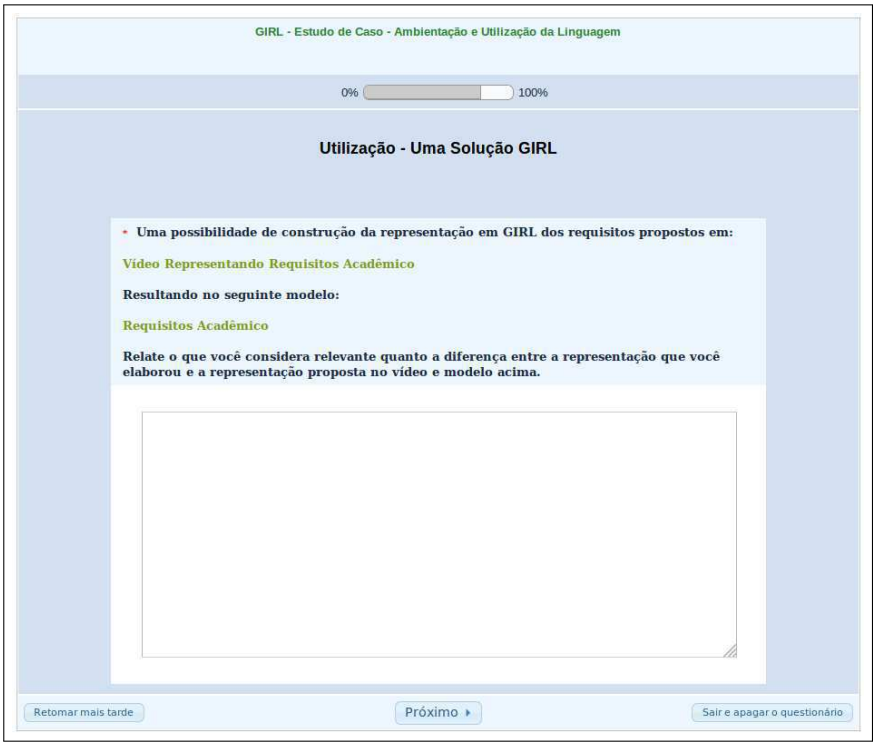

Figura C.14: Questionário -Modelo da Solução Proposta para Comparação

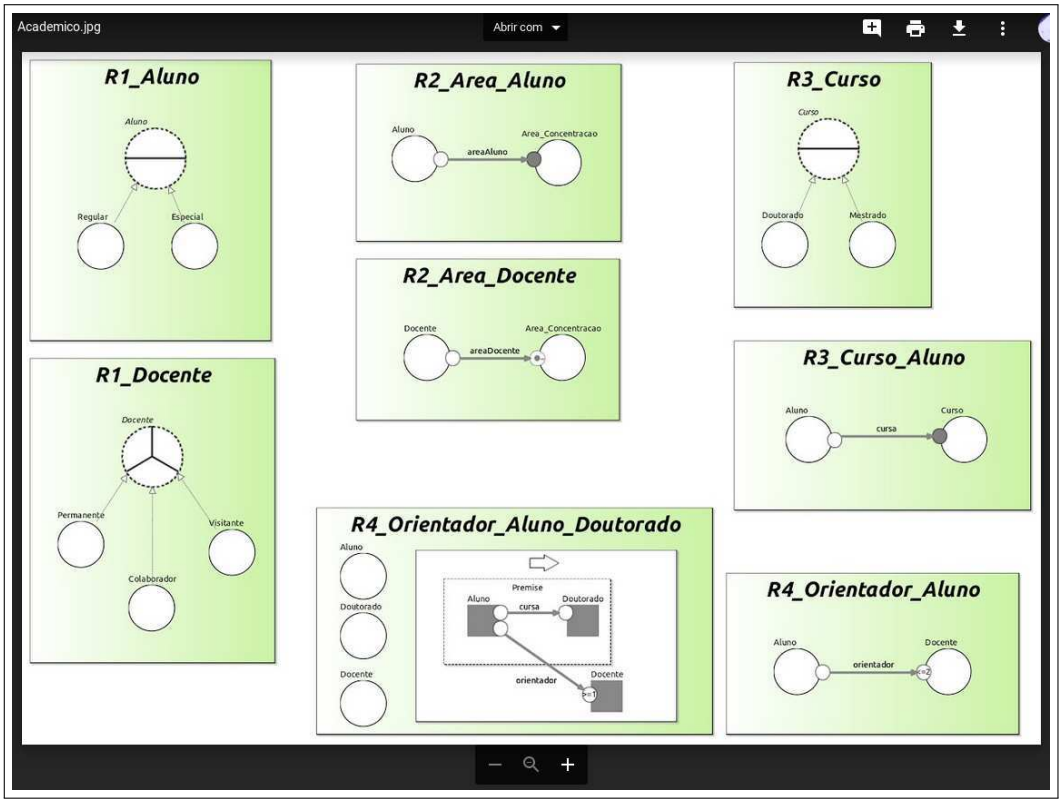

Figura C.15: Questionário - Solução Proposta

Figura C.16: Questionário - Avaliação da Linguagem GIRL - Questão 1

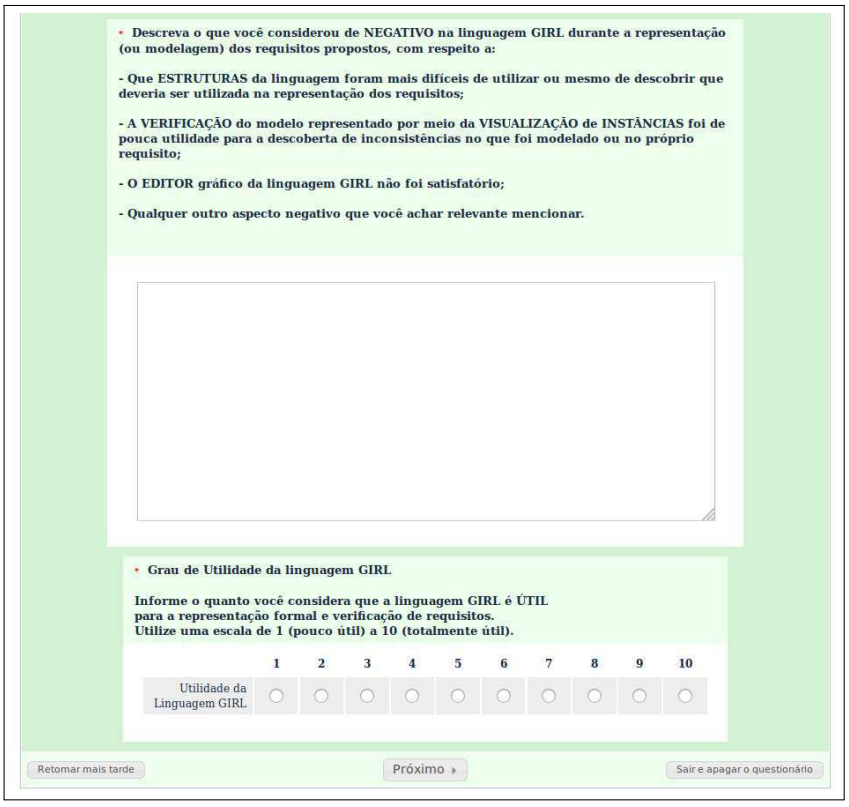

Figura C.17: Questionário - Avaliação da Linguagem GIRL - Questão 2 e 3

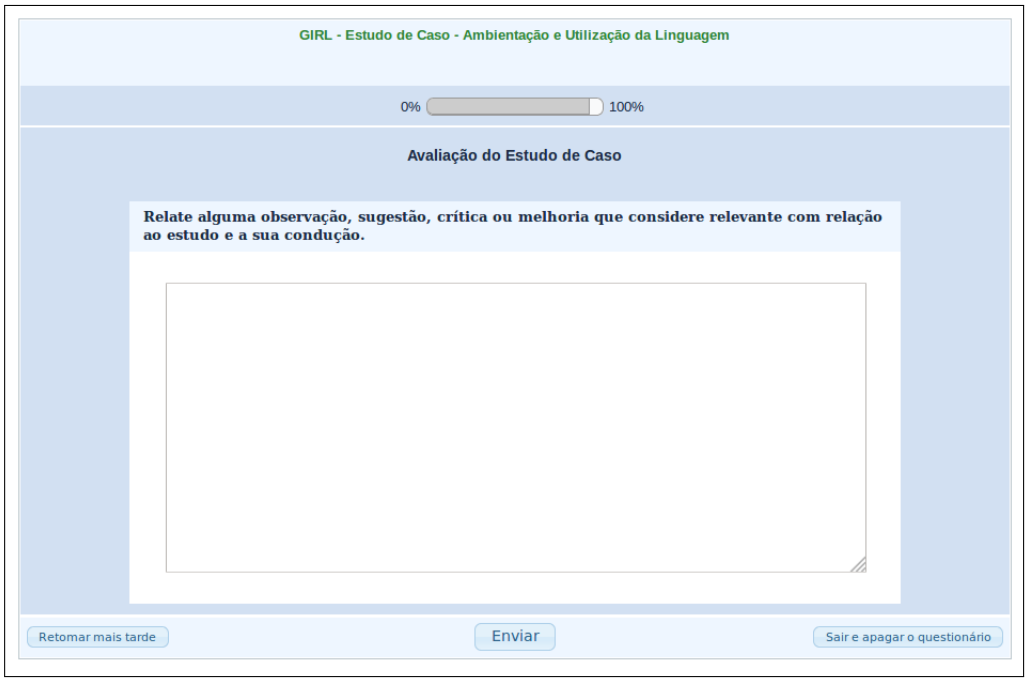

Figura C.18: Questionário - Avaliação do Estudo

Apêndice D

Dados Coletados

Tabela D.1: Caracterização dos Participantes

* Participantes *				
Participante	**Data de envio**	**Experiência Engenharia Requisitos**	**Conhecimento Lógica Matemática**	**Formação Acadêmica**
P00	24/07/2018 21:17	Nenhuma (OA)	Iniciante (01I)	PG-SS-Mestrado
P01	25/07/2018 11:00	Nenhuma (OA)	Avançado (03A)	Graduação (01F)
P02	23/07/2018 09:47	6 a 10 anos (6A10)	Intermediário (02I)	PG-LS (03F)
P03	23/07/2018 10:07	6 a 10 anos (6A10)	Avançado (03A)	PG-LS (03F)
P04	23/07/2018 08:13	6 a 10 anos (6A10)	Intermediário (02I)	PG-SS-Mestrado (05F)
P05	12/07/2018 10:57	6 a 10 anos (6A10)	Intermediário (02I)	PG-SS-Mestrado (05F)
P06	17/07/2018 15:39	3 a 5 anos (3A5)	Intermediário (02I)	PG-SS-Doutorado-incompleta (06F)
P07	23/07/2018 10:31	6 a 10 anos (6A10)	Intermediário (02I)	PG-LS-incompleta
P08	17/07/2018 14:49	1 a 2 anos (1A2)	Avançado (03A)	PG-SS-Mestrado (05F)
P09	17/07/2018 14:48	6 a 10 anos (6A10)	Intermediário (02I)	PG-SS-Mestrado (05F)
P10	27/07/2018 09:34	6 a 10 anos (6A10)	Intermediário (02I)	PG-SS-Doutorado-incompleta (06F)
P11		Nenhuma (OA)	Avançado (03A)	PG-SS-Doutorado (07F)

Tabela D.2: Graus de Entendimento Coletados na Ambientação

* Graus de Entendimento *											
GE-GrauEntendimento	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
GE Invariant	10	9	8	5	9	10	9	10	7	7	9
GE Entity	10	9	10	10	10	10	10	10	10	8	9
GE Entity Abstract	9	9	10	10	10	10	10	10	10	8	9
GE Entity Singleton	8	9	10	10	10	10	10	10	10	8	9

Tabela D.2: (continuação)

* Graus de Entendimento *											
GE-GrauEntendimento	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
GE Containment	9	9	8	10	10	8	10	10	10	9	9
GE Set Operation	10	9	9	1	10	7	10	8	10	8	10
GE Cardinality	10	9	10	10	10	10	10	8	10	9	10
GE Integer	10	9	10	10	10	10	10	8	10	9	10
GE Relational Operation	10	9	10	10	10	10	10	8	10	9	10
GE Relationship	10	9	10	10	10	10	10	9	10	8	10
GE Verificação	10	9	9	10	10	10	10	9	10	9	10
GE Logical Operation	10	9	10	10	10	10	8	9	10	9	10
GE Quantification	7	9	10	10	10	9	10	8	10	7	10
GE Implication	9	9	10	8	10	10	10	7	10	8	10
GU GIRL	9	9	10	8	10	9	9	9	10	9	9

Tabela D.3: Tempos Coletados na Ambientação e Utilização
('TP'= Tempo de Permanência em minutos,
'TU'= Tempo de Utilização em minutos,
Sinal Decimal = ',')

* Tempos - Ambientação e Utilização *											
Tempos	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
Ttotal	159,49	127,03	100,21	118,85	128,09	121,67	144,18	109,85	170,08	121,70	130,59
TPParticipante	0,32	0,23	0,22	0,31	0,31	0,15	0,17	0,28	0,17	0,30	0,29
TPModulo1	21,68	14,36	10,35	13,03	14,90	16,20	13,19	16,22	19,93	26,44	13,40
TPModulo2	9,58	4,93	5,58	8,21	6,55	10,16	7,26	10,21	8,45	9,72	6,68
TPModulo3	17,42	11,86	14,24	15,99	15,15	18,47	18,06	17,04	19,33	15,91	15,83
TPModulo4	4,82	2,80	4,92	3,83	4,91	4,70	7,90	6,01	6,42	4,83	4,45
TPModulo5	6,93	3,27	5,54	3,18	3,44	4,17	5,91	3,48	7,23	2,64	3,24
TPModulo6	10,46	5,69	7,70	8,19	8,16	9,01	8,63	8,24	16,04	6,50	8,18
TURequisitos11	11,39	7,36	10,68	11,28	7,50	17,65	9,38	9,07	18,12	5,68	14,14
TURequisitos2	11,10	5,61	5,71	12,62	14,07	4,60	13,78	6,00	6,58	7,90	12,78
TURequisitos3	11,82	8,51	7,19	12,91	13,73	8,65	13,18	4,95	11,11	3,14	20,56
TURequisitos4	14,88	47,01	10,63	20,28	22,38	21,51	27,71	19,21	35,72	20,17	22,28
TASolucao	12,45	1,99	10,13	4,54	6,30	3,51	9,17	3,94	9,19	4,87	4,42
TAGIRL	23,69	12,63	4,96	4,36	7,53	2,31	8,24	4,20	7,02	8,29	4,05
TAEstudo	2,97	0,77	2,37	0,11	3,18	0,59	1,60	1,00	4,76	5,32	0,28

Tabela D.4: Quantidade de Achados, Dúvidas e Pontos Negativos por Tipo Coletados na Ambientação e Utilização

* Dúvidas e Achados *					
Sigla	**Decrição**	**Achados**	**Ambientação**	**Utilização**	**Negativos**
AA	Achados – Outros				
Abstr	Entity Abstract - Achado	1			
AbstrGraf	Entity Abstract - Representação gráfica - Achado	3			
Alloy	Alloy		4	2	
Aplicação	Aplicação	2			

Tabela D.4: (continuação)

*	Dúvida e Achados				*
Sigla	**Decrição**	**Achados**	**Ambientação**	**Utilização**	**Negativos**
Cardinal	Cardinality		1	5	
Contain	Containment		6	1	
ContaXExten	Containment x Extends - Achado	4			
DifConj	Set Operation - Complement	1			
DifQtfConcl	Implication - Diferença entre Quantification na conclusão - Achado	1			
Editor	Editor		13	5	1
EditorID	Editor - Id-entificador das Estruturas	1			
Entity	Entity			2	
EntityAbs	Entity - Abstract		5	4	
EntityExtend	Entity - Extends			2	
EntityInvar	Entity - na Invariante		1	4	
EntityRelsh	Entity - nas Relashionships	1			
EntitySin	Entity - Singleton		2	6	
Enum	Enumeration - Associação	1		3	
Girl	Sobre a linguagem GIRL		2	1	
GirlBD	Girl e Banco de Dados - Associação		1	1	
GirlBom	Girl é bom - Achado	5			
GirlJava	Girl e Java - Associação	1			
ImplCopy	Implication - Cópias			1	
Implicat	Implicattion		4	2	2
ImplQtf	Implication - Quantification - uso		1		
Invariant	Invariant	1	8	7	
Logical	Logical Operation		6		
Metodol	Metodologia		2		
MultExpr	Multiplicity - Relational Expression			1	
MultInterf	Multiplicity - Interferência	1			
MultInterv	Multiplicity - Intervalo sem iniciar em zero		2	1	
Multiplici	Multiplicity		6	3	
MultpInvari	Multiplicity - na Invariant			1	
MultSome	Multiplicity - Semelhança expressão e some	2			
MultXCard	Multiplicity ou Cardinality	1		4	
NomeRelsh	Relationship - importância do nome	1			
Organiz	Organização		1	7	
QtfALLPremis	Implication - Quantification na premissa	1			
QtfComMult	Quantification com Multiplicity		2	4	
Quantif	Quantification	1		2	
Relational	Relational Operation				

Tabela D.4: (continuação)

* Dúvida e Achados *					
Sigla	**Decrição**	**Achados**	**Ambientação**	**Utilização**	**Negativos**
RelshDirecao	0 Relationship - Direção da relação		3		
RelshInva	Relationship - na Invariant			1	
Relship	Relationship		1	5	
RelshNome	Relationship - importância do nome			1	
RelshQtf	Relatinship - Quantification		1		
RelshXConta	Relationship x Containment			1	1
Requisito	Requisito	1	3	14	
SetOper	Set Operation		7		
SimCompl	Set Operation - Complement - Detectou incoerência no exemplo do módulo	1			
SimQtf	Quantification - Detectou a similaridade com Multiplicity	8			
SobrecConstr	Construtos - Negativo				1
VerifEnum	Verificador - identificou entidades Singletons - Enumerations	1			
VerifiBom	Verificador - Achou bom	6			
Verificador	Verificador			1	1

Tabela D.5: Quantidade de Dúvidas Coletadas na Ambientação e Utilização

* Graus de Entendimento *											
Dúvida	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
DA - Ambientação	10	2	5	9	3	10	3	14	15	10	1
DU - Utilização	11	3	12	4	12	10	8	4	11	9	8

Tabela D.6: Grau de Utilidade Registrado após a Utilização

* Graus de Utilidade *											
	P01	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
Grau de Utilidade	9	9	10	8	10	9	9	9	10	9	9

Tabela D.7: Grau de Entendimento *versus* Dúvidas na Ambientação / Utilização

* Grau de Entendimento *versus* Dúvidas na Ambientação / Utilização *			
Estrutura	**Entendimento Geral**	**Dúvidas Ambientação Utilização**	**Justificativa**

Tabela D.7: (continuação)

* Grau de Entendimento *versus* Dúvidas na Ambientação / Utilização *			
Estrutura	**Entendimento Geral**	**Dúvidas Ambientação Utilização**	**Justificativa**
Invariant (fig. D.1)	Baixo	8,5-A / 9-U	Possivelmente por falta de uma explicação mais detalhada no vídeo do módulo 1. Surgiram oito dúvidas sobre o conceito de invariante na ambientação, e persistiram sete dúvidas de conceito na utilização. No entanto, na avaliação da linguagem ela não figurou como difícil de utilizar.
Entity (fig. D.2)	Superior	0,5-A / 3-U	Oito atribuíram grau 10, um 9 e um 8. Na avaliação foi considerada fácil de utilizar.
Entity Abstract (fig. D.2)	Alto	5-A / 5-U	70% grau 10, dois 9 e um 8. A maioria das dúvidas da ambientação recaiu na similaridade com o conceito de classe abstrata em programação. Na utilização envolveu o uso da entidade abstrata no lugar da entidade pai em uma relação.
Entity Singleton (fig. D.2)	Alto	2-A / 7-U	70% grau 10, um 9 e dois 8. As dúvidas envolveram a aplicação de *enumeration*. O participante P04 resumiu bem: "*Se for aluno qualquer não é singleton, se for tipo, é singleton.*"
Containmet (fig. D.3)	Mediano	6-A / 2-U	Todos atribuíram Grau de Entendimento igual ou maior que 8, sendo cinco no grau 10. Apenas um participante utilizou o containment para expressar a invariante: *Um Aluno deve ter uma Área...* associando o 'ter' da expressão ao sentido de 'conter' do containment. Uma outra causa das dúvida pode estar no exemplo usado na ambientação que trouxe dúvidas de negócio - *O Banco Central está contido em Banco*.
Set Operation (fig. D.3)	Mediano	6-A /	Um atribuiu grau 1, para pouco entendimento e os demais, maior ou igual a 7. Na ambientação foi abordado apenas a operação Complement e com o exemplo de uma invariante que gerou duas dúvidas de requisito: Banco Central contido em Banco.
Logical Operation (fig. D.3)	Mediano+	6-A /	Melhor que o Containment. No entanto, levantou dúvidas por causa do exemplo da ambientação, que apresentava uma redundância, como mencionou P06: *Não fica claro na representação a logica do AND. No caso o entendimento para o programador seria um a verificação se as classes conta_corrente e conta_poupanca são heranças de conta.*

Tabela D.7: (continuação)

* Grau de Entendimento *versus* Dúvidas na Ambientação / Utilização *			
Estrutura	**Entendimento Geral**	**Dúvidas Ambientação Utilização**	**Justificativa**
Relationship (fig. D.5)	Alto	13-A / 16-U	Maior ou igual a 8, sendo 70% em 10. A maioria das dúvidas refere-se à multiplicidade. A direção da relação também foi questionada em virtude da multiplicidade da esquerda. E também a possibilidade da multiplicidade com expressão relacional expressar os dois limites do intervalo (ex.: ">=1" e "<=3"), pois, em GIRL, só é permitido referenciar um limite ("<=2", onde o limite inferior seria zero).
Verificação (fig. D.5)	Alto	4-A / 2-U	40% em 9 e 60% em 10. Uma dúvida da utilização foi a possibilidade de verificar apenas parte do modelo para evitar instâncias de entidades irrelevantes ao que se desejava .
Cardinality (fig. D.4)	Alto	1-A / 6-U	Apesar de ter apresentado alto entendimento na ambientação, na utilização ocorreram dúvidas sobre a representação da quantidade na relação utilizando a cardinalidade. Uma possível causa é a similaridade dos conceitos. Um participante tentou copiar a relação (seta) dentro da cardinalidade.
Integer (fig. D.4)	Alto		Estrutura simples que não foi necessária na utilização. Alguns até cogitaram utilizá-la na operação relacional, em vez de usar a multiplicidade.
Relational Operation (fig. D.4)	Alto		Semelhante a integer, sem utilização.
Implication (fig. D.6)	Mediano	4-A / 3-U	Estrutura considerada uma das mais complexas, apesar de ter registro de poucas dúvidas. A seção de avaliação da linguagem contém maiores detalhes.
Quantification (fig. D.6)	Mediano	2-A / 4-U	Como em Logical Operation, o exemplo da ambientação podia ser expresso por meio de multiplicidade. Apesar disso, o exemplo do módulo da implicação envolveu o uso de quantificação.

