Classificando e implementando *feedbacks* para aprendizado ativo em ferramentas CASE: o caso EERCASE

Robson N. Fidalgo, Edson A. Silva, Natália M. Franco

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
CEP 50740-560 – Recife – PE – Brasil

{rdnf, eas4, nmf}@cin.ufpe.br

Abstract. *The use of feedbacks is an important educational tool to enrich the active learning process. There are many studies about use of feedbacks in computational tools for general purpose. However, in the context of Computer Aided Software Engineering (CASE) tools, this is a neglected issue, because no recent work classifies and guides the implementation of feedbacks in CASE. From this observation, this paper proposes 1) a feedback classification for CASE tools and 2) a practical approach to implement them. As proof of concept of these proposals, they have been implemented in a CASE tool for database conceptual design.*

Resumo. *O uso de Feedbacks é um importante instrumento pedagógico para enriquecer o processo de aprendizagem ativa. Existem muitos trabalhos sobre feedback para ferramentas computacionais de propósito geral. Contudo, no contexto de ferramentas do tipo Computer Aided Software Engineering (CASE), este é um tema negligenciado, pois nenhum trabalho recente, que classifique e oriente a implementação de feedbacks em ferramentas CASE, foi encontrado. A partir desta constatação, este trabalho propõe: 1) uma classificação de feedbacks para ferramentas CASE e 2) uma abordagem prática para implementá-los. Como prova de conceito das propostas em questão, estas foram implementadas em uma ferramenta CASE para projeto conceitual de Banco de Dados.*

1. Introdução

Ferramentas do tipo *Computer Aided Software Engineering* (CASE) visam aumentar a produtividade e a qualidade dos artefatos de software. De acordo com o paradigma de *Model-Driven Development* (MDD) [Brambilla et al. 2012], a construção de uma ferramenta CASE requer a especificação de pelo menos 3 elementos [Brambilla et al. 2012]: 1) sintaxe concreta (i.e., a representação gráfica da linguagem de modelagem); 2) sintaxe abstrata (i.e., o metamodelo ou gramática da linguagem de modelagem) e 3) semântica (i.e., o significado de cada construtor e suas regras de semântica estática). Ferramentas CASE são empregadas em várias áreas do conhecimento. Na educação, estas podem ser usadas para dar *feedbacks* sobre a modelagem de um dado domínio, enriquecendo o aprendizado e provendo uma visão prática sobre o que os aprendizes podem encontrar na indústria. Além disso, o uso de uma ferramenta CASE provê um ambiente que favorece o aprendizado ativo, pois incentiva a interação do aprendiz com a ferramenta e com outros aprendizes.

DOI: 10.5753/cbie.sbie.2015.308

Uma das vantagens da utilização de ferramenta CASE como instrumento pedagógico é a possibilidade de esta fornecer *feedbacks* individualizados [Alves e Jaques 2014]. Considerando o contexto de aprendizado ativo com base em *feedbacks*, Shute (2007) define que os *feedbacks* em ferramentas computacionais devem moldar a percepção, a ação e a cognição do aprendiz, conduzindo-o a um caminho correto e à reflexão do seu comportamento. Na mesma direção, White (2003) e Kasprzak (2005) destacam que o *feedback* é um recurso fundamental na modalidade educacional *on-line*, pois contribui para a motivação do aprendiz, amparando-o em seus questionamentos, superando o isolamento e direcionando o seu caminhar. Ademais, segundo McKeachie *et al.* (1987), esse tipo de prática interativa promove o aprendizado ativo, pois ajuda a reter o conteúdo aprendido por mais tempo e motiva a pesquisa de conhecimento adicionais.

Apesar de *feedback* ser um recurso pedagógico que enriquece o processo de aprendizagem ativa, após uma revisão inicial no estado da arte, encontrou-se apenas duas referências [Jankowski 1995][Jankowski 1997] diretamente relacionadas à tipos de *feedbacks* em ferramentas CASE. Vale destacar que o trabalho de Jankowski explora o tema de forma teórica. Portanto, há uma carência de trabalhos que aborde o tema de forma prática (i.e., como implementar *feedbacks* em ferramentas CASE). Assim, visando avançar o estado da arte sobre *feedbacks* em ferramentas CASE, os objetivos/contribuições deste artigo são: 1) propor uma classificação de tipos de *feedbacks* para ferramenta CASE e 2) apresentar, de forma prática, como estes tipos *feedbacks* podem ser implementados.