Tabela D.8: Avaliação das Estruturas de GIRL - Respostas dos Participantes

* Que ESTRUTURAS da linguagem foram mais fáceis/difíceis de utilizar ou de descobrir quando utilizar. *		
Part.	**Mais Fáceis**	**Mais Difíceis**
P01	1) Estruturas mais fáceis: **invariante, entidade** (abstrata ou não), **relacionamento**.	Os **quantificadores e a cardinalidade** são os mais difíceis de saber quando utilizar.
P02	MAIS FACEIS: **Invariante, entidade, singleton, abstract, containment**	MAIS DIFICEIS: **Quantification, cardinalidade, implication**

Tabela D.8: (continuação)

*	Que ESTRUTURAS da linguagem foram mais fáceis de utilizar ou de descobrir quando utilizar.	*
Part.	**Mais Fáceis**	**Mais Difíceis**
P03	As **entidades e relacionamentos** foram fáceis de usar	A **implicação** foi difícil de usar
P04	Estruturas: **Entity, Singleton, Abstract, Relationship**.	A limitação foi por conta da usuária.
P05	Achei **bastante intuitiva as estruturas**, tanto sua representação do conceito das figuras, como o uso.	Uma dificuldade foi lembrar sobre o uso de **quantificar na relação dentro da implicação**.
P06	A parte de **invariantes, relacionamentos e entidades** foi bastante direta para uso.	O uso de **implicações e de quantificadores** foi mais complexo.
P07	A mais fácil e intuitiva até por ser utilizado em outros tipos de modelagem é a de **entidade e relacionamento**.	**Implicação não é intuitiva**, pois até para inserir os dados é necessário se atentar para a **ordem em que é feita**. Talvez fosse interessante pensar numa expansão da ferramenta para **geração de um código fonte** base, já que **demanda um tempo para especificar tudo** e esse tempo poderia ser reaproveitado nas etapas posteriores do projeto..
P08	As Estruturas **Entity, Relacionamento, Implicação** foram mais fáceis de compreender.	Achei difícil diferenciar o **contém (Containment) do possui (Relacionamento)**.
P09	Quanto as estruturas, destaco a facilidade de aprendizado e também a forma fácil de se modelar os requisitos. Também destaco a **completude dos componentes e da linguagem**, que possibilitam uma modelagem complexa, mas fácil e clara de se entender.	
P10	O **poder de expressividade dos construtos da linguagem**.	**Sobrecarga de alguns construtos** gerando dúvida de uso (e.g. no caso da **cardinalidade**). - Possibilidade de emprego de alguns **termos** cunhados (e.g. termos de banco de dados).
P11	(escrito ficha) "Maneira como relaciona vários entes. Mais fáceis: **entidades e relacionamento**"	"Mais difíceis: **implicação**. Não é tão difícil, mas, com relacionamentos, não é tão simples de enxergar."

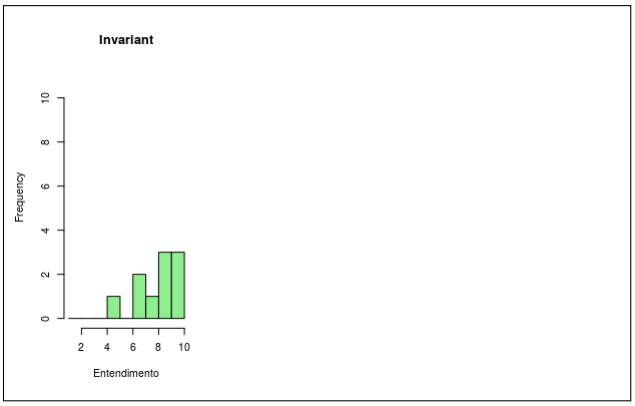

Figura D.1: Invariant - Grau de Entendimento

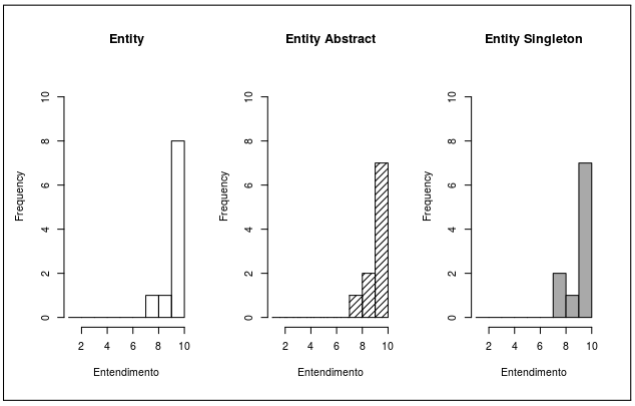

Figura D.2: Entity - Grau de Entendimento

Bom

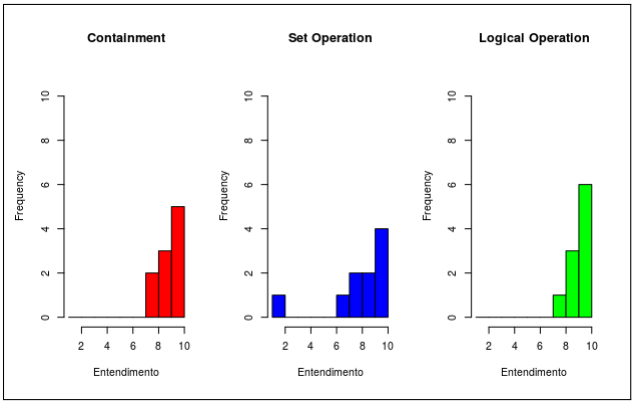

Figura D.3: Containment, Set Operation e Logical Operation - Grau de Entendimento

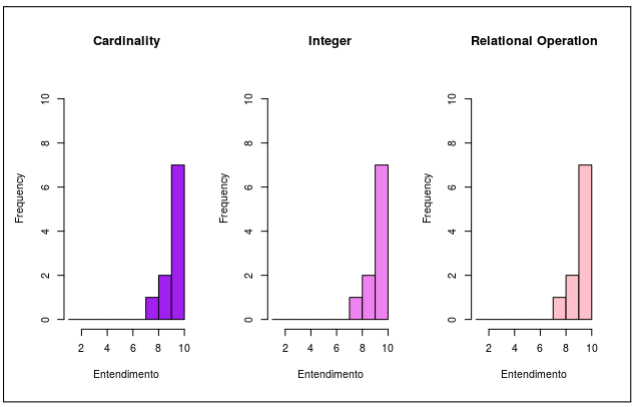

Figura D.4: Cardinality, Integer e Relational Operation - Grau de Entendimento

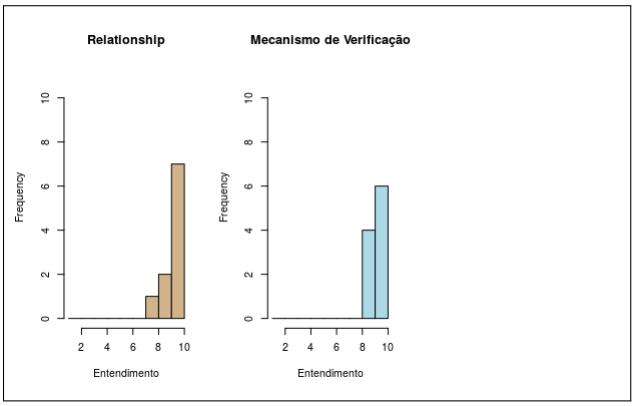

Figura D.5: Relationship, Mecanismo de Verificação - Grau de Entendimento

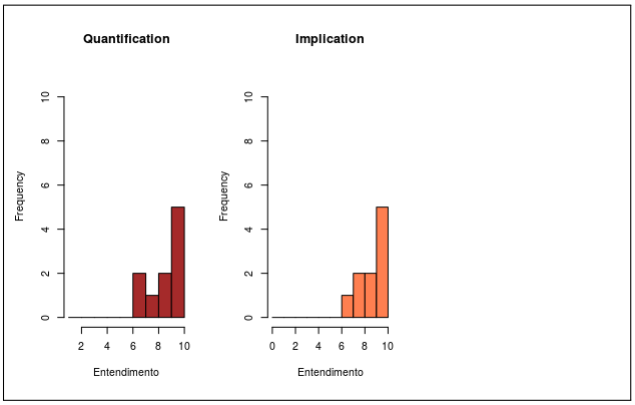

Figura D.6: Quantification e Implication - Grau de Entendimento

Tabela D.9: Avaliação da Verificação em GIRL - Respostas dos Participantes

*	VERIFICAÇÃO por meio da VISUALIZAÇÃO de INSTÂNCIAS foi de Útil/Não Útil	*
Part.	**Útil**	**Não Útil**
P01	2) Sim. **Extremamente útil**. Especialmente executando de forma **incremental**.	
P02	A VERIFICAÇÃO do modelo representado por meio da VISUALIZAÇÃO de INSTÂNCIAS foi útil para a descoberta de inconsistências no que foi modelado ou no próprio requisito: **SIM, o melhor de todos, ótimo para descobrir inconsistências já na modelagem**.	
P03	A verificação foi de **suma importância**	
P04	Foi, com ressalvas	Ressalva: **não funcionou em todas as vezes**. Talvez se tivesse sido exposto as **limitações**, como não usar nomes repetidos nas relações, teria sido melhor aproveitado.
P05	**Gostei muito do analisador** para verificar o que havia implementado. Na visão de especificador é importante para **mostrar essa visão até mesmo ao cliente** por sua representação simples.	
P06	A visualização de instâncias foi útil para validar se os requisitos foram modelados corretamente.	
P07	Foi importante sim, pois serve, para o programador, como um **forma de debugar o que foi especificado**, prevenindo até que passe algum erro que não seria pego caso o analista tivesse somente o modelo.	
P08	A **Verificação foi útil** para avaliar o modelo proposto.	
P09	A **verificação foi bastante útil e precisa**, embora ache necessário a enumeração das mesmas, para facilitar a identificação e navegação entre elas. Destaco como aspecto positivo as soluções apresentadas após a modelagem, que permite i**identificar possíveis falhas na modelagem**.	Na verificação, achei necessário uma melhor **navegação entre as soluções apresentadas** e identificação das mesmas.
P10	A **verificação semiautomática** dos diversos cenários.	
P11		

Tabela D.10: Avaliação do Editor GIRL - Respostas dos Participantes

*	O EDITOR gráfico da linguagem GIRL foi satisfatório ou não	*
Part.	**Positivo**	**Negativo**
P01	Sim. Tem bons atalhos de teclado para as operações mais recorrentes, mas permite manipulação via mouse das estruturas pra quem não conhece os atalhos.	
P02	O EDITOR gráfico da linguagem GIRL foi satisfatório: SIM	

Tabela D.10: (continuação)

* O EDITOR gráfico da linguagem GIRL foi satisfatório ou não *		
Part.	**Positivo**	**Negativo**
P03	O editor foi **razoável**.	Sobre o editor, algumas estruturas podem ser mais fáceis de **desenhar**: - **Implicação**: selecionar primeiro a premissa, e jogar numa área do gráfico, e a conclusão noutra área.
P04	Foi satisfatório.	
P05	Em geral **gostei de usar** a ferramenta.	
P06	O editor gráfico funcionou de forma satisfatória	
P07	**A ferramenta é simples e com poucas funcionalidades**, isso facilita para o usuário iniciante.	**Implicação não é intuitiva**, pois até para inserir os dados é necessário se atentar para a **ordem em que é feita**
P08	O editor ajuda bastante.	Achei difícil de **redimensionar** alguns componentes.
P09	O editor é simples, fluido e funcional, agradável e fácil de usar.	
P10	Usabilidade e intuitividade da ferramenta **Pouca necessidade de treinamento**	Uso de mais **tons/cores** para incrementar a expressividade dos construtos
P11		Dificuldade no manuseio.

Tabela D.11: Indicativos do Valor da Representação Gráfica

* Indicativos do Valor da Representação Gráfica *	
Part.	**Expressões/Observações que Indicam o Valor da Representação Gráfica**
P01	"Mesma quantificação da **matemática dos conjuntos**." Associação com os conceitos prévios da teoria matemática dos conjuntos.
P02	"Foi uma coisa visual, **não ficou cortado** (após o extends), ativou a ideia." Quando questionado sobre o que fez ver que a entidade era abstrata.
P02	"Parece Teoria dos Conjuntos: Invariante, o Universo e as **Bolinhas , os conjuntos**."
P02	"O nome diferente ajuda a **ver setas** na verificação."
P04	"**Escuro**, escuro". Referenciando a representação da multiplicidade ONE para as duas multiplicidades da relação Aluno tem Area.
P05	"**Vi** um que a relação que colocava na bolinha?" Lembrando que, no vídeo tinha visto a expressão relacional colocada na multiplicidade (bolinha).
P05	"É bem intuitiva, o **desenho** para o conceito."
P05	"Gostei muito do analisador para verificar o que havia implementado. Na visão de especificador é importante para **mostrar essa visão** até mesmo ao cliente por sua representação simples." Sobre a utilização das instâncias na comunicação com o cliente no desenvolvimento de software.
P06	"Uma área ... é cinza. **Cinza** é um, né?" Para representar a multiplicidade.
P07	"Direita só uma área!" Para colocar a multiplicidade 'apenas um' (ONE) na direita da relação. Apesar de falar assim, colocou a multiplicidade 'um ou mais' (SOME). Mais adiante **viu esse erro** na multiplicidade que deveria ser 'apenas um' e trocou a multiplicidade.
P07	"Não tá colocando mais de um Curso por Aluno. Tenho Curso que pode ou não ter Aluno.Tá correto!" Ao analisar o **gráfico** gerado com as relações das instâncias na verificação.
P07	"O mais legal foi poder criar instâncias e **ver o modelo** sendo aplicado."
P09	"Ele (Aluno) nem ficou particionado!" Percebendo que, no momento que outras entidades estendiam Aluno, não **apareceu o desenho das partições** em Aluno. Mas, deveria ser abstrata e particionar.)"
P09	"É bacana **ver** as possibilidades!" Na verificação.
P10	"Vai ganhar 3 fatias ... Obrigatoriamente tem que ser 3 tipos associados!" Onde fatias seriam o resultado do **desenho** das divisões da entidade abstrata quando outra entidade a estende."

Tabela D.11: (continuação)

*	Indicativos do Valor da Representação Gráfica	*
Part.	**Expressões/Observações que Indicam o Valor da Representação Gráfica**	
P11	Durante o vídeo do módulo 6, balançou a cabeça na construção da implicação enquanto **desenhava no ar uma linha diagonal** da esquerda-cima até a direita-baixo, indicando a próxima relação a colocar na implicação.	
P11	"Tá ok... são disjuntos!" Por causa da partição **exibida** da entidade abstrata.	
P11	"Ok... e são disjuntos!" Possível referência às partições **desenhadas** na entidade abstrata."	
P11	"Só pode cursar um Curso, então essa relação é única... existe uma unicidade aqui.. e que é aqui" Dizia isso, enquanto **apontava para a multiplicidade** de Curso.	

Tabela D.12: Avaliação do Estudo

*	Avaliação do Estudo	*
Part.	**Relate alguma observação, sugestão, crítica ou melhoria que considere relevante com relação ao estudo e a sua condução.**	
P01	Dependendo do sujeito, o **tempo** programado para a sessão pode extrapolar a previsão.	
P02	legal	
P03	Considero o experimento muito bem elaborado, o **material** foi esclarecedor e completo, e a **condução** extremamente cuidadosa e necessária.	
P04		
P05	Achei a condução interessante por ser ter a parte em **vídeo antes de cada módulo**. A divisão em **módulos** também é importante para melhor aprendizado. O **vídeo** foi bem fácil de entender, não achei cansativo. As perguntas do **questionário** foram diretas e sem ser cansativas também. Mesmo tendo **várias etapas** foi bem **motivante** usar a ferramenta.	
P06	Tentar **simplificar o uso de implicações** (esse seria necessário simplificar mais) **e de quantificadores.**	
P07	Sugerir que esse **curso** pudesse ser realizado virtualmente, por ser **demorado**.	
P08	A condução foi tranquila, as **dúvidas** foram sanadas e o **ambiente** estava de acordo com o esperado.	
P09	O estudo foi muito bem conduzido, com um **tutorial** simples, modulado, que possibilitou em um curto espaço de **tempo** o aprendizado da linguagem e ferramenta de modelagem. Todos os **artefatos** necessários estavam disponíveis, desde a **chamada para a participação** do experimento, com e-mail convidativo e formulário de cadastro para confirmação. As **ferramentas utilizadas para acompanhar** e registrar o experimento também foram ideais e bem escolhidas, possibilitando assim uma perfeita execução do experimento, sem contratempos. **Materiais ilustrativos** estavam expostos e a disposição para consulta, o que facilitou a execução da modelagem e uso da ferramenta.	
P10	- **Captura de audio, video e interação com o respondente**. - 'trainning session' antes da execução do experimento - **Amostra** de conveniência"	
P11		

Apêndice E

Modelos Elaborados por Conjunto de Requisitos

E.1 Conjunto de Requisitos 1

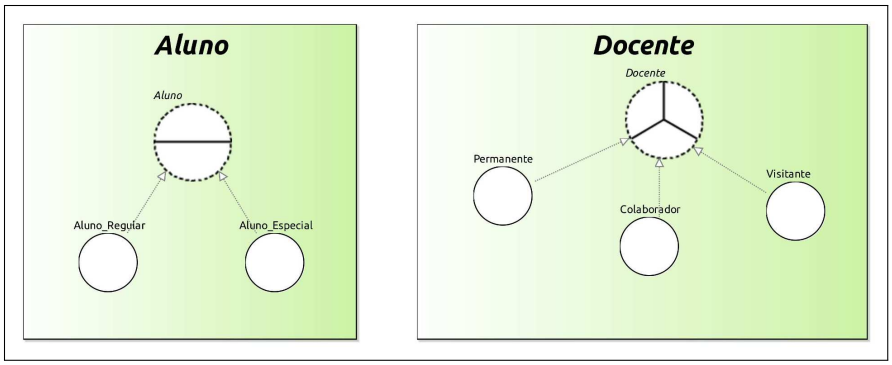

Figura E.1: Participante 01 – Modelo Conjunto 1

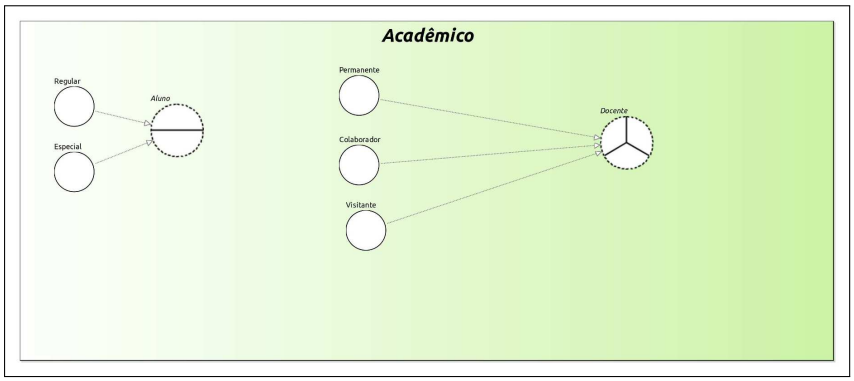

Figura E.2: Participante 02 – Modelo Conjunto 1

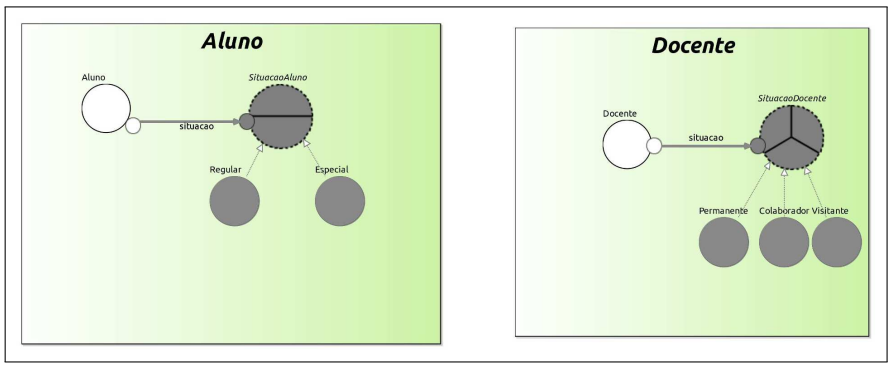

Figura E.3: Participante 03 – Modelo Conjunto 1

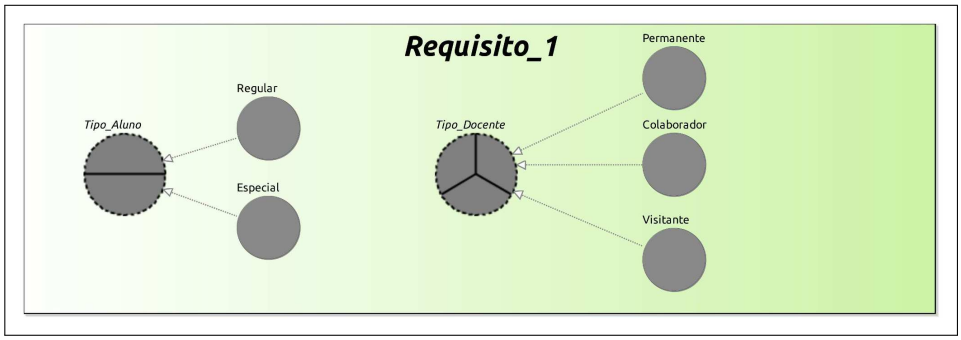

Figura E.4: Participante 04 – Modelo Conjunto 1

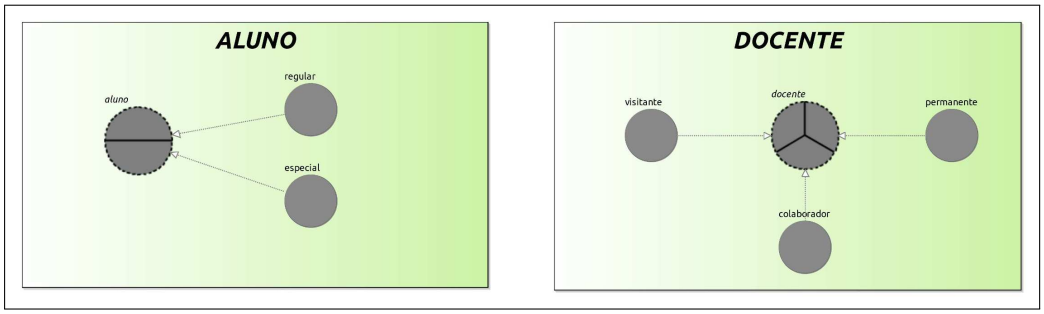

Figura E.5: Participante 05 – Modelo Conjunto 1

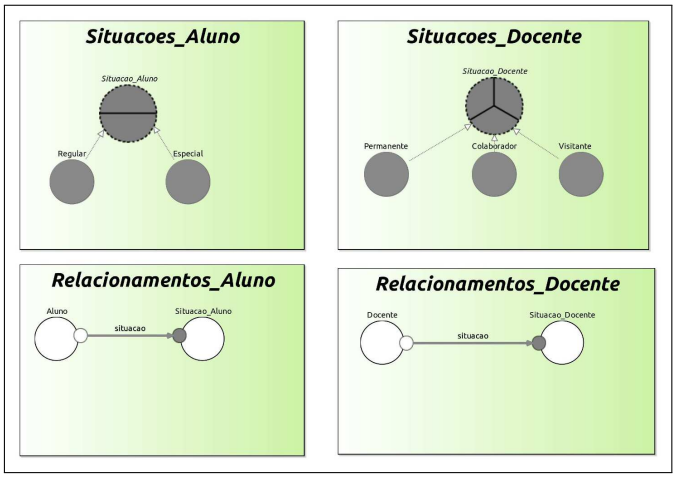

Figura E.6: Participante 06 – Modelo Conjunto 1

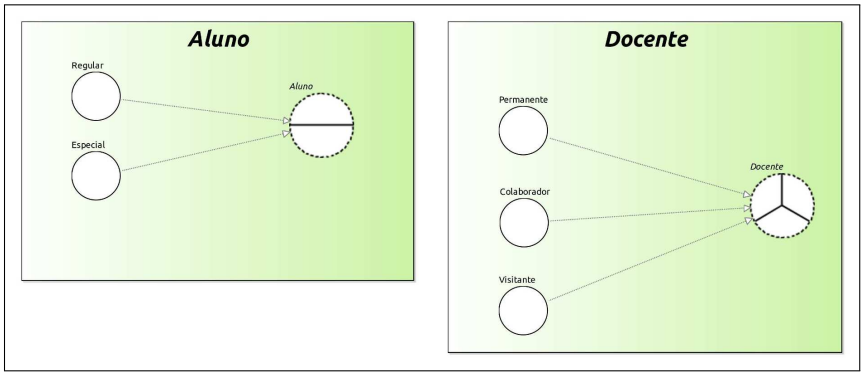

Figura E.7: Participante 07 – Modelo Conjunto 1

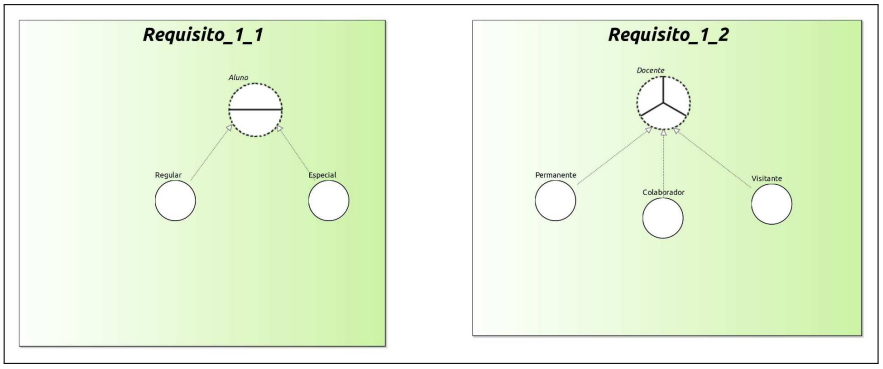

Figura E.8: Participante 08 – Modelo Conjunto 1

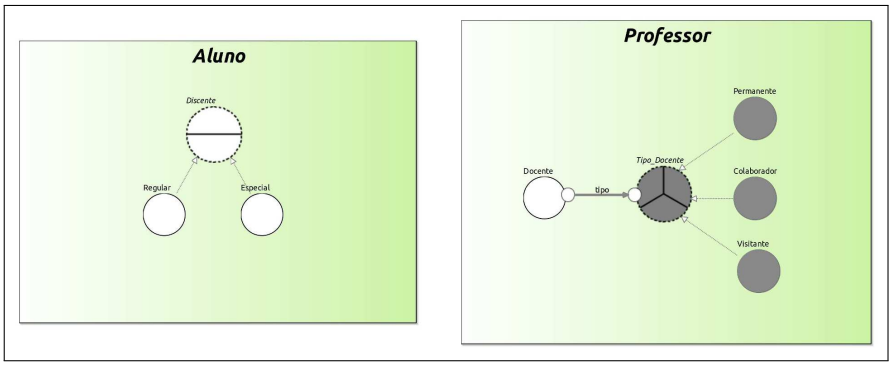

Figura E.9: Participante 09 – Modelo Conjunto 1

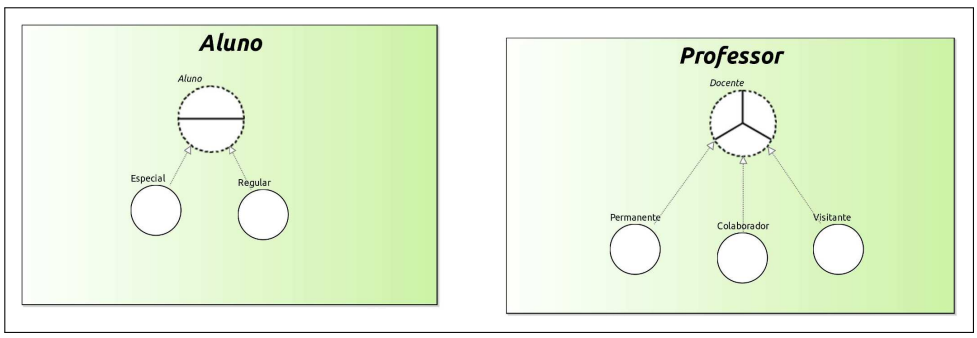

Figura E.10: Participante 10 – Modelo Conjunto 1

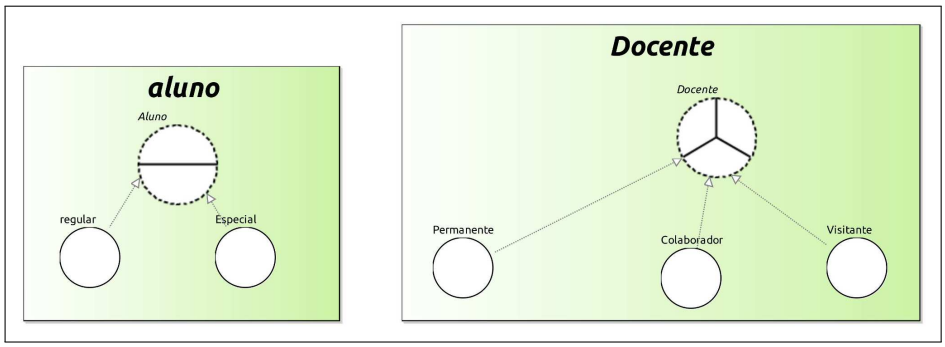

Figura E.11: Participante 11 – Modelo Conjunto 1

E.2 Conjunto de Requisitos 2

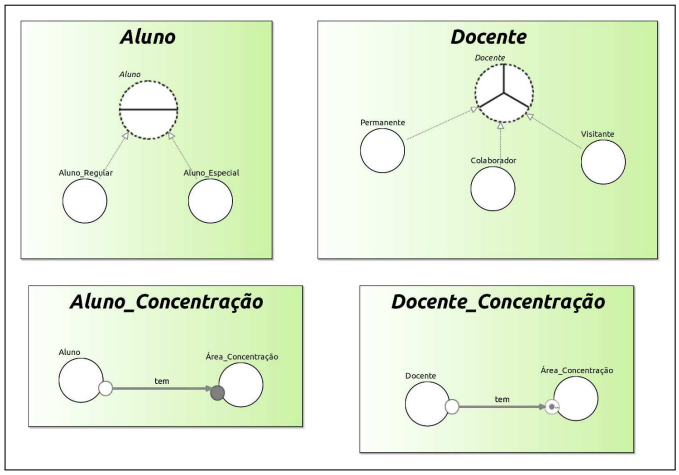

Figura E.12: Participante 01 – Modelo Conjunto 2

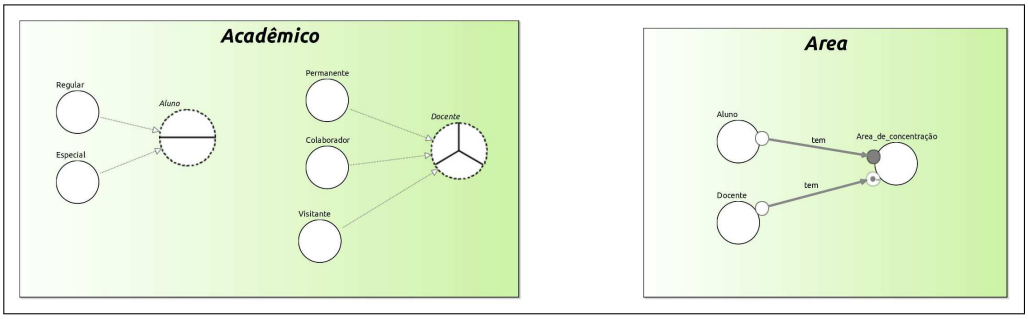

Figura E.13: Participante 02 – Modelo Conjunto 2

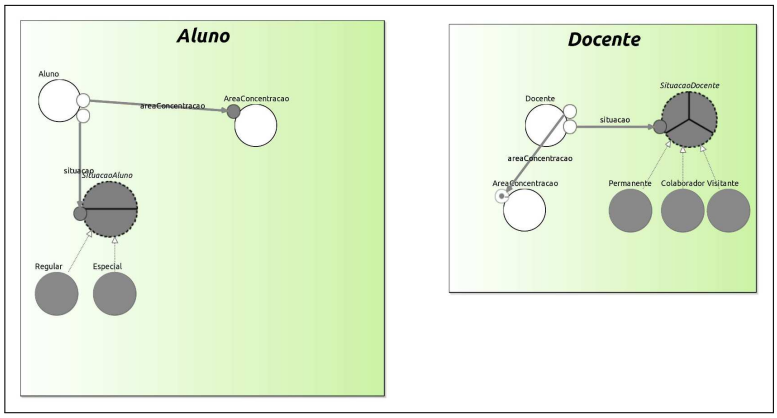

Figura E.14: Participante 03 – Modelo Conjunto 2

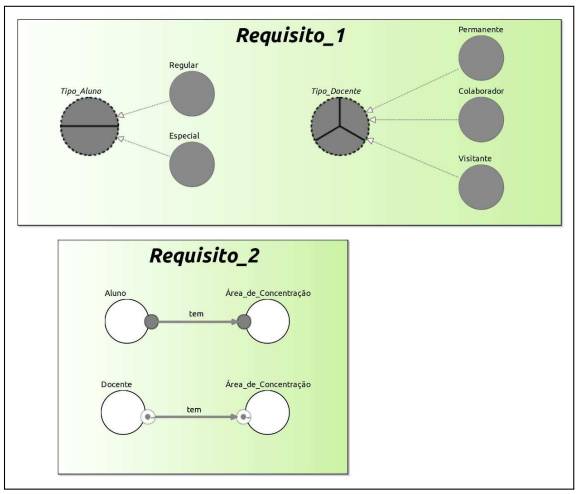

Figura E.15: Participante 04 – Modelo Conjunto 2

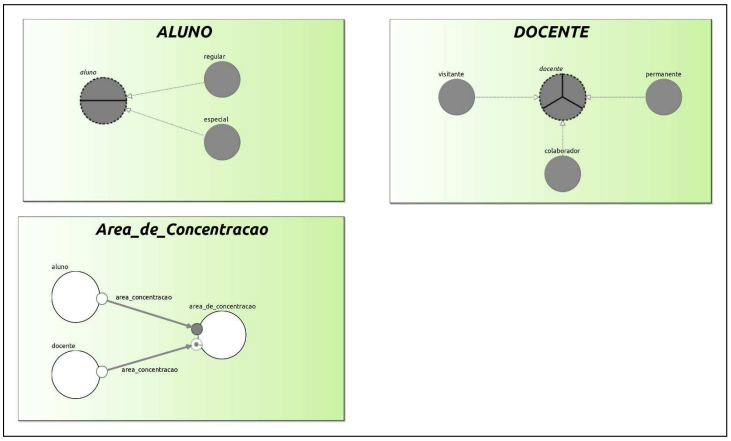

Figura E.16: Participante 05 – Modelo Conjunto 2

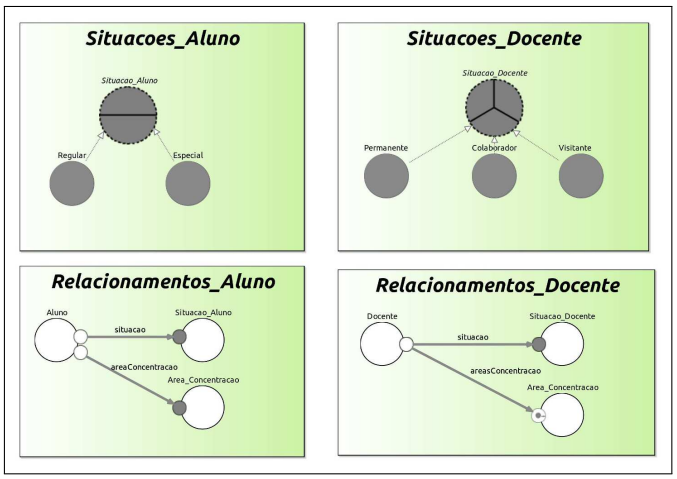

Figura E.17: Participante 06 – Modelo Conjunto 2

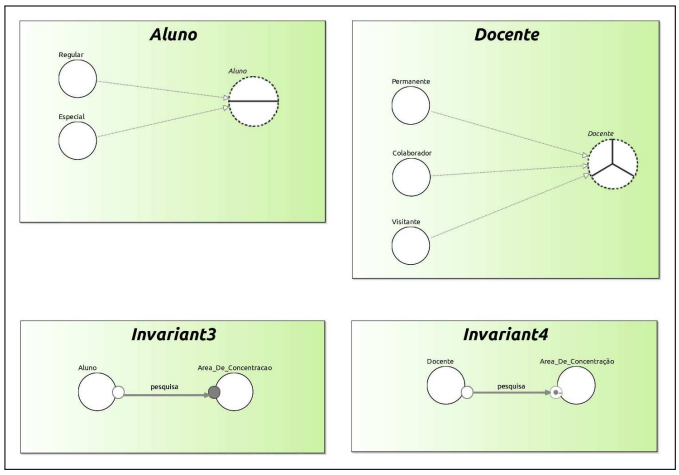

Figura E.18: Participante 07 – Modelo Conjunto 2

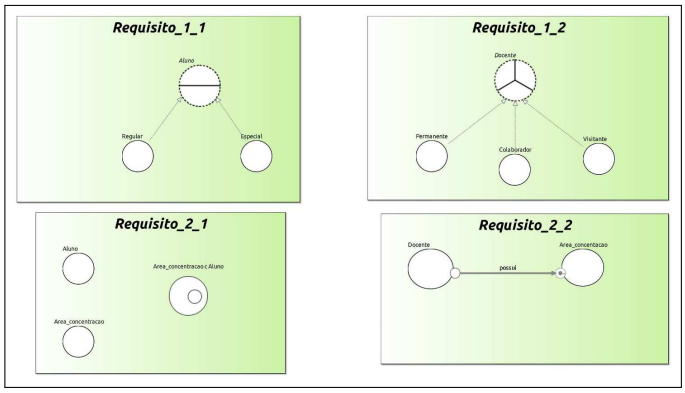

Figura E.19: Participante 08 – Modelo Conjunto 2

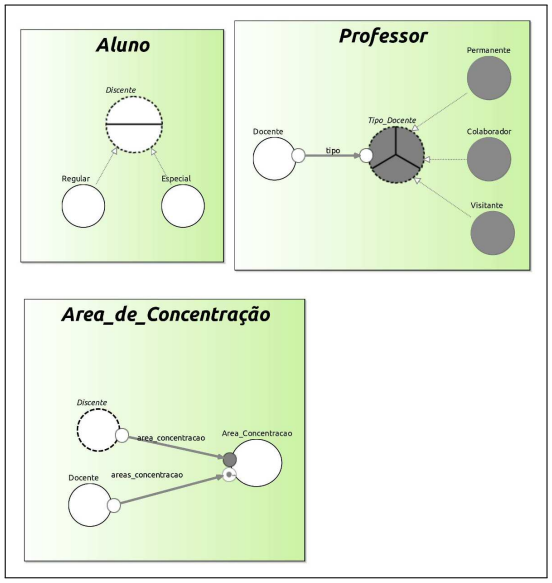

Figura E.20: Participante 09 – Modelo Conjunto 2

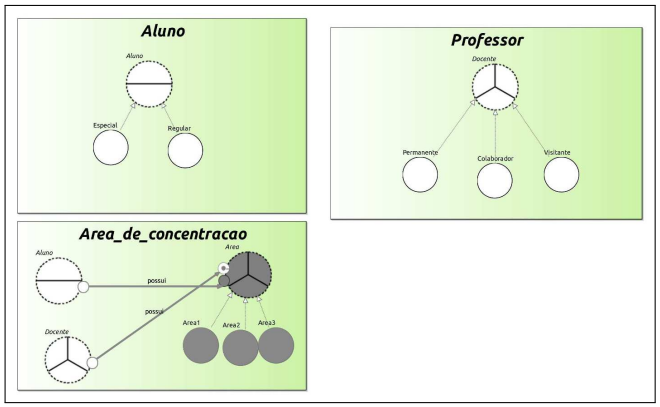

Figura E.21: Participante 10 – Modelo Conjunto 2

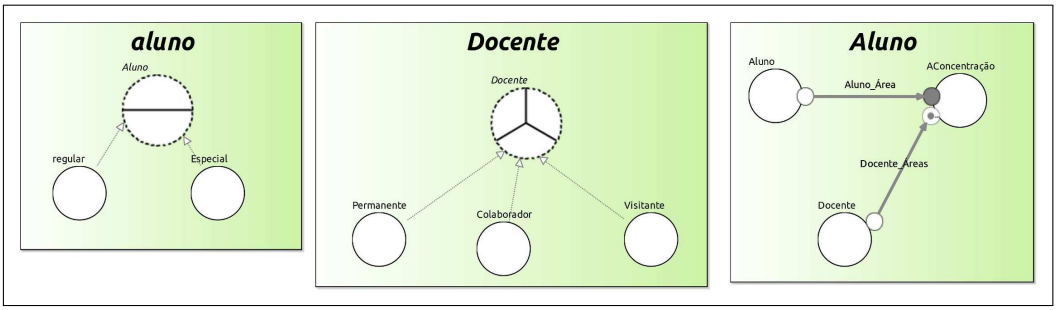

Figura E.22: Participante 11 – Modelo Conjunto 2

E.3 Conjunto de Requisitos 3

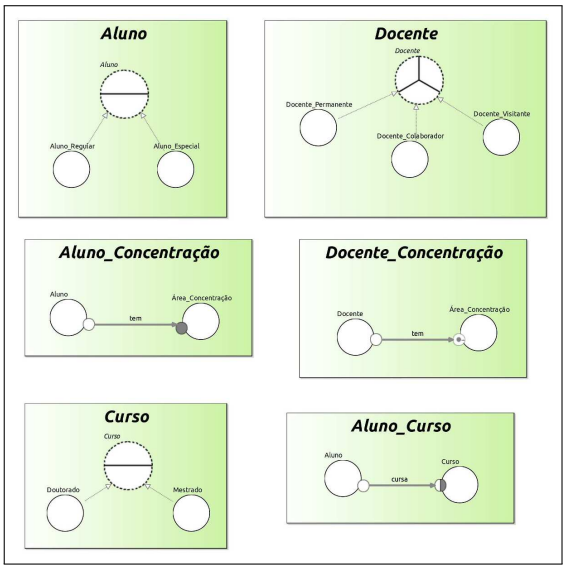

Figura E.23: Participante 01 – Modelo Conjunto 3

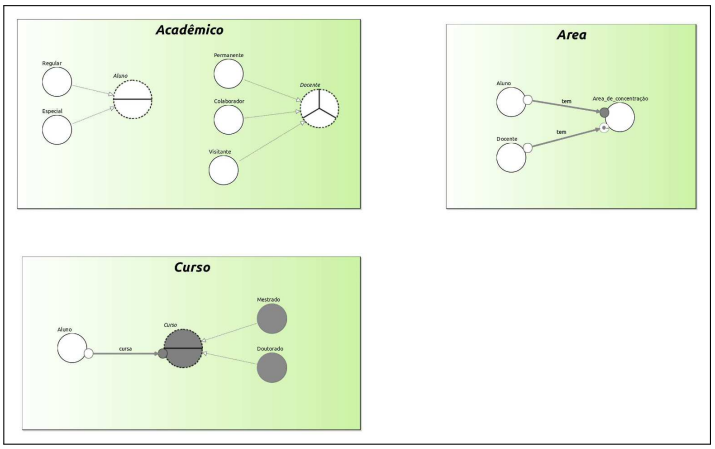

Figura E.24: Participante 02 – Modelo Conjunto 3

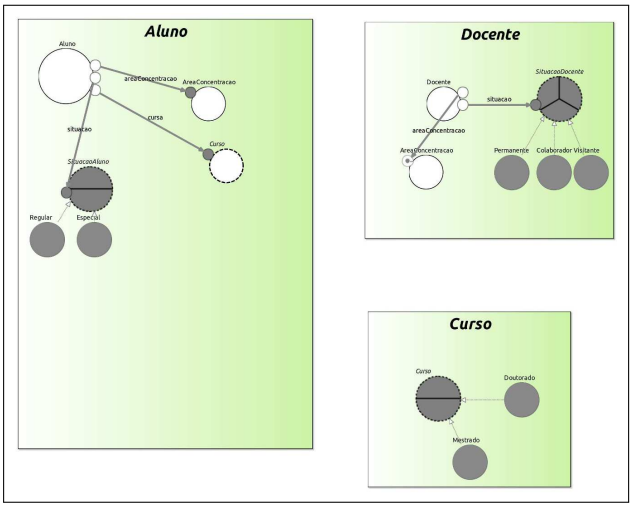

Figura E.25: Participante 03 – Modelo Conjunto 3

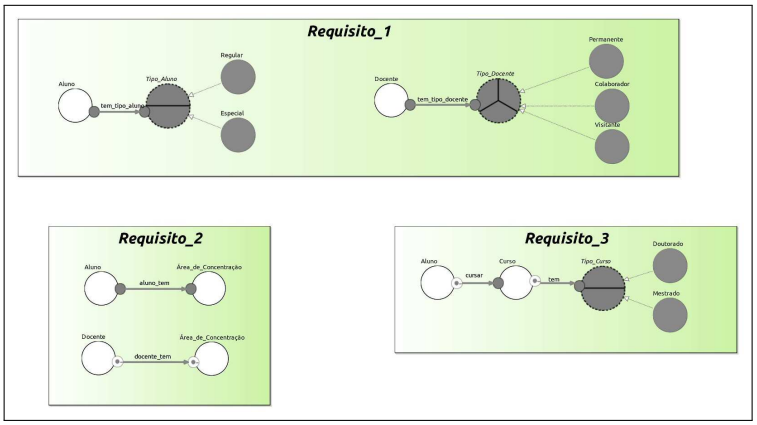

Figura E.26: Participante 04 – Modelo Conjunto 3

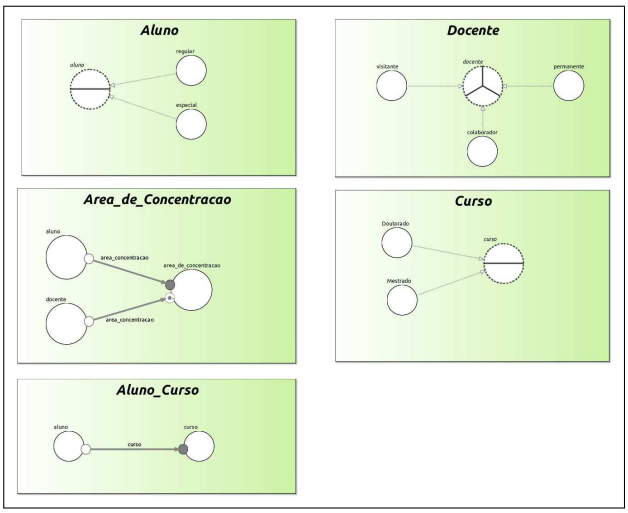

Figura E.27: Participante 05 – Modelo Conjunto 3

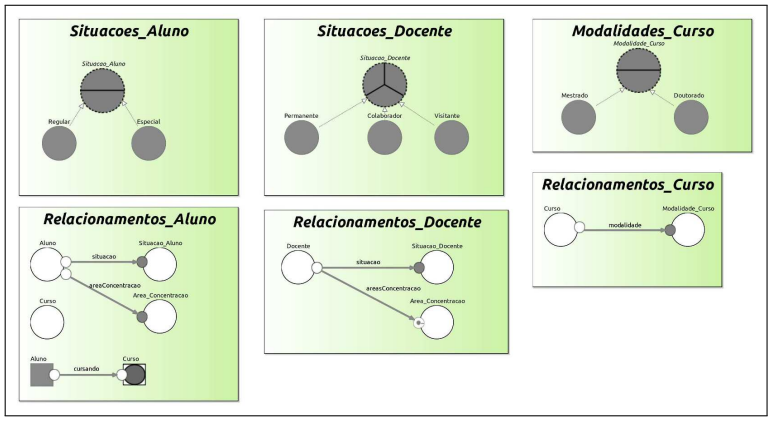

Figura E.28: Participante 06 – Modelo Conjunto 3

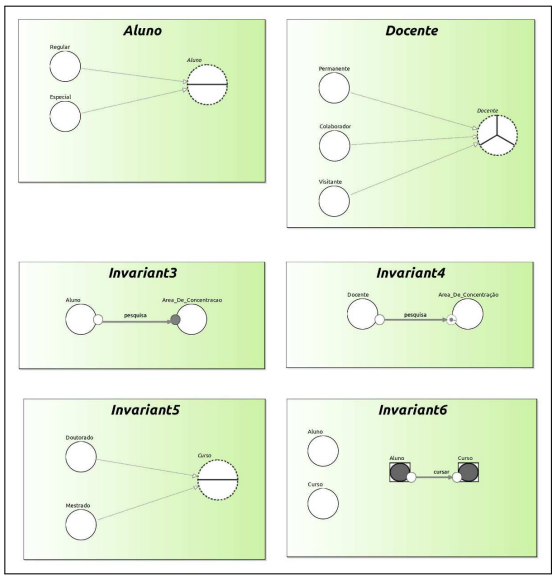

Figura E.29: Participante 07 – Modelo Conjunto 3

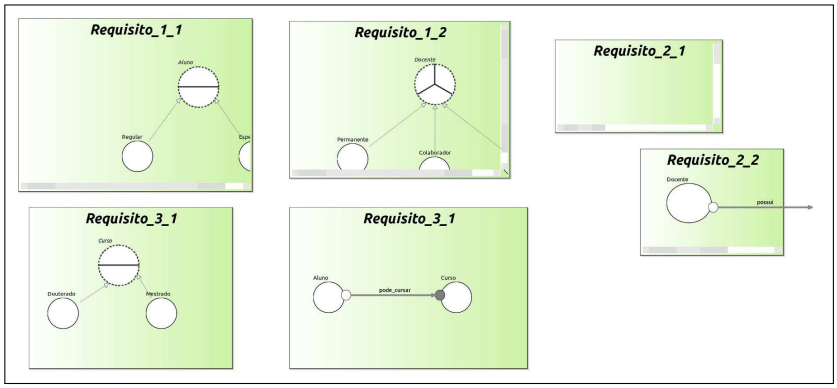

Figura E.30: Participante 08 – Modelo Conjunto 3

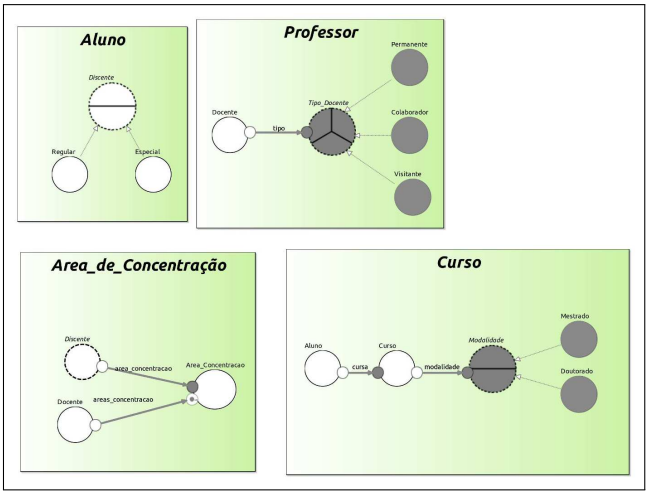

Figura E.31: Participante 09 – Modelo Conjunto 3

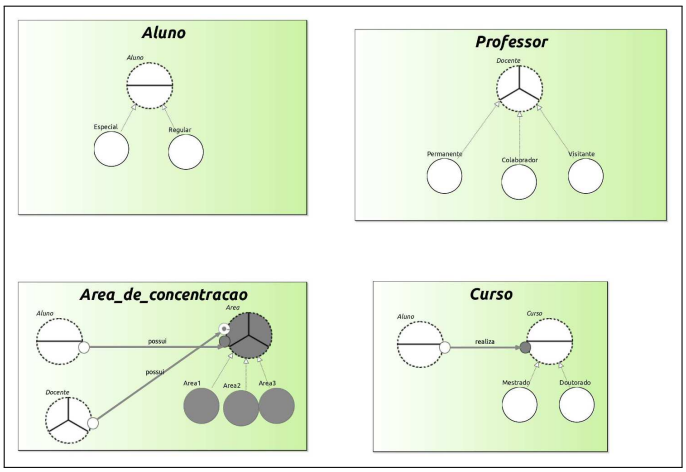

Figura E.32: Participante 10 – Modelo Conjunto 3

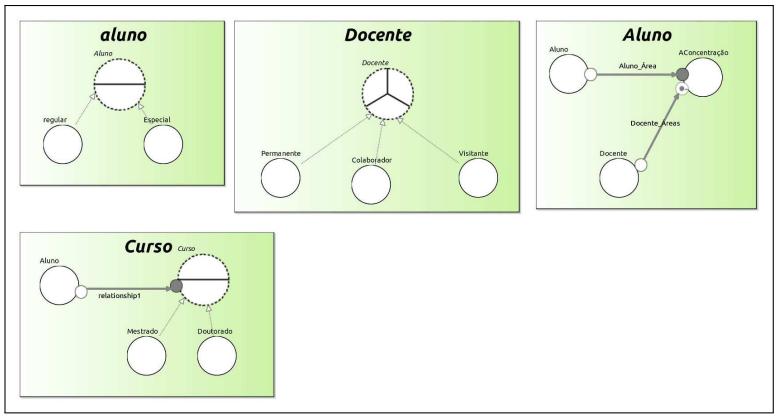

Figura E.33: Participante 11 – Modelo Conjunto 3

E.4 Conjunto de Requisitos 4

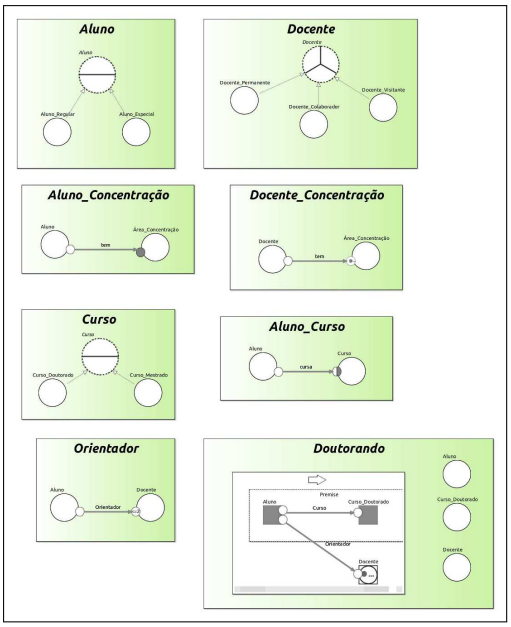

Figura E.34: Participante 01 – Modelo Conjunto 4

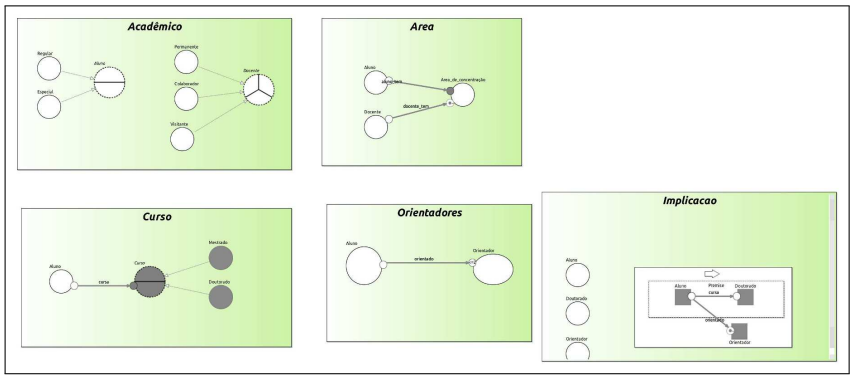

Figura E.35: Participante 02 – Modelo Conjunto 4

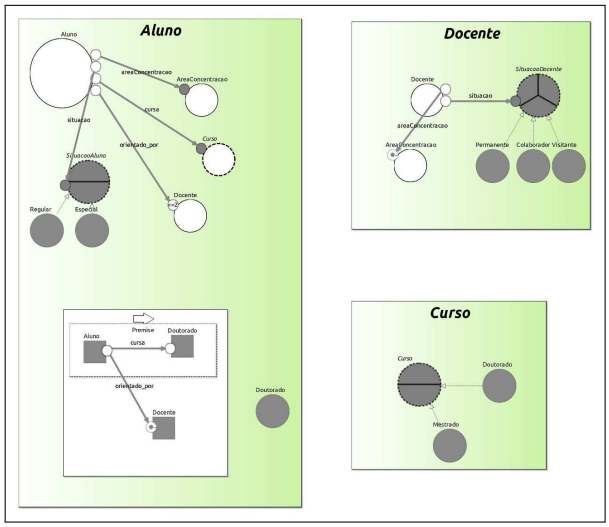

Figura E.36: Participante 03 – Modelo Conjunto 4

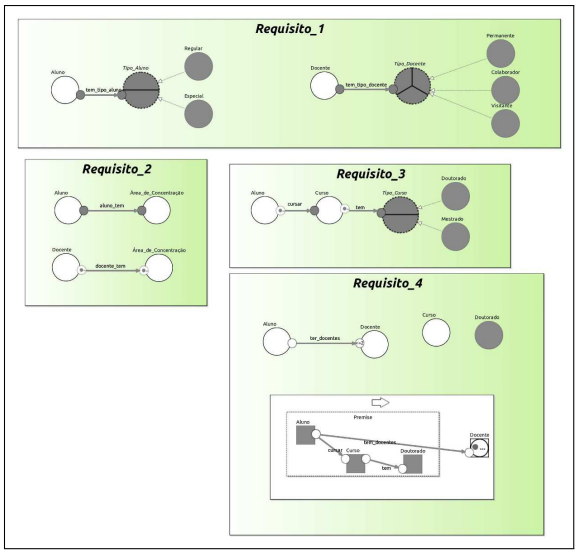

Figura E.37: Participante 04 – Modelo Conjunto 4

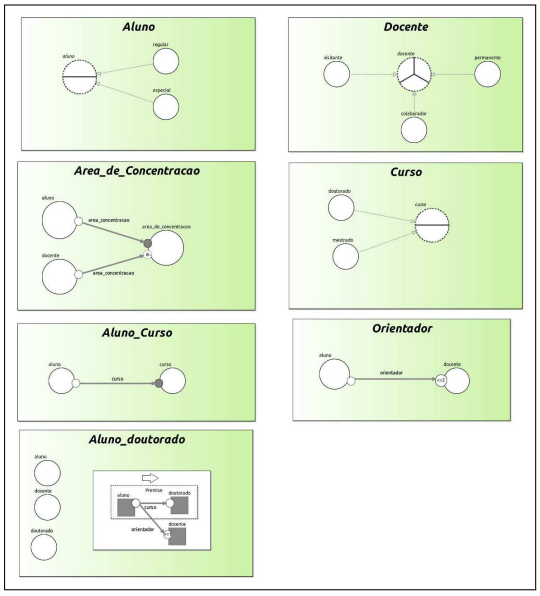

Figura E.38: Participante 05 – Modelo Conjunto 4

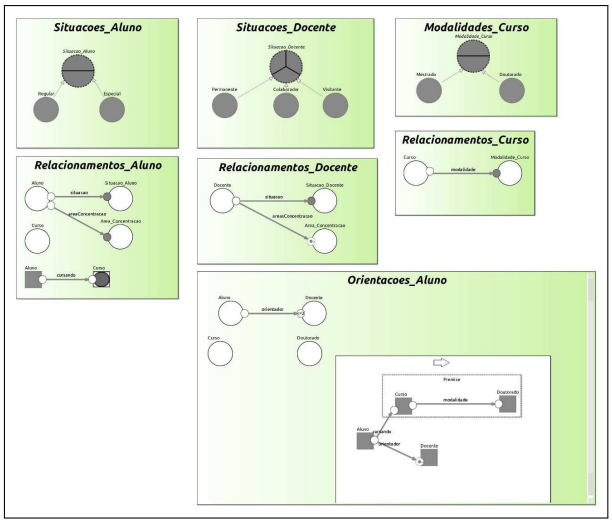

Figura E.39: Participante 06 – Modelo Conjunto 4

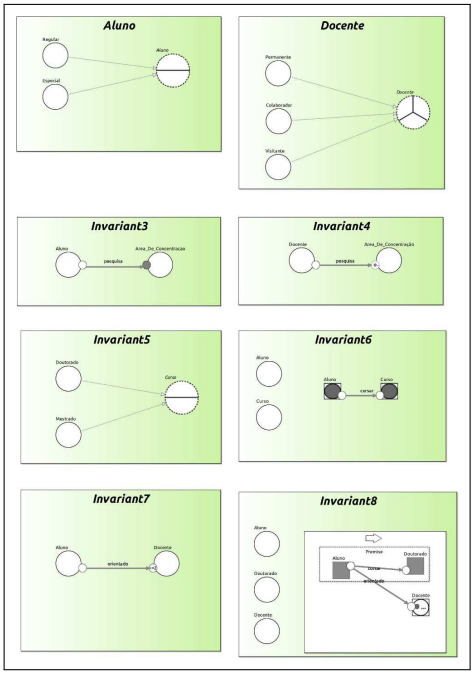

Figura E.40: Participante 07 – Modelo Conjunto 4

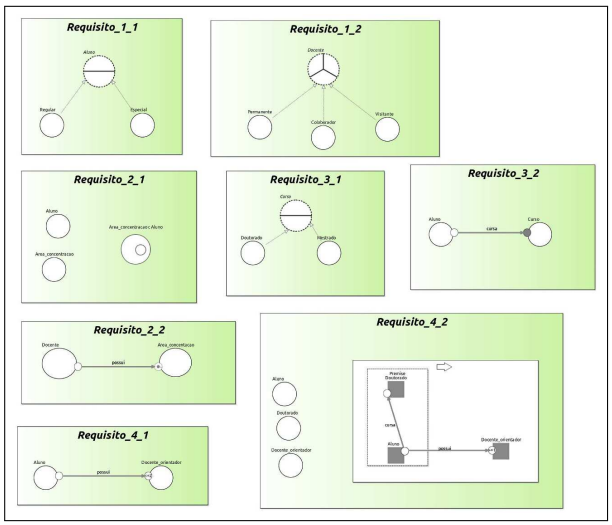

Figura E.41: Participante 08 – Modelo Conjunto 4

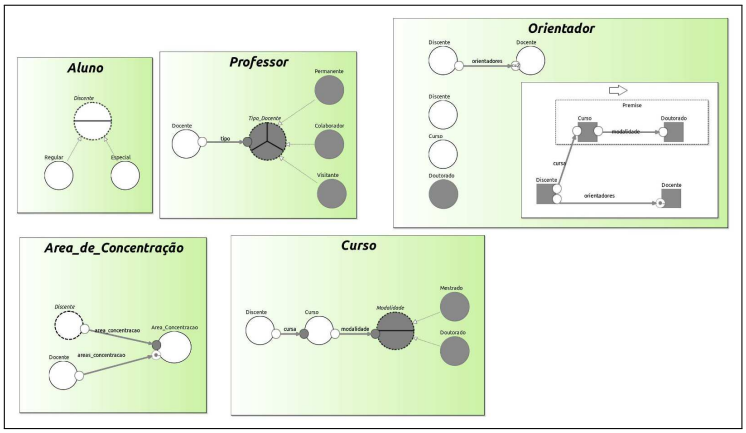

Figura E.42: Participante 09 – Modelo Conjunto 4

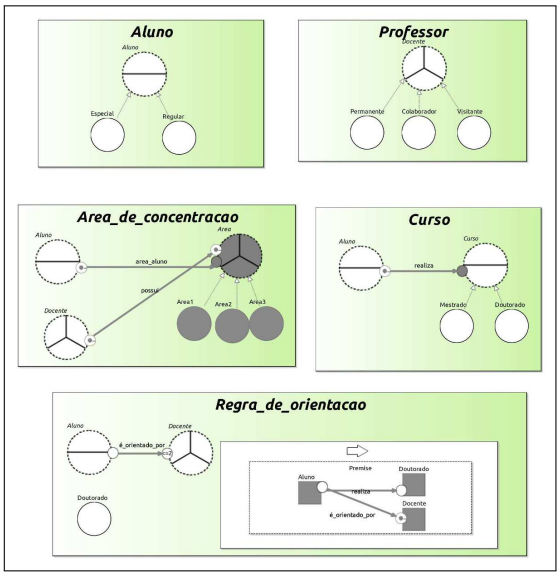

Figura E.43: Participante 10 – Modelo Conjunto 4

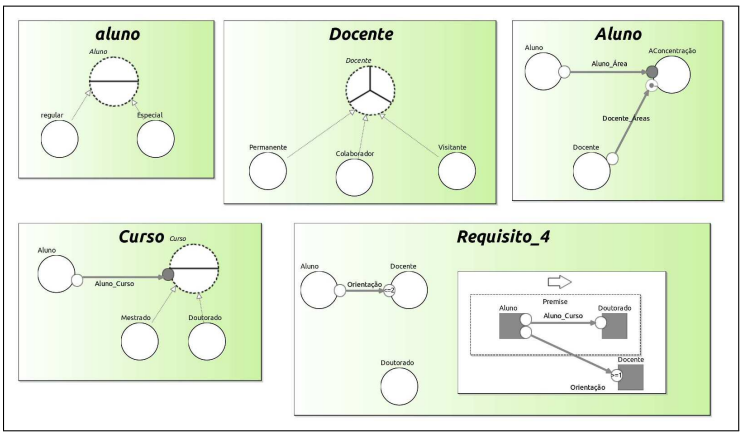

Figura E.44: Participante 11 – Modelo Conjunto 4

Apêndice F

Passos da Construção das Soluções

Tabela F.1: Requisito 1.1 - Passos das Soluções

*	Requisito 1.1										*
R1.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ Aluno abs.	+ Aluno abs.	+ Aluno nab.	+ Tipo_ Aluno nab.	+ Aluno abs.	+ Situ-acao_ Aluno nabs.	+ Aluno nab.	+ Aluno nab.	+ Discente nab	+ Aluno nab	+ Regular nab.
2	+ Regular nab.	+ Regular nab.	+ Situação nab. x Situação + Situação abs.	+ Regular nab.	+ Regular nab.	+ Regular nab.	- Aluno nab.	+ Regular nab.	+ Regular nab.	- Aluno nab.	+ Especial nab.
3	+ Especial nab.	+ Especial nab.	+ Regular nab.	+ Especial nab.	+ Especial nab.	+ Especial nab.	+ Aluno abs.	+ Especial nab.	+ Especial nab.	+ Aluno abs.	? extends Regu-lar de Invariant
4	+ extends Regular e Especial de Aluno	+ extends Regular e Especial de Aluno	+ Especial nab.	- Tipo_ Aluno nab. i Tipo_ Aluno abs.	+ extends Regular e Especial de Aluno	+ singleton Regular e Especial	+ Regular nab.	- Aluno nab,	+ extends Regular e Especial de Aluno	+ Regular nab.	Intervi - conceito abstract.
5			+ singleton em Re-gular e Especial	+ extends Regular e Especial de Aluno	+ singleton Regular e Especial	- Situ-acao_ Aluno nabs.	+ Especial nab.	+ Aluno abs.	Ao fa-zer R1.2 decidiu:	+ Especial nab.	+ Aluno abs.
6			x Situação nab.	+ Tipo_ Aluno abs.		+ Situ-acao_ Aluno abs.	+ extends Regular e Especial de Aluno	+ extends Regular e Especial de Aluno	- Discente nab.	+ extends Aluno de Regular	+ extends Regular e Especial de Aluno

Tabela F.1: (continued)

*	Requisito 1.1										*
R1.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
7			+ Situação abs.	+ singleton em Regular e Especial		+ extends Regular e Especial de Situacao_ Aluno			+ Discente abs.	- extends Aluno de Regular	
8			+ extends 3 e 4 de Situação	+ extends 2 e 3 de Tipo_ Aluno		+ Aluno nab.				+ extends Regular e Especial de Aluno	