O método para alcançar os objetivos supracitados seguiu as seguintes etapas: 1) busca e investigação de trabalhos correlatos; 2) identificação do problema de pesquisa; 3) proposta de uma solução para o problema de pesquisa; e 4) avaliação da solução proposta. Vale destacar que as etapas "3" e "4" foram realizadas na ferramenta para projeto conceitual de Banco de Dados (BD) chamada EERCASE [Alves et al. 2014], pois esta: 1) dá apoio ao ensino do modelo EER, o qual é um conteúdo obrigatório nos cursos de computação [ACM/IEEE 2008][ACM/AIS 2010] e, em se tratando de Projeto Conceitual de BD, o modelo EER é tão expressivo quanto usar o Diagrama de Classe da UML [Bavota et al. 2011]; 2) é a única ferramenta CASE que dá suporte a todos os construtores da notação de Elmasri e Navathe [Elmasri e Navathe 2010] (a mais próxima à notação original de Chen [Chen 1976]) e 3) é reconhecida como um caso de sucesso da IDE eclipse (http://eclipse.org/epsilon/users).

O restante deste artigo está organizado da seguinte forma: na Seção 2, faz-se uma revisão do estado da arte de *feedbacks* em ferramentas CASE; na Seção 3, discute-se a classificação proposta neste trabalho; na Seção 4, mostra-se uma abordagem prática para implementar *feedbacks* em ferramentas CASE baseadas em MDD; 4) na Seção 5, apresenta-se e avalia-se como os *feedbacks* são providos pela EERCASE; e na Seção 6 faz-se as considerações finais e indicação trabalhos futuros.

2. Revisão do estado da arte de *feedbacks* em ferramentas CASE

Apesar de existirem muitos trabalhos [e.g., Mory (2004) e Vasilyeva et al. (2007)] que propõem tipos de *feedback* para ambientes computacionais, foram encontrados apenas duas referências [Jankowski 1995][Jankowski 1997] diretamente relacionadas ao tema

"*feedbacks*" em ferramentas CASE. Ou seja, este tema tem recebido pouca atenção da comunidade científica e industrial. Vale destacar que, diferentemente de outras ferramentas computacionais, as ferramentas CASE exigem uma visão particular do tema em questão, pois têm a peculiaridade de serem baseadas em linguagens diagramáticas e apoiarem as atividades de Engenharia de Software. Os parágrafos a seguir apresentam uma compilação dos principais tipos de *feedback* para ferramentas computacionais de propósito geral e CASE.

Um *feedback* pode assumir muitas formas e não há um consenso sobre os seus diferentes tipos de aplicação. Os trabalhos de Mory (2004) e Vasilyeva et al. (2007) analisam diferentes tipos de *feedbacks*, os quais podem ser classificados em três dimensões [Saul et al. 2010]: resposta, ocorrência e apresentação, ilustrados na Figura 1. Nesta figura, observa-se que na interseção das dimensões tem-se "Nenhum-*feedback*" – nível mínimo que relata apenas uma contagem do desempenho sem nenhuma referência aos itens avaliados. Na sequencia, na dimensão Resposta, tem-se: "Conhecimento-da-resposta" – nível de *feedback* básico que indica se a resposta está correta ou incorreta, sem fornecer informação que promova conhecimento sobre a resposta; "Responder-até-acertar" – difere do anterior, pois mantém o aprendiz no mesmo tópico até que este acerte a resposta; "Conhecimento-da-resposta-correta" – ao contrário dos anteriores, este informa qual é a resposta correta, mas não dá nenhuma explicação elaborada; "Contingente-ao-tópico" – difere dos anteriores, pois, quando ocorre um erro, fornece informação elaborada sobre a resposta correta do tópico e "Contingente-à-resposta" – difere dos anteriores, pois, para cada resposta dada, explica porque esta é correta ou incorreta. Note que estes tipos de *feedback* também são classificados como "Verificativos" (apenas relatam se houve acerto ou erro) ou "Elaborativos" (além de verificativos, fornecem informações elaboradas sobre as respostas). Por sua vez, na dimensão Ocorrência, tem-se: "Imediato" – informado assim que a tarefa é concluída e "Atrasado" – apresentado após a conclusão de um grupo de tarefas ou ao término de um período. Por fim, na dimensão Apresentação, tem-se: "Textual" – textos; "Gráfica" – imagens; "Animada" – vídeos e "Audível" – sons.