9			+ relship Aluno 'situacao' Situação	Depois de R3.2 : + Aluno nab.		+ Situacao_ Aluno nab.					
10			+ mult ONE em Situação	+ relship Aluno 'tem_ tipo_ aluno' Tipo_ Aluno		+ relship Aluno 'relation-ship_ ' Situacao_ Aluno					
11			Trocou nome Situacao_ Aluno	+ mult. ONE em Aluno (esq)		+ mult ONE em Situacao_ Aluno'					
12				+ mult ONE em Tipo_ Al...							

Tabela F.2: Requisito 1.2 - Passos das Soluções

*	Requisito 1.2										*
R1.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ Docente abs.	+ Docente abs.	+ Docente nab.	+ Docente abs.	+ Docente abs.	+ Situacao_ Docente abs.	+ Docente abs.	+ Docente abs.	+ Docente nab.	+ Docente abs.	+ Docente abs.
2	+ Permanente nab.	+ Permanente nab.	+ Situacao abs.	+ Permanente nab.	+ Permanente nab.	+ Permanente nab.	+ Permanente nab.	+ Permanente nab.	+ Tipo_ Docente nab.	+ Permanente nab.	+ Permanente nab.
3	+ Colaborador nab.	+ Colaborador nab.	+ Permanente nab.	+ Colaborador nab.	+ Colaborador nab.	+ Colaborador nab.	+ Colaborador nab.	+ Colaborador nab.	+ Permanente nab.	+ Colaborador nab.	+ Colaborador nab.
4	+ Visitante nab.	+ Visitante nab.	+ Colaborador nab.	+ Visitante nab.	+ Visitante nab.	+ Visitante nab.	+ Visitante nab.	+ Visitante nab.	+ Colaborador nab.	+ Visitante nab.	+ Visitante nab.
5	+ extends Permanente, Colaborador e Visitante de Docente	+ extends Permanente, Colaborador e Visitante de Docente	+ Visitante nab.	+ singleton Permanente, Colaborador e Visitante	+ extends Permanente, Colaborador e Visitante de Docente	+ singleton Permanente, Colaborador e Visitante	+ extends Permanente, Colaborador e Visitante de Docente	+ extends Permanente, Colaborador e Visitante de Docente	+ Visitante nab.	+ extends Permanente, Colaborador e Visitante de Docente	+ extends Permanente, Colaborador e Visitante de Docente
6			+ singleton Permanente, Colaborador e Visitante	+ extends Permanente, Colaborador e Visitante de Docente	+ singleton Permanente, Colaborador e Visitante	+ extends Permanente, Colaborador e Visitante de Situacao_ Docente			+ singleton Permanente, Colaborador e Visitante		

Tabela F.2: (continuação)

*	Requisito 1.2										*
R1.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
7			+ extends Permanente, Colaborador e Visitante de Situacao	Depois de R3.2 : + Docente nab.		Colocando relationship em Aluno			+ relship Docente 'tipo' Tipo_ Docente (sem multiplicity)		
8			+ relship Docente 'situacao' Situacao	+ relship Docente 'tem _ tipo_ docente' Tipo_ Docente		+ relship Docente 'situacao' Situacao			Em R3.1:		
9			+ mult ONE em Situacao	+ mult. ONE em Docente (esq)		+ mult ONE em Situacao_ Docente			+ mult ONE em Tipo_ Docente		
10			Intervi - nomes = de entidades Situacao	+ mult ONE em Tipo_ Docente		Verificou					
11			Trocou nome Situacao_ Docente								
12			Verificou								

Tabela F.3: Requisito 2.1 - Passos das Soluções

*	Requisito 2.1										*
R2.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ Aluno nab.	+ Aluno nab.	+ Area-Concentracao nab. (na inv de R1.1)	+ Aluno nab.	+ Aluno nab.	+ Area_ Concentracao (na invariante de R.1.1)	+ Aluno nab.	+ Aluno nab.	+ Discente nab.	+ Copy Aluno (de R1.1)	+ AConcentração
2	+ Área_ Concentração nab.	+ Área_ de_ Concentração nab.	+ relship Aluno 'area_ concentracao' Area-Concentracao	+ Área_ de_ Concentração	+ Area_ de_ concentracao nab.	+ relship Aluno 'area-Concentracao' Area_ Concentracao	+ Area-De-Concentracao nab.	+ Area_ concentracao nab.	+ Area_ Concentracao nab.	+ 'Area' (abs)	+ singleton em AConcentração
3	? Containment	+ relship Aluno 'tem' Área_ de_ Concentração	+ mult. ONE em Area-Concentracao	Foi para R2.2 fazendo ao mesmo tempo	+ relship Aluno 'area' Area_ de_ concentracao	+ mult. ONE em Area_ Concentracao	+ relship Aluno 'relationship_' AreaDe-Con	+ Containment	+ relship Discente 'area_ concentracao' Area_ Concentracao	+ Area1	Intervi - conceito de singleton
4	? Multiplicity ou Cardinality	+ mult ONE em Área_ de_ Concentração		+ relship Aluno 'tem' Área_ de_ Concentração	+ mult ONE em Area_ de_ concentracao		+ mult. ONE em Aluno (esquerda)	+ copiou 1º Area_ concentracao	+ mult. ONE em Area_ Concentracao	+ Area2	+ Aluno nab.

Tabela F.3: (continuação)

*	Requisito 2.1										*
R2.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
5	+ relship Aluno ‘relation-ship_ ’ Área_ Concen-tração	Durante R4.2		? Cardina-lity	+ mult. ONE em Aluno		- mult. ONE em Aluno	+ copiou 2º Aluno		+ Area3	+ relship Aluno ‘Aluno_ Área’ AConcen-tração
6	+ mult ONE em Area_ Concen-tração	+ trocou nome relship ‘aluno_ tem’		+ mult. SOME em Aluno (esquerda)	Em R2.2: i nome relship ‘area_ concen-tracao		+ mult. SOME em AreaDe-Concen-tracao (dizendo só 1)			+ singleton em Area1, Area2 e Area3	+ mult. ONE em AConcen-tração
7	Durante R2.2: + nome relship ‘tem’:			- mult. SOME em Aluno (esquerda)			+ nome relship ‘pesquisa’			+ extends Area1, Area2 e Area3 de Area	
8				+ mult. SOME em Área...			x mult. SOME em Area...			+ relship Aluno ‘possui’ Area	
9				- mult. SOME em Área...			+ mult. SOME em Area...			+ mult. ONE em Area	
10				+ mult. ONE em Aluno (esq)							
11				- mult. ONE em Área...							

Tabela F.4: Requisito 2.2 - Passos das Soluções

*	Requisito 2.2										*
R2.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ Docente nab.	+ Docente nab.	+ Area-Concentracao nab. (na inv de R1.21)	+ Docente	+ Docente	+ Area_Concentracao (na invariant R1.2)	+ Docente nab.	+ Docente nab.	+ Docente (mesma invariant)	Copiou Docente de R1.2	+ Docente nab.(mesma invariant de R2.2)
2	+ Área_Concentração nab.	+ relship Docente 'tem' Área_de_Concentração (do R.2.1)	+ relship Docente 'area_concentracao' Area-Concentracao	+ Área_de_Concentração (duas na mesma invariant)	+ relship Docente 'area' Area_de_concentracao	+ relship Docente 'relationship_' Area_Concentracao (plural no nome)	+ AreaDe-Concentração nab. (nome diferente da de Aluno)	+ Area_concentracao nab.	+ relship Docente 'areas_concentracao' Area_Concentracao (plural)	+ relship Docente 'possui' Area	+ relship Docente 'Docente_Áreas' AConcentracao
3	+ relship Aluno 'relationship_' Área_Concentração	+ mult SOME em Área_de_Concentração	+ mult. SOME em Area-Concentracao	+ relship Docente 'tem' Área_de_Concentração	+ mult SOME em Area_de_concentracao	+ mult. SOME em Area_Concentracao	+ relship Docente 'pesquisa' AreaDe-Concentração	+ relship Aluno 'possui' Area_concentracao		+ mult. SOME em Area	? Cardinality (1 ou mais? Apontando para multiplicity)
4	+ mult ONE em Area_Concentração	Durante R4.2	Verificou	+ mult. SOME em Docente (esquerda)	+ nome relship 'area_concentracao	Verificou	Viu erro na mult da relship de Aluno:	+ mult. SOME em Area_concentracao			+ mult SOME em AConcentração

Tabela F.4: (continuação)

*	Requisito 2.2										*
R2.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
5	+ nome relship ‘tem’	+ trocou nome relship ‘docente_ tem’		+ mult. SOME em Área_ de_ Concentração		+ nome relship Docente ‘areas-Concentracao’ Area_ Concentracao (plural no nome)	+ mult. SOME em AreaDeConcentração				
6				Verificou x Erro			Verificou				

Tabela F.5: Requisito 3.1 - Passos das Soluções

*	Requisito 3.1										*
R3.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ trocou nome colocando Docente_ no entidades que estendem Docente	+ Curso nab.	+ Curso nab. (na invariant R1.1 e para o R3.2)	+ Tipo_ Curso abs.	+ Curso abs.	Iniciou pelo R3.2	+ Curso abs.	+ Curso nab.	? Set Operation (porque não usou)	+ Curso abs.	+ Mestrado nab.
2	+ Curso abs.	+ Mestrado nab.	+ Curso abs (em nova invariant)	+ Mestrado nab.	+ Doutorado nab.	+ Modalidade_ Curso abs.	+ Doutorado nab.	+ Doutorado nab.	+ Modalidade abs.	+ Mestrado nab.	+ Doutorado nab.
3	+ Mestrado nab.	+ Doutorado nab.	+ Mestrado nab.	+ Doutorado nab.	+ Mestrado nab.	+ Mestrado nab.	+ Mestrado nab.	+ Mestrado nab.	+ Mestrado nab.	+ Doutorado nab.	+ Curso abs.
4	+ Doutorado nab.	+ extends Mestrado e Doutorado de Curso	+ Doutorado nab.	+ singleton Mestrado e Doutorado	? relationship	+ Doutorado nab.	+ extends Doutorado e Mestrado de Curso	- Curso nab.	+ Doutorado nab.	+ extends Mestrado e Doutorado de Curso	+ extends Mestrado e Doutorado de Curso

Tabela F.5: (continuação)

*	Requisito 3.1										*
R3.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
5	+ extends Mestrado e Doutorado de Curso	- Curso nab.	+ extends Mestrado e Doutorado de Curso + extends	+ extends Mestrado e Doutorado de Tipo_ Curso	+ extends Doutorado e Mestrado de Curso	+ singleton Mestrado e Doutorado		+ Curso abs.	+ singleton Mestrado e Doutorado		
6	Após R4.1:	+ Curso abs.		+ Curso nab.		+ relship Curso 'modalidade' Modalidade_ Curso		+ extends Doutorado e Mestrado de Curso	+ extends Mestrado e Doutorado de Modalidade		
7	+ renomeou iniciando Curso_ Mestrado e Doutorado	+ extends Mestrado e Doutorado de Curso		+ relship Curso 'relationship' Tipo_ Curso		+ mult. ONE em Modalidade_ Curso			+ Curso nab.		

Tabela F.5: (continuação)

*	Requisito 3.1										*
R3.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
8				+ mult SOME em Curso (esq)					+ relship Curso 'modalidade' Modalidade		
9				+ mult ONE em Tipo_ Curso					+ mult. ONE em Modalidade		
10				? Cardinality							
11				+ Cardinality							
12				+ copia Curso p/ Cardinality							
13				- Cardinality (é redundante)							

Tabela F.6: Requisito 3.2 - Passos das Soluções

*	Requisito 3.2										*
R3.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	? relation-ship	+ Aluno nab.	+ Curso abs. (na invariant R1.1)	+ Aluno nab. (na mesma invariant)	+ Aluno nab.	+ ALL (Aluno) (iniciou por esse)	+ ONE(Aluno)	+ Aluno nab.	+ Aluno nab. (na invariant de R3.1)	+ copiou Aluno	+ Aluno nab. (na invariant de R3.1)
2	+ Aluno nab.	+ relship Aluno ‘tem’ Curso	+ relship Aluno ‘cursa’ Curso	+ relship Aluno ‘cursar’ Curso	+ Curso nab.	+ ONE (Curso)	+ ONE(Curso)	+ Curso nab.	+ relship Aluno ‘cursa’ Curso	+ copiou Curso	+ relship Aluno ‘relation-ship_ ’ Curso
3	+ relship Aluno ‘relation-ship_ ’ Curso	Verificou	+ mult. ONE em Curso	+ mult. ONE em Curso	+ relship Aluno ‘curso’ Curso	? relation-ship	+ relship ONE(Aluno) ‘cursar’ ONE(Curso)	+ relship Aluno ‘pode_ cursar’ Curso	+ mult. ONE em Curso	+ relship Aluno ‘realiza’ Curso	+ mult. ONE em Curso
4	+ mult. LONE em Curso	+ trocou nome relship para ‘cursa’	? Quan-tification ou como fez?	Verificou	+ mult. ONE em Curso	+ relship ALL(Aluno) ‘cur-sando’ ONE(Curso)	Verificou	+ mult. ONE em Curso		+ mult. ONE em Curso	Verificou. Deu erro.

Tabela F.6: (continuação)

*	Requisito 3.2										*
R3.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
5	+ trocou nome da relation-ship para ‘cursa’		Verificou	Corrigir ligar Aluno a Tipo_ Aluno R1.1 e Docente a Tipo_ Docente	Verificou	Verificou					
6				+ mult SOME em Aluno (esq)							
7				Verificou (deu erro)							

Tabela F.7: Requisito 4.1 - Passos das Soluções

*	Requisito 4.1										*
R4.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ Aluno nab.	+ Aluno nab.	+ Docente nab. Na invariante R1.1	+ Aluno nab.	+ Aluno nab.	+ Aluno nab.	+ Aluno nab.	+ Aluno nab.	+ Aluno nab.	Após modelar o segundo:	+ Aluno nab.
2	+ Docente nab.	+ Orientador nab.	+ relship Aluno ‘orientado_ por’ Docente	+ Docente nab.	+ Docente nab.	+ Docente nab.	+ Docente nab.	+ Docente nab.	+ renomeou Aluno para Discente	+ relship Aluno ’é_ orientado_ por’ Docente	+ Orientador1 nab.
3	+ relship Aluno ‘Orientador’ Docente	+ relship Aluno ‘orientado’ Orientador	+ ALL	+ cardinality	? cardinality	+ relship Aluno ’orientador’ Docente	+ relship Aluno ’relationship_ ’ Docente	+ relship Aluno ’tem’ Docente	+ Docente nab.	? cardinality	+ Orientador2 nab.
4	+ mult. “<=2” em Docente	+ mult. “<=2”	? cardinality	+ copiou Docente p/ cardinality	+ relship Aluno ‘orientador’ Docente	+ mult “<=2” em Docente	? cardinality	+ mult.”<=2” em Docente	+ relship Discente ’orientadores’ Docente	+ integer	? representação
5			+ cardinality	+ Integer 2	+ mult. “<=2” em Docente		+ mult. ONE em Aluno	Durante R4.2:	? Set Operation	Intervi - multiplicity	Intervi - Há entidade maior que Orientador1 e 2?