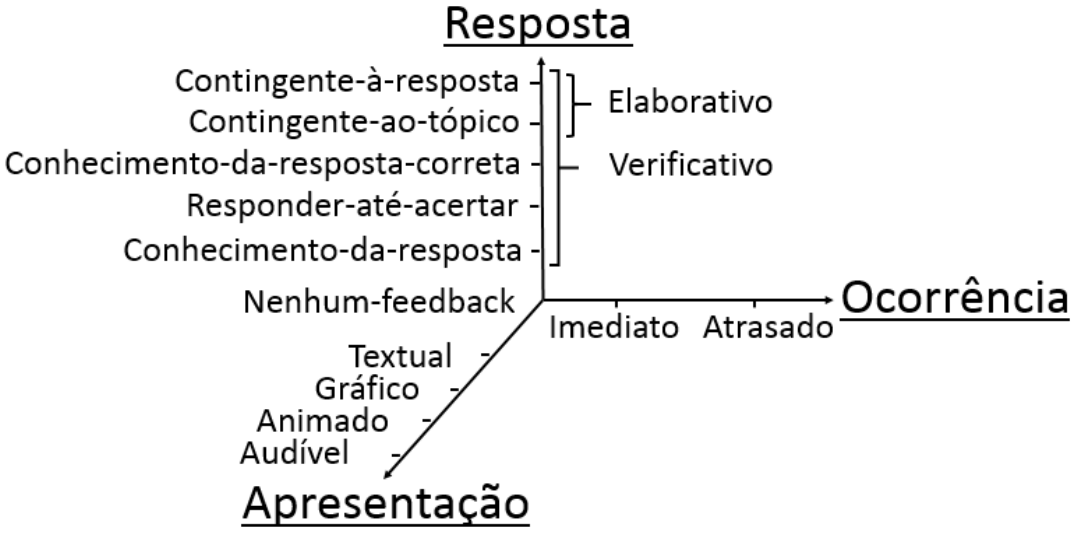

Figura 1 – Dimensões do feedback [Saul et al. 2010]

Jankowski (1997) define que um *feedback* em uma ferramenta CASE tem três propriedades: momento (i.e., durante a modelagem, ao sair/salvar, ou ao terminar a modelagem), invocação (i.e., automática ou solicitada) e aplicação (i.e., obrigatória ou opcional). Com base nestas propriedades, Jankowski explora três cenários: restritivo (i.e., os erros são automaticamente exibidos e todos devem ser corrigidos), guiado ativo (i.e., os erros são automaticamente exibidos e o usuário decide se quer corrigi-los) e guiado passivo (i.e., a exibição dos erros é solicitada pelo usuário e ele decide se quer corrigi-los). A Figura 2 resume esta discussão.

	Restritivo	**Guiado Ativo**	**Guiado Passivo**
Momento	- Durante a modelagem - Ao sair/salvar	- Durante a modelagem - Ao sair/salvar	- Durante a modelagem - Ao terminar a modelagem
Invocação	- Automática	- Automática	- Solicitada
Aplicação	- Obrigatória	- Opcional	- Opcional

Figura 2 – Propriedades de *feedbacks* em ferramenta CASE. Adaptado de [Jankowski 1997]

3. Classificando *feedbacks* para ferramentas CASE

Dado que ferramentas CASE são baseadas em linguagens diagramáticas que podem gerar código interpretável ou executável, este trabalho, diferentemente do trabalho de Jankowski (1995, 1997), propõe uma classificação de *feedbacks* para ferramentas CASE que é baseada nos tipos de erros básicos em programação e nos tipos básicos de *feedback* visto na seção anterior. Para isto, cada elemento da dimensão Resposta é classificado como: erro sintático, erro semântico ou aviso. Na Figura 3, mostra-se como esta união é feita.