Tabela F.7: (continuação)

*	Requisito 4.1										*
R4.1	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
6			Intervi - expressão na multiplicity	+ LESS_EQUALS – Bug ficou “>=2”	+ mult. ONE em Aluno		- mult. ONE em Aluno	+ trocou nome de Docente para Docente_Orientador	+ mult. “<=2” em Docente	+ mult. “<=2” em ALL(Docente)	+ Docente nab.
7			+ mult. “<=2”	+ relship Aluno ‘ter_docentes’ Docente	Intervi - leitura da mult. ONE.		+ mult. “<=2” em Docente		? cardinality		+ relship Aluno ’Orientação’ Docente
8				+ mult. “<=2” em Docente	- mult. ONE de Aluno		+ nome relship ’orientado’		+ cardinality		? Relational Operation
9									+ copiar ’orientadores’ em cardinality		+ LESS_EQUAL em multiplicity de Docente
10									+ Docente em cardinality		Intervi - expressão na multiplicity
11											+ mult. “<=2” em Docente

Tabela F.8: Requisito 4.2 - Passos das Soluções

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
1	+ Curso nab.	+ Aluno nab.	i ALL(Aluno) Na invariante R1.1	+ implication (na invariant de R4.1)	! implication	+ ALL(Curso)	+ Doutorado nab.	+ Aluno nab.	+ Discente nab.	Iniciou pelo segundo.	+ Doutorado nab.
2	+ Curso_ Doutorado nab.	+ Curso abs.	+ ALL(Doutorado)	+ ALL(Aluno)	+ Docente (na invariante de R4.1)	+ ALL(Doutorado)	+ Docente nab.	+ Doutorado nab.	+ Curso nab.	? Set Operation	! implication
3	? relationship entre entidades	+ Doutorado nab.	+ ALL(Docente)	Indiquei que Doutorado pode assumir Tipo_ Curso na relationship	+ relship Aluno 'relationship' Docente	+ ALL(Aluno)	+ relship Doutorado 'relationship_ ' Docente	+ Docente_ Orientador nab.	+ relship Discente 'relationship_ ' Curso	+ 3 ALL	Intervi - relationship na implication tem que ser quantificada
4	+ ALL(Aluno)	+ extends Doutorado de Curso	+ relship ALL(Aluno) 'cursa' ALL(Doutorado)	+ Doutorado nab.	Intervi - relationships permitidas em implication	+ ALL(Docente)	+ singleton em Regular e Especial	+ ALL(Aluno)	+ Doutorado	+ copiou Aluno da invariante R1.1	+ ALL
5	+ ALL(Curso_ Doutorado)	+ relship Aluno 'relationship_ ' Curso	+ ALL(Aluno) 'orientador_ por' ALL(Docente)	+ ALL(Doutorado)	+ Aluno nab.	+ relship ALL(Curso) 'modalidade' ALL(Doutorado)	- singleton em Regular e Especial	+ ALL(Doutorado)	+ relship Curso 'relationship_ ' Doutorado	+ copiou Docente da invariante R1.2	? copiar relship 'Orientação' para a quantification

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
6	+ relship ALL(Aluno) ‘Curso’ ALL(Curso_ Douto- rado) x premissa	- Aluno, Curso e relship ‘relation- ship_ ’	+ mult. SOME em ALL(Docente)	+ Tipo_ Curso nab.	+ Orienta- dor nab.	+ relship ALL(Aluno) ’cur- sando’ ALL(Curso)	+ singleton em Dou- torado	+ ALL (Docente_ Orienta- dor)	Intervi - implica- tion com quantifi- cation nas relation- ships	+ copiou Douto- rado da invariante R3.1	+ copiou Douto- rado em ALL
7	+ Docente nab.	+ Aluno	Verificou	+ ALL(Tipo_ Curso)	Renomeou Orienta- dor p/ Docente	+ relship ALL(Aluno) ’orienta- dor’ ALL(Docente)	- singleton em Douto- rado	+ relship ALL(Aluno) ’relation- ship_ ’ ALL(Douto- rado)	- relship primeira	+ copiou Aluno, Docente e Douto- rado para 3 ALL	+ ALL(Aluno)
8	+ SOME(Docente)	+ Curso		+ SOME(Docente)	+ Douto- rado	+ mult. SOME em ALL(Docente)	- relship Dou- torado ’relation- ship_ ’ Docente	+ relship ALL(Aluno) ’relation- ship_ ’ ALL(Docente_ Orienta- dor)	- relship segunda	+ relship ALL(Aluno) ’realiza’ ALL(Douto- rado)	+ ALL(Docente)

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
9	+ relship ALL(Aluno) ‘Orientador’ SOME(Docente)	+ relship Aluno ‘cursa’ Curso		+ Curso nab.	+ ALL(Aluno)	+ implication	+ implication	+ mult. “>=1” em ALL(Docente_ Orientador)	+ ALL(Curso)	+ relship ALL(Aluno) ’é_ orientado_ por’ ALL(Docente)	+ relship ALL(Aluno) ’Aluno_ Curso’ ALL(Doutorado)
10	+ implication	+ relship Doutorado ‘orientador’ Orientador		+ ALL(Curso)	+ ALL(Docente)	+ copiou relship ’modalidade’	+ ALL(Aluno)	+ copiou duas relships juntas premissa ficou na segunda.	+ ALL(Doutorado)	+ implication	+ ALL(Aluno) ’Orientação’ ALL(Docente)
11	Intervi - cópia na implication	? Cardinality		+ relship ALL(Curso) ‘tem’ ALL(Doutorado)	+ ALL (Doutorado)	+ copiou relship ’cursando’	+ ALL(Doutorado)	- desfez copia relships	+ relship ALL(Curso) ’modalidade’ ALL(Doutorado)	+ copiou ’realiza’	+ mult. “>=1” em ALL(Docente)
12	Copiou premissa	? Quantification		+ relship ALL(Aluno) ‘cursar’ ALL(Curso)	+ relship ALL(Aluno) ‘curso’ ALL(Doutorado)	+ copiou relship ’orientador’	+ relship ALL(Aluno) ’cursar’ ALL(Doutorado)	+ trocou nome da primeira relship para ’cursa’	? relship ALL(Discente) ALL(Docente)	+ copiou ’é_ orientado_ por’	Verificou

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
13	Copiou conclusão	? Logical Operation		+ relship ALL(Aluno) ‘tem’ SOM(Docente)	+ relship ALL(Aluno) ‘orienta-dor’ ALL(Docente)	Verificou	+ SOME(Docente)	+ trocou nome da segunda relship para ’possui’	+ mult. SOME em ALL(Docente)	+ mult. ONE em ALL(Douto-rado)	
		Intervi - relships p/ implica-tion		+ copiou relship 11 p/ implica-tion	+ mult. “<=2” em ALL(Docente)		+ relship ALL(Aluno) ’orien-tado’ SOME(Docente)	+ copiou relship ’cursa’ para a implica-tion	+ ALL(Discente)	+ mult. SOME para a es-querda de todas as relações exceto a do R4.1	
14		- Aluno, Curso e relship ‘cursa’		+ copiou relship 12 p/ implica-tion	+ implica-tion		? mult. >=1 em SOME(Docente) = quantifi-cation	+ copiou relship ’pos-sui’ para implica-tion	+ ALL(Docente)	+ mult. SOME em ALL(Docente)	

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
15		+ Aluno nab.		+ copiou relship 13 p/ implication	+ copiou relship ‘curso’		Verificou	Verificou	+ relship ALL(Discente) ’orientadores’ ALL(Docente)	Verificou. Deu erro = nome das relationships (trocamos possui para area_aluno) e multiplicity da esquerda das relações na quantification	
16		+ Cardinality		Verificou	+ copiou relship ‘orientador’				+ tentando copiar as duas relships sem a ligação entre Discente e Curso.	Verificou.	
17		! Implication		Bug.	Trocou mult. “<=2” para “>=1”				Intervi mostrado a lacuna.		

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
18		Retirou R4.1 Refez R4.1			Verificou				+ relship ALL(Discente) 'cursa' ALL(Curso)		
19		+ Aluno nab.							+ implication		
20		+ Curso nab.							+ copiou 'cursa' para implication		
21		Renomeou Curso p/ Doutorado e desfez							+ copiou 'orientadores' e 'modalidade' para implication		
22		Renomeou Orientador para Docente							Verificou		
23		+ Containment									
24		+ copiou Docente depois Orientador									

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
25		+ Trocou para Orientador in Docente									
26		Parada p/ sessão expirada questionário									
27		+ Implication									
28		+ Curso									
29		+ Doutorado									
3		+ Docente									
31		+ Orientador									
32		Intervi - dica montar relship com quantification									
33		+ ALL(Aluno)									
34		+ ALL(Doutorado)									

Tabela F.8: (continuação)

*	Requisito 4.2										*
R4.2	**P01**	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**	**P09**	**P10**	**P11**
35		i relship ALL(Aluno) ‘cursa’ ALL(Douto- rado)									
36		- Curso e Docente									
37		+ ALL(Orienta- dor)									
38		+ relship ALL(Aluno) ‘orien- tado’ ALL (Orienta- dor)									
39		+ copy relship cursa									
4		+ copy relship orientado									

Apêndice G

Anotações de Dúvidas, Achados, Problemas e Sugestões

Tabela G.1: Achados Registrados em Observações / Questionário

* Achados *			
Part.	**Decrição**	**Sigla**	**Tipo**
P03	E: 'Obrigatoriamente, mesmo nome, mesmo conceito! ' (apontando para as entidades de Área de Concentração em Aluno e Docente)	AA	AA
P09	E: 'Eu devia ter pensado em: "menos é mais". Quis complicar pra mostrar que entendi.'	AA	AA
P10	E: 'Resolve o que eu tinha ...' (Sobre a sugestão que fez no final do módulo 2 em relação à Relational Operation com a multiplicidade.)	AA	AA
P09	E: 'Vou mudar uma coisa aqui (na primeira invariante)... Não faz sentido eu instanciar um discente sem saber que tipo ele é (se Regular ou Especial)'	Abstr	AA
P10	E: ' Vai ganhar 3 fatias ... Obrigatoriamente tem que ser 3 tipos associados!' (fatias = divisões da entidade abstrata)	AbstrGraf	AA
P11	E: 'Ok... e são disjuntos!' (possível referência às partições da entidade abstrata)	AbstrGraf	AA
P11	E: 'Tá ok... são disjuntos!' (por causa da partição da entidade abstrata)	AbstrGraf	AA
P08	E: 'Poderia usar no meu mestrado. Gerei código JSON.' (mostrando uma tela em que o usuário poderia gerar várias possibilidades se fosse com GIRL.)	Aplicação	AA
P09	E: 'Trabalhei em ITAOS... implementei a ferramenta gráfica... (mostrou oágina do ITAOS) ... é complicado!'	Aplicação	AA
P03	D: 'Porque usa na operação está contido ao invés da estensão (olhando para o Quadro de estensão de entidades)?' (identificou a redundância em que o conceito de containment seria o mesmo descrito na estensão de Contas exibido no primeiro vídeo)	ContaXExte	AA
P06	D: 'Qual a diferença entre "está contido"e especialização? Quando aplicar "está contido"?'	ContaXExte	AA
P07	Não fica claro na representação a logica do AND. No caso o entendimento para o programador seria um a verificação se as "classes"conta_corrente e conta_poupanca são heranças de conta.	ContaXExte	AA

Tabela G.1: (continuação)

*	Achados		*
Part.	**Decrição**	**Sigla**	**Tipo**
P11	D: 'Posso colocar as duas Contas_ em Conta? (referenciando a entidade abstrata)	ContaXExte	AA
P11	E: 'Num é Banco - Banco Central, não... Banco Central está contido em Banco? (relembrando a invariante do vídeo do Módulo 1)	DifConj	AA
P03	D: 'Por que lá (no quadro) você colocou ONE e aqui ALL?' (Sobre a diferença do quantificador na entidade Bloquado)	DifQtfConcl	AA
P06	Destacou a possibilidade de ao copiar/colar manter o mesmo identificador (viu id nas exibições dos vídeos). Achou que isso poderia ser uma fonte de problema. Resolveu não copiar para evitar possíveis problemas.	EditorID	AA
P02	E: 'Era mais fácil colocar orientador só na relação em vez de criar uma entidade Orientador'	EntityRelsh	AA
P01	Sacou (ou relembrou) o conceito que Situação no Módulo 1 era uma enumeration.	Enum	AU
P02	E: 'O soft em si tá até muito bom!'	GirlBom	AA
P03	E: 'Gostei, de verdade!' (quando perguntei se gostou da linguagem)	GirlBom	AA
P04	E: 'Esse ferramenta é boa pra dar aula!'	GirlBom	AG
P07	E: 'Gostei dessa ferramenta. Nunca tinha usado uma ferramenta para requisitos.'	GirlBom	AG
P09	Quanto as estruturas, destaco a facilidade de aprendizado e também a forma fácil de se modelar os requisitos. Também destaco a completude dos componentes e da linguagem, que possibilitam uma modelagem complexa, mas fácil e clara de se entender.	GirlBom	AA
P09	E: 'Em Java ele criaria uma coleção com no máximo 2.'	GirlJava	AA
P04	D: 'Conceito de invariante tá confuso. Nome Banco para invariante. Uma hora Banco é entidade, outra hora é invariante.' (ao ver a primeira invariante do modelo)	Invariant	AA
P05	Conseguiu ver que a relação '<=2' do requisito 1º) interfere no 2º).	MultInterf	AA
P02	Sacou que 'Dá o mesmo efeito >=1 ou multiplicidade 'cinza e 3 pontinhos' (SOME)	MultSome	AU
P03	E: '>= 1 é a mesma coisa de SOME (do dele na relação 'orientado por' da implicação)!'	MultSome	AA
P04	E: 'Cardinalidade = 1, acho que fica redundante' (referenciando a cardinalidade =1 fica redundante com a multiplicidade ONE)	MultXCard	AA
P02	E: 'Mesmo nome na hora de validadar o 'tem' de Docente confunde com o 'tem' de Aluno	NomeRelsh	AA
P07	E: 'Todo aqui fica a impressão que é todo Vencido... Fica ilógico.' (Todo = ALL)	QtfALLPremis	AA
P07	O video deixa claro onde pode ser empregado a quantificação.	Quantif	AA
P07	Lendo o Tutorial, achou a expressão: "Assim, representamos uma Conta como uma entidade abstrata que pode ser Conta Corrente ou Conta Poupança". (Estratégia de mitigar conflito pela linguagem do autor)	Requisito	AA
P07	E: 'Entendi que Banco Central englobava todos os Bancos, mas como você disse que Banco Central era um Banco, o conceito seria Banco - Banco Central.' (demonstrou que se Banco era pai de Banco Central, o contrário dos parâmetros na operação complemento seria o correto. Achou a troca do complemento. Achou mais coerente Banco - Banco Central.	SimCompl	AA
P01	D: Multiplicidade via relação representaria melhor.	SimQtf	AA
P02	D: Multiplicidade da relação vs Quantificação - qual a diferença?	SimQtf	AA

Tabela G.1: (continuação)

*	Achados		*
Part.	**Decrição**	**Sigla**	**Tipo**
P03	D: 'Qual a diferença disso (a relação com os quantificadores) e fazer (relationship - desenhando na tela com os dedos: bolinha na esquerda linha e bolinha na direita) e situação (o nome da entidade da direita) sendo singleton?' (trocando a multiplicidade ONE com o singleton - ambos bolas preenchidas de cinza)	SimQtf	AA
P06	D: 'Qual a diferença de usar isso (quantificadores) e a multiplicidade?'	SimQtf	AA
P07	Na representação R4_Orientador_Aluno_Doutorado, a multiplicidade seria desnecessária já que existe um quantificador que já representa os valores solicitados.	SimQtf	AA
P07	D: 'Não entendi esse quantificador. Ele não podia usar o relacinamento?' (referenciando que poderia ser restingido na multiplicidade apenas)	SimQtf	AA
P09	D: 'Qual a diferença de usar essa quantificação e a cardinalidade (querendo dizer multiplicidade) na relação?'	SimQtf	AA
P10	D: 'Overhead do que fez?' E: 'Poderia fazer com a relatioship e multiplicity - símbolos que você já tem!'	SimQtf	AA
P06	E: 'Só as e-num' (quando só apareceu as singletons)	VerifEnum	AA
P01	E: 'A cereja do bolo foi a Verificação, ver os negocinhos aparecendo!'	VerifiBom	AG
P02	E:'Gostei mais do validador! De tudo! É fantástico!	VerifiBom	AG
P05	E: 'É massa, vice! Principalmente a parte para testar! 'Até para o cliente fica melhor de mostrar! Análise é massa!'	VerifiBom	AA
P05	E: 'Massa, vice!'	VerifiBom	AA
P07	E: 'O mais legal foi poder criar instâncias e ver o modelo sendo aplicado.'	VerifiBom	AG

Tabela G.2: Dúvidas na Ambientação Registradas em Observações / Questionário

*	Dúvidas - Ambientação		*
Part.	**Decrição**	**Sigla**	**Tipo**
P02	D: É do Alloy as saídas das Instâncias?	Alloy	DA
P04	D: 'Preciso entender de Alloy, é? Eu nunca vi!'	Alloy	DA
P09	D: 'Alloy é usado?'	Alloy	DA
P09	D: 'Ele não mostra a análise, os casos, tipo, caso 1, caso 5, para se ter ... Solução 1, solução 2... para mencionar: caso tal, não era para ser... Tipo barra de estatus...'	Alloy	DA
P08	D: 'Cardinalidade da informação sobre titular? Cardinalidade tá dizendo quantos clientes são titulares de uma conta é?'	Cardinal	DA
P02	D: Containment - O Banco Central está dentro de um Banco (fisicamente ou negócio)?	Contain	DA
P03	D: 'Banco Central contém Banco, ou Banco Central está contido?' (apontando pro Quadro - quadro está diferente)	Contain	DA
P04	D: 'Qual é a tradução de 'Containment'?'	Contain	DA
P06	D: 'Banco Central está contido em Banco?'	Contain	DA
P08	D: 'Acho que entendi tudo... Não entendi o 'C' (cê) do "está contido"... não seria Banco Cental contém Banco?'	Contain	DA
P08	D: 'Banco contém Banco Central?'	Contain	DA
P01	D: Tem outro jeito de fazer containment? Pela aba Property? Pouco intuitivo o ctrl+C e ctrl+V.	Editor	DA
P03	D: 'Como é esse arranjado?'	Editor	DA

Tabela G.2: (continuação)

*	Dúvidas - Ambientação		*
Part.	**Decrição**	**Sigla**	**Tipo**
P03	D: 'Isso apareceu como? ' (sobre o CtrlZ sendo dado e aparecendo invariantes já preenchidas no vídeo)	Editor	DA
P04	D: 'É <F2> pra mudar é?' (querendo entender como mudar a multiplicidade)	Editor	DA
P04	D: 'Primeiro o de dentro, né? (querendo confirmar que copia primeiro o elemento que está contido)	Editor	DA
P06	D: 'Pode exportar pedaço do diagrama?'	Editor	DA
P08	D: 'Copiou primeiro o Banco Central e segundo o Banco?'	Editor	DA
P08	D: 'Ctrl C e Ctrl V na cardinalidade?'	Editor	DA
P08	D: Perguntou como faz a organização, arrangement.	Editor	DA
P09	D: 'Palleta tem todas as possibilidades?'	Editor	DA
P09	D: 'Por que aqui no início ... ajuste... pensei que tinha ficado dentro (a conclusão dentro da premissa)'	Editor	DA
P09	D: 'Posso botar, criar premissa fora e colar, depois fazer a conclusão e colar?'	Editor	DA
P10	D: 'A ordem importa, né?' (sobre argumentos da operação complemento)	Editor	DA
P01	D: Abstract é o mesmo sentido da programação? O sentido de abstract é o de Aproveitamento. Entendo que seja o esqueleto de alguma coisa.	EntityAbs	DA
P06	D: 'Quando tem interseção (entre as entidades estendidas), como representar?'	EntityAbs	DA
P06	D: 'Tem herança múltipla?'	EntityAbs	DA
P08	D: 'A nível de código, a classe abstrata Conta tem a classe Conta_Corrente? Se tá gerando a classe Conta e a Conta_Corrente seria um atributo de Conta?'	EntityAbs	DA
P08	D: 'Esse abstract é como se fosse uma classe abstrata mesmo?'	EntityAbs	DA
P01	D: Singleton. Situação Singleton - como? é diferente da programação? O: Mais na frente ao retornar para ver vídeo, ou foi para descobrir como fazer entre Multiplicidade, verbalizou que na Invariant a Entidade Situação seria uma Enumeration.	EntitySin	DA
P09	D: 'Singleton é o mesmo do código em Java?'	EntitySin	DA
P10	D: 'Existe relação das entidades entre as invariantes?'	EntityInvar	DA
P07	D: 'Essa linguagem tem compilador? Tipo UML?' (isso para mapear para implementação as entidades, relacionamentos, etc. para aproveitar o já foi modelado em GIRL para a codificação. no momento pouco antes do zoom do vídeo para mostrar a multiplicidade.)	Girl	DA
P09	D: 'Isso é plug-in?'	Girl	DA
P08	D: ' ... é muitos para muitos é?' (apontando para a relação com as duas multiplicidade da bolinha branca - SET)	GirlBD	DA
P01	D: Dúvida no uso da quantificação para Implicação.	Implicat	DA
P09	D: 'Se colocar a premissa errada, como fazer?'	Implicat	DA
P09	D: 'Se pegar Copy/Cola invertido?'	Implicat	DA
P01	D: 'Não explicou o que era invariant' - Deve ser um conceito da Eng.Requisitos que não conhecia.	Invariant	DA
P04	D: 'O que é invariante?	Invariant	DA
P05	D: 'Invariante ficou meio... Qual a extensão disso? Classificar, Escopo? Só o conceito mesmo que não entendi.'	Invariant	DA
P06	E: 'Invariante é a mesma de UML? Algo que não pode mudar.'	Invariant	DA
P08	D: 'Tudo fica dentro desse invariante, né?'	Invariant	DA
P09	D: 'O que é invariante?'	Invariant	DA
P09	D: Continuou com dúvida sobre invariante.	Invariant	DA

Tabela G.2: (continuação)

*	Dúvidas - Ambientação		*
Part.	**Decrição**	**Sigla**	**Tipo**
P10	D: 'Invariante é tipo um container? Como um set em que tá criando os objetos?'	Invariant	DA
P04	D: 'Posso botar o AND dentro do NOT?' (entendendo que o AND pode ser um elemento dentro do NOT)	Logical	DA
P06	D: 'O que tá querendo dizer?' (sobre o AND)	Logical	DA
P06	D: 'Pode AND dentro do NOT?'	Logical	DA
P07	D: 'Fica meio ... não é intuitiva a representação do AND na explicação.'	Logical	DA
P09	D: 'Eles vão ficando dentro? Exemplo um OR com um AND dentro?'	Logical	DA
P09	D: 'Só pode usar Containment?'	Logical	DA
P10	D: 'Você conhece BRmodelo para Banco de Dados, modelo conceitual?	Metodol	DA
P10	D: 'Você conhece i* (istar), Chaos? Muito do que você fez já existe. Isso é uma ameaça à validade?'	Metodol	DA
P06	D: 'Conta pode 1 ou 2?' (alusão à possibilidade de operações relacionais com 'and' (conjunção) na multiplicidade)	MultInterv	DA
P08	D: 'Pode setar intervalo de 1 a 8?'	MultInterv	DA
P04	D: 'Isso tem utilidade?' (apontando para a multiplicidade vazio) 'Se tá ligado a zero porque está ligado? Fica com cara de gambiarra! Mas, sei lá, às vezes precisa!'	Multiplici	DA
P04	D: Achou diferente como estavam as instâncias, pois estava pensando que o SOME da direita tinha o conceito de multiplicidade do SOME da esquerda. Confundiu o SOME da esquerda com o SOME da direita.	Multiplici	DA
P08	D: ' ... é muitos para muitos é?' (apontando para a relação com as duas multiplicidade da bolinha branca - SET)	Multiplici	DA
P08	D: ' 0 ou 8, né? ' (sobre o '<=8' - corrigi dizendo que é até oito, de zero a oito)	Multiplici	DA
P08	D: 'Preenchido 1, 0 ou 1, com 1 bolinha de 1 ou mais?' (corrigi que o 1 ou mais é uma bolinha e três pontinhos)	Multiplici	DA
P10	D: 'Que granularidade colocar na invariante? Conta e Situação mesmo (invariantes do vídeo)? Meu medo é numa granularidade pequena gerar uma explosão.' (muitos ou poucos elementos em invariante/diagrama)	Organiz	DA
P05	D: 'Relacionamento com quantificação não precisa de multiplicidade, né?'	QtfComMult	DA
P09	D: 'Permite editar multiplicidade aqui (em relationships com quantificação)?' (Apontando para a multiplicidade da relação no vídeo)	QtfComMult	DA
P03	D: 'A direção da esquerda para a direita (da relationship), pode ser bidirecional? Poderia ter outra relação de Cliente para Conta? Entendi que no contexto da invariante (apontando o nome da invariante Conta) seria o que está aí... mas num contexto de Cliente?'	RelshDirecao	DA
P06	D: 'Por que a direção da seta é única?'	RelshDirecao	DA
P10	D: 'Existe Tipo para fazer relações ternárias ou mais?'	Relship	DA
P10	D: 'Banco Central está contido em Banco? Não seria Banco em Banco Central? (dúvida quanto ao negócio)	Requisito	DA
P10	D: 'Podia ter uma terceira - pagamento_parcial? (apontando como se fosse uma relação a mais. Depois viu que seria uma nova entidade estendendo Situação)	Requisito	DA
P11	D: 'Mas cada Conta num só tem um Titular?' (dúvida sobre requisito)	Requisito	DA
P01	Não entendeu Banco Central contido em Banco - na prática. 'Mas como foi o requisito, está ok.'	SetOper	DA
P03	E: 'Entendi sim... sei pra que você vai usar não, mas entendi!' (sobre o Set Operation)	SetOper	DA

Tabela G.2: (continuação)

*	Dúvidas - Ambientação		*
Part.	**Decrição**	**Sigla**	**Tipo**
P04	D: 'Q que tá querendo representar na operação de conjunto? Qual é o da listrinha?' (na segunda invariante com a operação 'Complement'	SetOper	DA
P05	D: 'Banco Central num contém Banco?' (alusão ao BC ter controle sobre todos os bancos)	SetOper	DA
P09	D: 'Set Operation é ação de "setar"? (referenciando termo 'set' de programação)	SetOper	DA
P10	D: 'SetOperation, questão do complemento né?'	SetOper	DA
P09	E: 'É bacana ver as possibilidades!'	VerifiBom	AA
P01	Há outros tipos de implicação? Tipo "se e somente se"?	Implicat	DM
P01	Não entendi bem o uso de quantificadores para usar a implicação.	ImplQtf	DM
P01	Quando se digita algo como <=8 na multiplicidade da relação, o analisador consegue captar isso como regra?	Multiplici	DM
P07	A seta utilizado no relacionamento deixa a impressão que o relacionamento apenas parte de conta para cliente, e não um relacionamento de duas vias, como explicado no video. Cliente possui conta e conta possui cliente.	RelshDirecao	DM
P01	Acho que me confundi um pouco na relação entre dois quantificadores utilizando relacionamentos.	RelshQtf	DM
P06	Senti um pouco de dificuldade para entender a aplicação da parte de conjuntos. Fora isso, só exemplificou com a parte de diferença.	SetOper	DM

Tabela G.3: Dúvidas na Utilização Registradas em Observações / Questionário

*	Dúvidas - Utilização		*
Part.	**Decrição**	**Sigla**	**Tipo**
P09	D: 'Ele (tela Alloy) diz quantas soluções tem?'	Alloy	DU
P04	D: 'Como é esse negócio de Cardinalidade?'	Cardinal	DU
P05	E: 'A cardinalidade...'	Cardinal	DU
P09	D: 'Eu poderia modelar com a cardinalidade?'	Cardinal	DU
P03	D: 'Vixe... é cardinalidade, é?' (tentando repesentar "até 2"orientadores via cardinalidade)	Cardinal	DU
P11	D: 'Cardinalidade é a representação para "1 ou mais"?'	Cardinal	DU
P02	Colocando Containment para Orientador in Docente	Contain	DU
P01	D: Como acessar propriedades?	Editor	DU
P01	Tentando copiar quantificadores para a implicação sem conseguir.	Editor	DU
P06	D: 'Pode copiar?' (sobre entidades ou outra estrutura para copiar em novas invariatnes)	Editor	DU
P10	D: 'Como vejo os metadados?' (properties da multiplicidade)	Editor	DU
P10	D: 'Copia como?' (como copiar as relationships na implicação)	Editor	DU
P06	D: 'Tô na dúvida. ... Na verdade é Situação Aluno.'	Entity	DU
P07	Perguntou se podia botar na mesma área (fazer a relação utilizando a mesma entidade AreaDeConcentracao)	Entity	DU
P03	D: 'Qual a diferença de usar Curso abstrato e não abstrato?'	EntityAbs	DU
P04	D: 'Num existe um 'implement' não, né?' (alusão ao singleton (??))	EntityAbs	DU
P06	D: 'Posso botar Doutorado direto?'	EntityAbs	DU
P11	Expliquei o sentido de Invariante e a necessidade de uma entidade.	EntityAbst	DU
P07	D: 'Ele (a ferramenta) consegue entender Bloqueado no lugar de Situação_Cartão?' (usar a entidade estendida Bloqueado no lugar da entidade pai Situação_Cartão na relação)	EntityExtend	DU

Tabela G.3: (continuação)

*	Dúvidas - Utilização		*
Part.	**Decrição**	**Sigla**	**Tipo**
P10	D: 'Curso... ou copio a especialização que a gente sabe que é Doutorado?' (para decidir que entidade copiar Curso ou Doutorado)	EntityExtend	DU
P06	D: 'Posso botar Situação sem singleton ou abstract?' (sobre atributos da mesma entidade em invariantes diferentes)	EntityInvar	DU
P07	D: 'Tem ligação com demais Alunos pelo nome?' (sobre as outras entidades Aluno das outras invariantes)	EntityInvar	DU
P09	E: 'Eu ainda tava duvidando que ia usar mesmas entidades.' (sobre a possibilidade do uso das definições das outras invariantes)	EntityInvar	DU
P10	Tentou copiar entidade de outra invariante na quantificação.	EntityInvar	DU
P03	D: 'Não ficou muito claro pra mim por que você quis representar assim (sem os singletons em Doutorado e Mestrado)... não é considerando o seu errado, mas...!'	EntitySin	DU
P04	D: 'Qual o sentido? Se for aluno qualquer não é singleton, se for tipo, é singleton. É um, ou outro, ou outro. Pra essa situação tá ok. Eu gosto de e E-num' (manteve o singleton - focou no ou...ou... ou para decidir e no gosto pelo singleton)	EntitySin	DU
P06	D: 'A questão do singleton... '	EntitySin	DU
P09	D: 'Não lembro como fazer tipo enumeration.'	EntitySin	DU
P11	E: 'Essa situação (sobre tipos de Docentes) é única.'	EntitySin	DU
P11	Intervi explicando que existem mais áreas de concentração, relembrando ideia de conjunto unitário o que não se aplicaria.	EntitySin	DU
P04	E: 'Imagino que isso seja tipo um e-num' (para Regular e Especial)	Enum	DU
P08	E: 'Se eu colocar estensão aqui, Regular vais estender Aluno. Ou tipo Regular, ou tipo Especial... então isso aqui é enumeração.'	Enum	DU
P09	D: 'Não lembro como fazer tipo enumeration.'	Enum	DU
P01	D: 'O que é linguagem GIRL? É tudo de Gráfico?	Girl	DU
P07	E: 'Tô tentanto entender em relação ao Banco de Dados. Aluno tem Área e Área não tem Aluno - N pra 1.' (para definir multiplicidade)	GirlBD	DU
P01	Expliquei que só podia copiar a relação.	ImplCopy	DU
P02	D: Percebida - O que colocar na implicação?	Implicat	DU
P07	D: 'Faz tudo fora primeiro, depois bota dentro?' (sobre criar as relações fora da implicação primeiro, depois copiar para a implicação)	Implicat	DU
P03	D: 'To sem criatividade para nomear invariantes.' (pensando em tudo que estava colocando na invariante Aluno e Docente e usando apenas esses nomes simples)	Invariant	DU
P05	D: 'Crio nova invariante?'	Invariant	DU
P05	D: 'Posso colocar qualquer nome, né?'	Invariant	DU
P07	E: 'O conceito do nome de invariante - pensei que era parte da lógica que era utilizada como descrever o modelo.'	Invariant	DU
P09	E: 'Mesmo sem entender bem o que é... (invariante)'	Invariant	DU
P10	D: 'Coloco em outra invariante?' (decidiu usar invariante Aluno)	Invariant	DU
P11	Nesse momento passou a referenciar a invariante 'aluno' como sendo a entidade aluno.	Invariant	DU
P11	Clicou no "LESS_EQUAL"da Relatinal Operation e tentou colocar dentro da multiplicidade de Docente.	MultExpr	DU
P06	D: 'Se eu quisesse 1 ou 2.' (querendo saber como faria , semelhante à duvida na Ambientação)	MultInterv	DU
P05	D: 'Fecho um na esquerda?	Multiplici	DU
P05	D: 'Num tem problema não com o '<=2'? (dizendo que não haveria conflito com a multiplicidade '<=2' da relação orientador do primeiro requisito)	Multiplici	DU

Tabela G.3: (continuação)

*	Dúvidas - Utilização		*
Part.	**Decrição**	**Sigla**	**Tipo**
P11	Creio que criou as duas entidades Orientado_ associando ao termo "até 2"do requisito. Isso se vê na fala inicial "2 entidades".	Multiplici	DU
P05	D: 'Só que eu fiquei em dúvida com as multiplicidades (apontando para as multiplicidade das relações do requisito 4 com expressão relacional). Mas não conflita não, né?'	MultpInvari	DU
P01	E: 'Cardinalidade misturada com Multiplicidade'	MultXCard	DU
P01	E: 'Multiplicidade ou Cardinalidade?' Cardinalidade é conjunto.	MultXCard	DU
P03	D: 'É na multiplicidade... mas como ?'	MultXCard	DU
P09	D: 'Na relação (multiplicidade), já é suficiente?'	MultXCard	DU
P02	D: Na mesma invariante ou em outra?	Organiz	DU
P03	D: 'Posso colocar tudo junto? ' (na mesma invariante)	Organiz	DU
P05	D: 'Posso ajeitar nomes?' (das invariantes do conjunto 1 para deixar apenas a inicial maiúscula)	Organiz	DU
P05	D: 'Tenho que deixar a entidade aqui, né' (sobre criar nova entidade aluno igual a que colocara na invariante anterior)	Organiz	DU
P06	D: 'Melhor separar os dois em duas invariantes?'	Organiz	DU
P09	D: 'Posso modelar na mesma caixinha (invariante)?'	Organiz	DU
P09	D: 'Tem alguma diferença modelar assim (do jeito dele)?'	Organiz	DU
P03	D: 'Poderia pegar isso (mapear o 'até 2') na quantificação?'	QtfComMult	DU
P03	D: 'Posso representar assim, ou por quantificadores?'	QtfComMult	DU
P03	D: 'Posso representar assim? Ou por quantificadores?'	QtfComMult	DU
P07	D: 'A relação de quantificação tem multiplicidade?'	QtfComMult	DU
P01	D: qual o quantificador de Docente?	Quantif	DU
P07	D: 'Esse EXISTS é um ou mais?' (significado do quantificador EXISTS)	Quantif	DU
P03	D: 'Posso relacionar (criar relationships entre) entidades de outras invariantes?'	RelshInva	DU
P01	E: 'Usar relacionamento e multiplicidade?' (Dúvida para decidir o que usar)	Relship	DU
P05	D: 'Aqui tenho que criar relações com multiplicidade, é?'	Relship	DU
P06	D: 'Como expresso algo.algo - algo-ponto-algo?' (em referência a aluno.cursando.modalidade)	Relship	DU
P06	D: 'É relationship, é?' (sobre como conectar as quantificações)	Relship	DU
P11	D: 'Posso relacina-lo (Aluno) com Curso ... estendendo... ?'	Relship	DU
P03	D: 'E para o nome das relações... se forem iguais, tem problema ? ' (as duas relações para situação de Aluno e Docente também estavam iguais.)	RelshNome	DU
P08	D: 'Porque o 'contém' e o 'relacionamento' tem ideia de 'ter'. (justificando a sua escolha) Entidade 'tem' a outra... E... Fiquei com dúvida em relationship versus containment.'	RelshXConta	DU
P01	E: 'É relacionamento ou inclusão ?' (containment)	RelsXConta	DU
P01	D: como elicitar (deve ser entender) o requisito mais do que com a ferramenta.	Requisito	DU
P01	D: Dúvida para entender o que significa 'deve' (do Requisito 1 e 2) ou 'pode' (do Requisito 3).	Requisito	DU
P03	D: 'Quando você escreve no requisito "pode ser"e "deve ser", você quer dizer coisas diferentes, ou a mesma coisa?' (informei da simples variação de termos para a mesma coisa)	Requisito	DU
P05	D: 'É um Tipo é, isso?	Requisito	DU
P05	D: 'Um curso pode ter zero alunos?'	Requisito	DU

Tabela G.3: (continuação)

*	Dúvidas - Utilização		*
Part.	**Decrição**	**Sigla**	**Tipo**
P06	D: 'É nível (a classificação), Mestrado e Doutorado? ' (querendo nomear o que é a classificação de curso)	Requisito	DU
P08	D: 'Curso pode ser Doutorado. Mas, tem que ser um dos dois né? Não pode ser nada.'	Requisito	DU
P08	D: 'Não pode ser os dois, né?'	Requisito	DU
P09	D: 'Na pós, a gente define o que, esse... ? (significando tipo de curso)	Requisito	DU
P09	D: 'Pensando... Curso de Pós(-graduação) né? Não tem outros!'	Requisito	DU
P10	D: 'Área pode ser singleton, né? Como a questão do e-num.'	Requisito	DU
P10	D: 'Especialização Total ou Parcial, exemplo MBA?'	Requisito	DU
P10	D: 'Você chama aqui Orientador como Docente, né?'	Requisito	DU
P10	Modificando as multiplicidades dos elementos da esquerda das relações anteriores (exceto da relação com Entidades do requisito 4) para SOME.	Requisito	DU
P05	D: 'Pode diminuir a quantidade? (de instâncias visualizadas) 'Posso só testar a implicação?' (prevendo que ia ter algumas instâncias não significativas - maior quantidade - para o que queria verificar)	Verificador	DU
P08	Achei difícil de redimensionar alguns componentes.	Editor	DN
P05	Uma dificuldade foi lembrar sobre o uso de quantificar na relação dentro da implicação.	Implicat	DN
P06	O uso de implicações e de quantificadores foi mais complexo.	Implicat	DN
P08	Achei difícil diferenciar o contém (Containment) do possui (Relacionamento).	RelshXConta	DN
P10	Sobrecarga de alguns construtos gerando dúvida de uso (e.g. no caso da cardinalidade).	SobrecConstr	DN

Tabela G.4: Problemas e Sugestões Registrados em Observações / Questionário

*	Problemas e Sugestões		*
Part.	**Decrição**	**Sigla**	**Tipo**
P04	Bug do Analyser - quando tem incosistência. (duas relações ter_docente e tem_docente	BUG	BUG
P04	BUG: O <= está exibindo >='	Bug	BUG
P06	Não achou instância de aluno de mestrado sem orientador - muito extensa a quantidade de instâncias. (G2A Colocou o SOME ALL(Docente) na relação geral)	BUG	BUG
P01	No contexto do trabalho do participante, um aluno pode ficar no sistema sem curso, antes do cadastramento. (verificar com ele)	O-Participan	O
P02	(15:53) Percebi que o último requisito online ' - Alunos do curso Doutorado devem ter um ou mais Docentes orientadores.' estava != ao impresso '- Se um Aluno cursa Doutorado, então ele deve ter um ou mais Docentes orientadores.'	Problema	PR
P02	(16:15+-)Sessao do Questionário caiu - (eu interrompi e fui verificar a sessão do questionário)	Problema	PR
P02	Eclipse caiu (15:51)	Problema	PR
P04	Deu erro na verificação - não foi feita a verificação. Não tive a ideia de abrir o Alloy pra tentar fazer. Próximas, farei isso. Depois da sessão, identifiquei que era um erro por causa de inconsistências no modelo e o analisador tentava abrir um arquivo que não existia (o caminho estava para a máquina de Dayvson)	Problema	PR

Tabela G.4: (continuação)

*	Problemas e Sugestões		*
Part.	**Decrição**	**Sigla**	**Tipo**
P04	Rodou e deu ERRO de novo no Analyser.	Problema	PR
P04	Solicitei que modificasse as duas relações do requisito 2 para aluno_tem e docente_tem. Duas relações com mesmo nome para Area de Concentração.	Problema	PR
P10	Deu inconsistência. Ajustei nomes de relacionamentos iguais 'possui' para entidade Area e retirei as multiplicidades da esquerda da relação da premissa. Rodou.	Problema	PR
P11	Deu problema no Analyser. Não copiei corretamente o projeto de modelos.girl. Corrigi.	Problema	PR
P02	Não sacou de início o 'Se' (problema = o requisito não estava com o 'Se' = o impresso estava e o da workspace não)	Problema Diferença Requisito	PR
P07	E: 'Poderia ser auto-completar para aproveitar nomes já usados. Em representações maiores, procurar nome fica difícil.'	Editor	SU
P10	Destacou baixa usabilidade de Relational Operation. Sugeriu algo semelhante à multiplicidade.	Relational	SU
P10	S: 'Bom colocar isso - Não ternária!'	Relship	SU
P10	- Captura de audio, video e interação com o respondente.		SU
P10	- Possibilidade de emprego de alguns termos cunhados (e.g. termos de banco de dados).	GirlBD	SU
P09	Na verificação, achei necessário uma melhor navegação entre as soluções apresentadas e identificação das mesmas.	Verificador	DN

Tabela G.5: Observações Repetidas

*	Observações Repetidas		*
Part.	**Decrição**	**Sigla**	**Tipo**
P01	1) Os quantificadores e a cardinalidade são os mais difíceis de saber quando utilizar.	=	
P01	No R6 b) Acredito que não entendi bem o uso dos quantificadores junto com implicações. E acabei usando o lugar errado pra representar o requisito "ter um ou mais".	=	
P06	Não entendi muito bem quando utilizar Abstract + Herança e quando usar containment.	=	
P06	Poderia ter comentado rapidamente que poderia fazer AND dentro de OR, etc.	=	
P06	Poderia usar um exemplo que só pudesse utilizar quantificação: ficou confuso porque já existe multiplicidade para isso. Então, quando uso quantificação?	=	
P07	Não fica claro por a necessidade da quantificação, pois poderia ser utilizado a multiplicidade.	=	
P08	Achei complicado diferenciar o Containment do o Relacionamento comum.	=	

Apêndice H

Registro dos Participantes Sobre as Estruturas dos Módulos

Tabela H.1: Observações Registradas pelo Participante ao Final de cada Módulo

*	Observações após Ambientação no Módulo					*
Part.	**Módulo 1**	**Módulo 2**	**Módulo 3**	**Módulo 4**	**Módulo 5**	**Módulo 6**
P01	São simples de entender e manipular.		Quando se digita **algo como <=8 na multiplicidade** da relação, o **analisador consegue captar isso como regra**?		Acho que me confundi um pouco na relação entre dois quantificadores utilizando relacionamentos.	Não entendi bem o uso de **quantificadores para usar a implicação**. Há **outros tipos de implicação**? Tipo se e somente se?
P06	Não entendi muito bem quando utilizar **Abstract e Herança** e quando usar containment.	Senti um pouco de dificuldade para entender a aplicação da **parte de conjuntos**. Fora isso, **só exemplificou com a parte de diferença**.		Poderia ter comentado rapidamente que poderia fazer **AND dentro de OR**, etc.	Poderia usar um **exemplo** que só pudesse utilizar **quantificação**: ficou confuso porque **já existe multiplicidade para isso**. Então, quando uso **quantificação**?	
P07			A seta utilizado no relacionamento deixa a impressão que o relacionamento apenas parte de conta para cliente, e não um **relacionamento de duas vias, como explicado no video**. Cliente possui conta e conta possui cliente.	Não fica claro na representação a **logica do AND**. No caso o entendimento para o programador seria um a verificação se as classes conta_corrente e conta_poupanca são **heranças de conta**.	Não fica claro por a necessidade da quantificação, pois poderia ser utilizado a multiplicidade.	O video deixa claro onde pode ser empregado a quantificação.
P09	Devido a falta de utilização em linguagens de programação do termo **Invariant**, para mim, não ficou bem claro a sua utilização e em qual contexto.		Na exibição da solução do mecanismo de **verificação, achei pertinente enumerar as soluções**, para facilitar a identificação de cada caso apresentado, se possível.		Apenas fiquei em dúvida sobre **quando utilizar operadores de multiplicidade entre relações e quando utilizar quantificação**.	