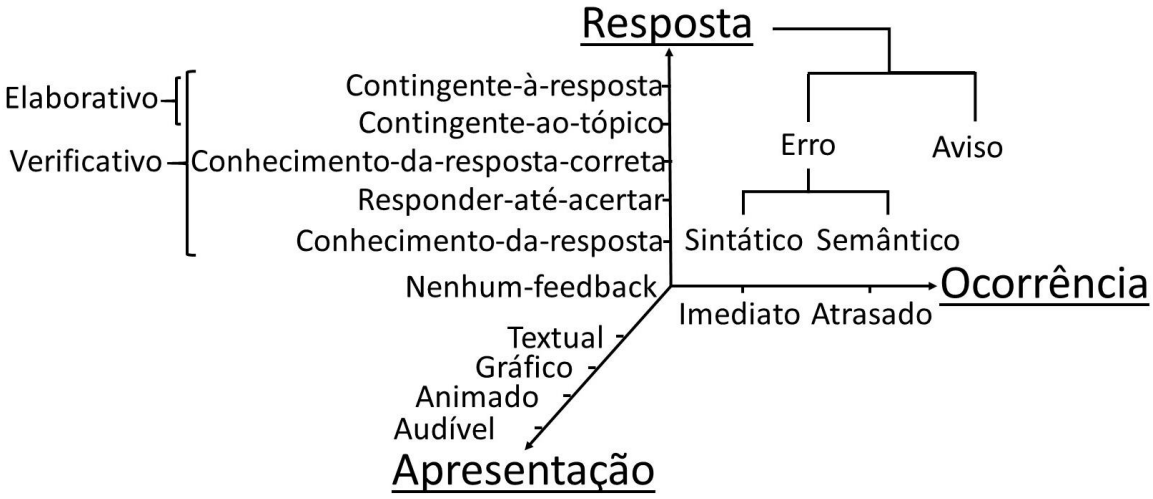

Figura 3 – Classificação de *feedback*. Adaptado de [Saul et al. 2010]

Na Figura 3, um *feedback* de erro sintático trata ações de modelagem que tentam violar regras do metamodelo. Este tipo de *feedback* deve ser de ocorrência imediata e pode ser apresentado de qualquer forma. Um *feedback* de erro semântico trata ações de modelagem que violam regras de boa formação contextual. Semelhante ao tipo anterior, este pode ser apresentado de qualquer forma. Contudo, ao contrário do tipo anterior, este pode ser de ocorrência imediata ou atrasada. Um *feedback* de aviso trata ações de modelagem que violam regras de boas práticas. As ocorrências e formas de apresentação deste tipo de *feedback* são iguais as do tipo anterior. Ressalta-se que este último tipo de

feedback é apenas sugestivo. Contudo, os dois primeiros tipos são restritivos. Isto é, devem ser corrigidos, pois, caso contrário, o modelo não será considerado como bem-formado. De modo a exemplificar estes *feedbacks*, será considerada a sintaxe, a semântica e as boas práticas do modelo EER. Assim, tem-se os seguintes exemplos: 1) Erro sintático – um relacionamento não pode estar ligado diretamente com outro relacionamento e um atributo não pode pertencer a mais de uma entidade ; 2) Erro semântico – toda entidade deve que ter ao menos um atributo identificador e não podem existir duas entidades com o mesmo nome; e 3) Aviso – inicie o nome de uma entidade com letra maiúscula e use papeis para facilitar a leitura de auto-relacionamento.

4. Implementando *feedbacks* na EERCASE

A EERCASE segue o paradigma MDD e é desenvolvida na plataforma *Eclipse* utilizando as seguintes tecnologias: *Graphical Modeling Project* (GMP) [GMP 2015] (arcabouço de tecnologias básicas para construir ferramentas CASE em *Java*) e *Epsilon Framework* (arcabouço de tecnologias que simplifica a construção de ferramentas CASE em GMP) [Kolovos et. al. 2011], as quais estão em conformidade com os padrões *Meta Object Facility* (MOF) e *Object Constraint Language* (OCL). As seções a seguir apresentam como implementar *feedbacks* sobre restrições sintáticas, restrições semânticas e críticas.

4.1 Implementando *feedbacks* sobre restrições sintáticas

Em uma ferramenta CASE baseada em MDD, seus *feedbacks* sobre erros sintáticos são especificados a partir da implementação em MOF do seu metamodelo, pois este descreve, sem ambiguidade, como os construtores da linguagem de modelagem podem ser relacionados. Ou seja, a partir de um metamodelo pode-se emitir *feedbacks* que restringem a ocorrência de erros sintáticos [Kelly and Tolvanen 2008]. Na Figura 4, mostra-se o metamodelo *Enchanced Entity Relationship Meta Model* (EERMM) [Fidalgo et al. 2012] [Fidalgo et al. 2013] usado no desenvolvimento da ferramenta EERCASE. Como base neste metamodelo a EERCASE pode dar *feedbacks* que impedem restrições sintáticas como, por exemplo, uma Ligação de Relacionamento entre Entidades ou entre Relacionamentos, pois uma Ligação de Relacionamento deve ter como origem uma Entidade e como destino um Relacionamento. Ressalta-se que, por falta de espaço, a explicação do metamodelo e o conjunto de *feedbacks* sobre restrições sintáticas providas pela EERCASE não são discutidos. Contudo, estas informações podem ser encontradas em http://cin.ufpe.br/~eercase.