Apêndice I

Experiências e Registro dos Participantes com a Verificação de Requisitos

Tabela I.1: Experiências Durante a Verificação -
Em: R1 a R4, Modelagem do Conj. de Requisitos 1 a 4
Em: +, pontos positivos

*	Experiências na Verificação	*
Part.	**Em**	**Descrição**
P01	R4	"Viu as duas relações 'cursar' e 'Curso' em Aluno e viu que eram nomes diferentes."
	R4	Descobriu um aluno sem nenhum Orientador. Viu que não tinha problema por causa do 'pode'.
	R4	Sugeri que mudasse o nome da relação 'Curso' da implicação para 'cursa'. A mesma da relação do requisito 3.
	R4	Sugeri que verificasse se relação de 'orientador' estava igual no modelo. Estava igual.
	R4	Modificou e rodou o Analisador
	R4	"Ah... bem mais simples!" Sem a relação Curso duplicada em cursa.
	R4	"Mestrado pode ter ou não orientador."
	+	Extremamente útil. Especialmente executando de forma incremental (Registro nos pontos positivos sobre o mecanismo de verificação).
P02	R3	Viu que entendeu trocado as setas nas relações 'tem' por 'cursa'.
	R4	"O nome diferente ajuda a ver setas na verificação".
	+	A VERIFICAÇÃO do modelo representado por meio da VISUALIZAÇÃO de INSTÂNCIAS foi útil para a descoberta de inconsistências no que foi modelado ou no próprio requisito: SIM, o melhor de todos, ótimo para descobrir inconsistências já na modelagem (Registro dos pontos positivos).
P03	R1	"Opa, deu problema ... " Sobre um Docente e um Aluno com situação 'Regular'. Não tinha salvado. Salvou e rodou novamente.
	R1	"Pronto!"
	R2	"Quero um Docente com duas Áreas." Para ver uma instância com essa característica.
	R2	"Satisfeito!"
	R4	Rodou e olhou várias instâncias.
	R4	"Ok, jovem!"
	+	A verificação foi de suma importância (Registro nos pontos positivos).
P04	R3	Solicitei que modificasse as duas relações do requisito 2 para aluno_tem e docente_tem. Duas relações com mesmo nome para Area de Concentração, assim resolveria a inconsistência. Só então verificou.

Tabela I.1: (continuação)

*	Experiências na Verificação	*
Part.	**Em**	**Descrição**
	R3	Viu que não tinha ligado Aluno a Tipo_Aluno.
	R3	Corrigiu o requisito 1 adicionando as entidades Aluno e Docente e relações ligando aos respectivos Tipos.
	R3	"Aluno só pode cursar 1 curso!" E colocou SOME na relação na multiplicidade de Aluno na relação 'cursa'. Quando verificou apresentou falha no Analyser. Razão, a multiplicidade em Aluno na esquerda da relação cursar.
	+-	Foi (útil), com a ressalva que não funcionou em todas as vezes. Talvez se tivesse sido exposto as limitações, como não usar nomes repetidos nas relações, teria sido melhor aproveitado (Registro nos pontos negativos).
P05	R3	"Um aluno só pode ter um curso."
	R3	"Um curso pode ter zero alunos?". Pensando na multiplicidade da esquerda da relação 'curso.
	R3	Observador perguntou se nos requisitos tinha algo sobre isso. Respondeu que não.
	R3	O observador interviu perguntando o porquê de ter apenas uma instância para Regular, Especial e as entidades de Docentes. Como não sabia, o observador destacou que eram as entidades singleton. O participante retirou os singletons por causa da intervenção do observador.
	R4	"Pode diminuir a quantidade? (de instâncias visualizadas) Posso só testar a implicação?" Prevendo que ia ter algumas instâncias não significativas para o que queria verificar.
	R4	"Só que eu fiquei em dúvida com as multiplicidades." Apontando para as multiplicidade das duas relações do requisito 4 com expressão relacional. "Mas não conflita não, né?"
P06	R1	"Só as e-num." Quando numa instância apareceram apenas as entidades singletons.
	R1	Pensou que tinha lido duas relações erradas. Estava bastante atento às instâncias.
	R2	"Eita.... 'relationship' aqui!" Percebendo o nome 'relationship' na relação, indicando que não tinha nomeado o relacionamento em Docente. Renomeou e verificou novamente.
	R2	"Enfim, tá ok!"
	R3	"Quero ver se tem aluno que tá cursando nada!"
	R3	"Ok, acho que tá certo!"
	R4	"Queria Aluno de Mestrado com zero." Zero significando nenhum orientador.
	R4	Não achou instância de aluno de mestrado sem orientador. Estava demorando para encontrar. No entanto havia um *bug* na transformação de GIRL para ALLOY que atribuiu a multiplicidade SOME na relação 'orientador' da *signature* Aluno. Nesse caso, só permitiria um ou mais orientadores e, por isso, o analisador do ALLOY não mostraria nenhuma instância com as características que o participante estava tentando encontrar.
	+	A visualização de instâncias foi útil para validar se os requisitos foram modelados corretamente (Registro nos pontos positivos).
P07	R2	"Posso rodar?"
	R2	"Beleza!"
	R3	"Não tá colocando mais de um Curso por Aluno. Tenho Cursos que pode ou não ter Aluno. Tá correto!"
	Sug.	Você pode até tirar uns *prints* e considerar como Caso de Uso (Sugerido nos registros da avaliação de GIRL.
	+	Foi importante sim, pois serve, para o programador, como um forma de "debugar" o que foi especificado, prevenindo até que passe algum erro que não seria pego caso o analista tivesse somente o modelo (Registro nos pontos positivos).
P08	R2	A Verificação foi útil para avaliar o modelo proposto
P09	R2	"Não sei se tá certo...". Mas, não verificou.
	R3	"Não quero rodar... .fazer como faço ... Eu sempre faço tudo, depois é que testo em programação."
	R4	"É bacana ver as possibilidades!"
	R4	"E eu nem gosto de modelagem..."
	R4	"Ele (tela Alloy) diz quantas soluções tem?"
	+	A verificação foi bastante útil e precisa, embora ache necessário a enumeração das mesmas, para facilitar a identificação e navegação entre elas (Registro nos pontos positivos).

Tabela I.1: (continuação)

*	Experiências na Verificação	*
Part.	**Em**	**Descrição**
	+	Destaco como aspecto positivo as soluções apresentadas após a modelagem, que permite identificar possíveis falhas na modelagem (Registro nos pontos positivos).
	-	Na verificação, achei necessário uma melhor navegação entre as soluções apresentadas e identificação das mesmas (Registro nos pontos negativos).
P10	R4	Deu inconsistência. Ajustei nomes de relacionamentos iguais 'possui' para entidade Area e retirei as multiplicidades da esquerda da relação da premissa. Em seguida verificou.
	+	A verificação semiautomática dos diversos cenários (Como ponto positivo registrado). Quando referenciou o mecanismo 'semiautomático' de verificação, explicou que considerou um mecanismo sem formalismo.

Apêndice J

Codificação para Registro de Passos na Modelagem dos Requisitos em GIRL

Tabela J.1: Codificação do Registro de Passos na Modelagem

*	Codificação dos Passos	*
Estrutura	**Ação**	**Descrição**
Entity - Não Abstrata	Criar	**+** <nome_entidade> **nab.**
Entity - Abstrata	Criar	**+** <nome_entidade> **abs.**
Entity - Singleton	Atribuir	**+ singleton em** <nome_entidade> (ou lista de entidades)
Entity - Extends	Criar	**+ extends** <nome_entidade_filha> de <nome_entidade_pai>
Relationship	Criar	**+ relship** <nome_entidade_esquerda> '<nome_relationship>' <nome_entidade_direita> ou <referencia_quantificador_esquerda> '<nome_relationship>' <referencia_quantificador_direita>
Multiplicity	Criar	**+ mult** <tipo_multiplicidade> em <nome_entidade_OU_referencia_quantificador_direita> tipo_multiplicidade = **ONE**: apenas 1, **SOME**: 1 ou +, **LONE**: 0 ou 1, **NO**: 0, **EXPR**: expressão relacional
Quantification	Criar	**+** <tipo_quantification> (<nome_entidade_expressao>) tipo_quantification = **ALL**, **ONE**: apenas 1, **SOME**: 1 ou +, **NO**: 0
Integer	Criar	**+ integer** <número_inteiro>
	Remover	**-** <identificador_elemento_para_Remover>
	Outros Textos	Texto explicativo resumido.

Apêndice K

Gráficos da Análise Quantitativa

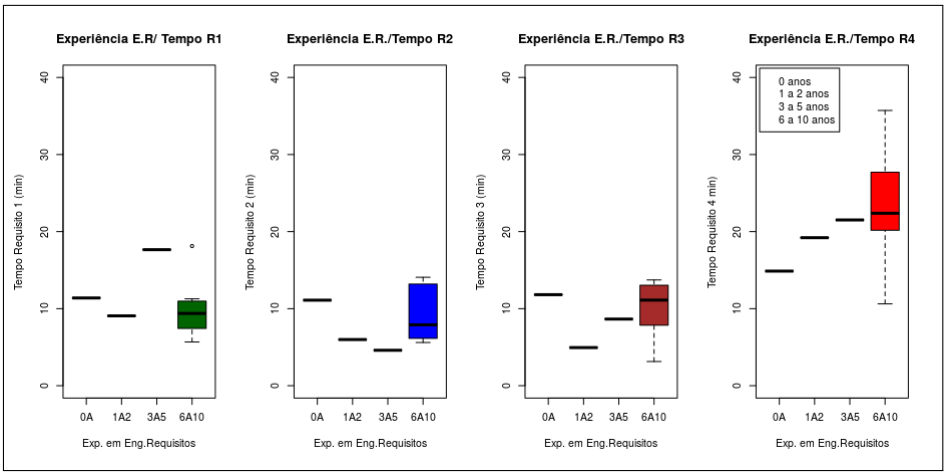

Figura K.1: Experiência em Engenharia de Requisitos / Tempo Utilização

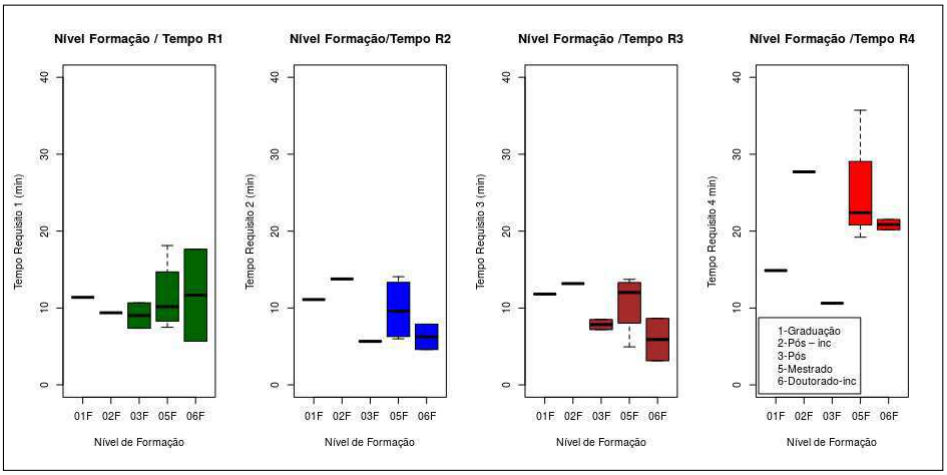

Figura K.2: Nível de Formação / Tempo Utilização

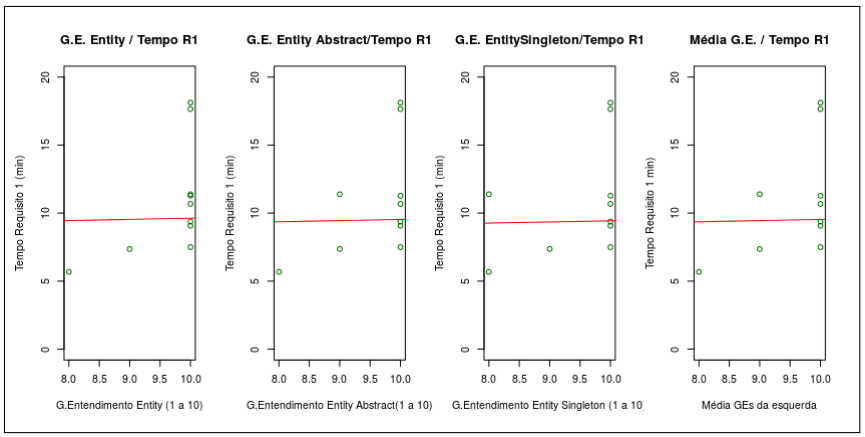

Figura K.3: Grau de Entendimento de Estruturas Usadas / Tempo Requisito 1

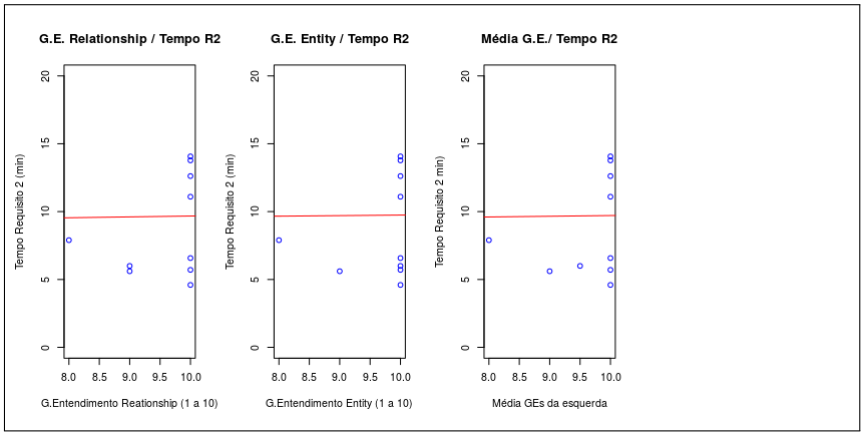

Figura K.4: Grau de Entendimento de Estruturas Usadas / Tempo Requisito 2

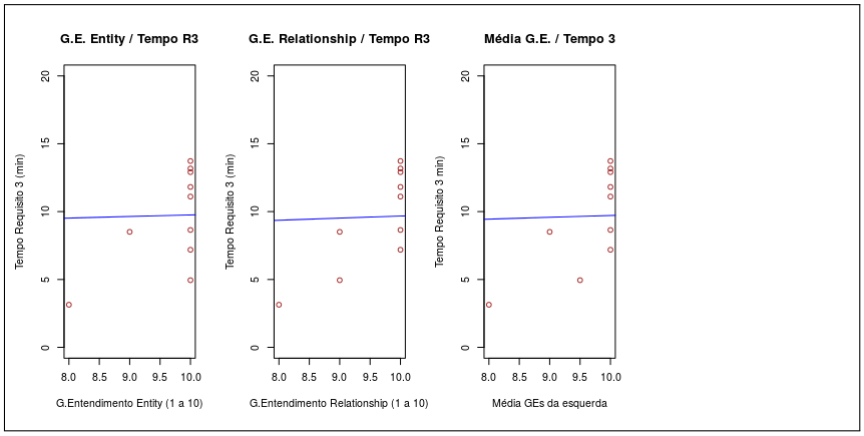

Figura K.5: Grau de Entendimento de Estruturas Usadas / Tempo Requisito 3

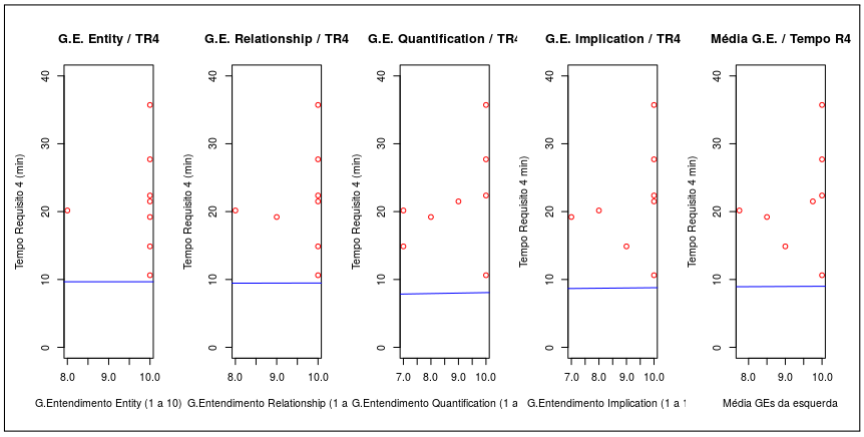

Figura K.6: Grau de Entendimento de Estruturas Usadas / Tempo Requisito 4

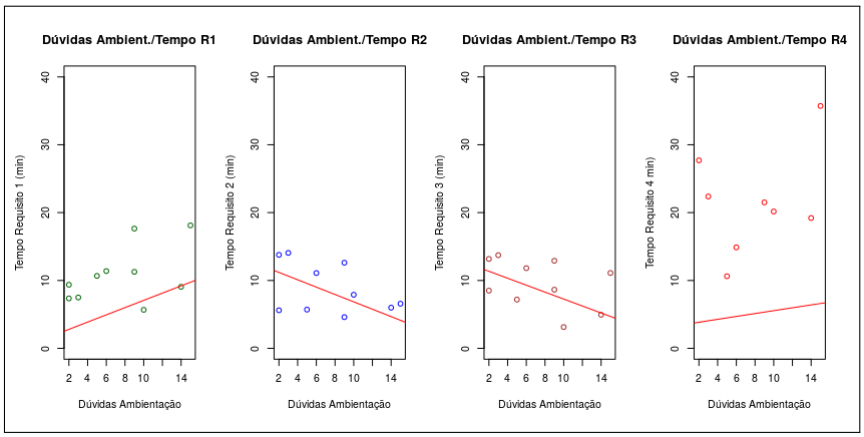

Figura K.7: Dúvidas na Ambientação / Tempo de Utilização Requisitos

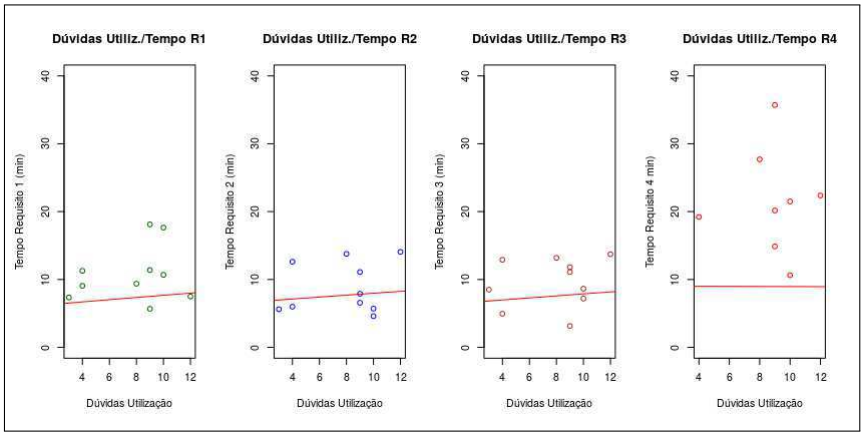

Figura K.8: Dúvidas na Utilização / Tempo de Utilização Requisitos