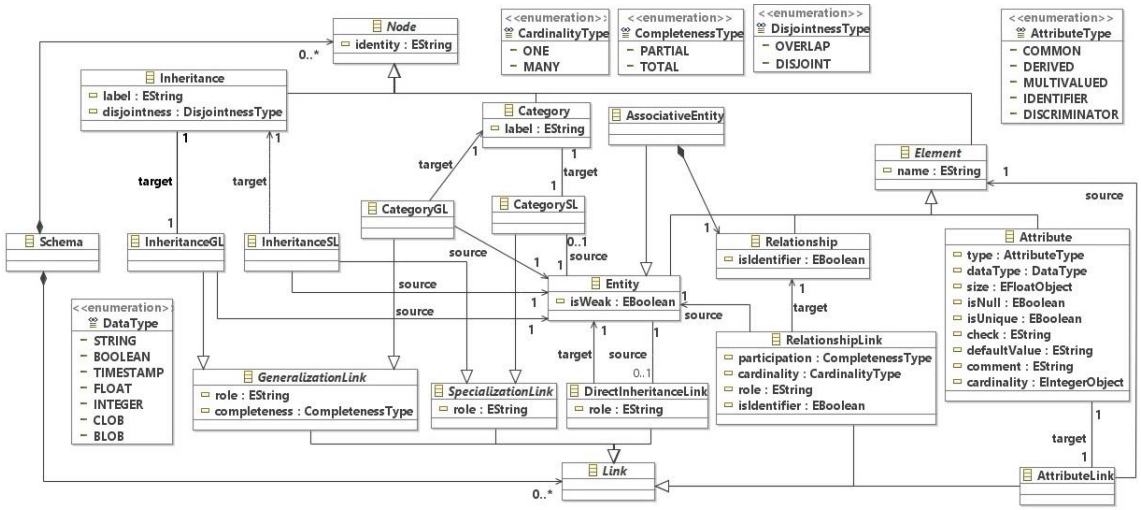

Figura 4 - Metamodelo EERMM [Fidalgo et al. 2012] [Fidalgo et al. 2013]

4.2 Implementando *feedbacks* sobre erros semânticos e avisos

Ferramentas CASE implementadas com base na OCL podem dar *feedbacks* sobre restrições semânticas ou sobre críticas relacionadas à violação de boas práticas. A EERCASE utiliza a linguagem *Epsilon Validation Language* (EVL) [Kolovos et al. 2015], uma implementação de OCL, para prover os *feedbacks* em questão. Com EVL pode-se especificar *feedbacks* para: 1) informar mensagens elaboradas; 2) impor pré-condições entre restrições ou críticas e 3) especificar uma ou mais formas de correção. A sintaxe de EVL é apresentada na Listagem 1.

```
1.  context {
2.      (guard (:expressão)|({blocoInstrução}))?
3.      (constraint|critique) {
4.          (guard (:expressão)|({blocoInstrução}))?
5.          (check (:expressão)|({blocoInstrução}))?
6.          (message (:expressão)|({blocoInstrução}))?
7.          fix {
8.              (guard (:expressão)|({blocoInstrução}))?
9.              (title (:expressão)|({blocoInstrução}))?
10.             do {blocoInstrução}
11.         }
12.     }
13. }
```

Listagem 1- Sintaxe concreta de EVL

Na Listagem 1, um *feedback* tem um contexto (cf. linha 1, *context*), o qual corresponde a um construtor da linguagem de modelagem (i.e., a uma metaclasse) e pode ter uma pré-condição (cf. linha 2, *guard*). Um *feedback* pode ser do tipo restrição (um erro sintático ou semântico) ou crítico (um aviso) (cf. linha 3, *constraint/critique*). O primeiro restringe erros semânticos e o segundo dá avisos sobre violações de recomendações. Uma restrição ou crítica pode ter uma pré-condição (cf. linha 4, guard), uma condição (cf. linha 5, *check*) e uma mensagem elaborada (cf. linha 6, *message*). Além disso, um *feedback* pode sugerir um conjunto de correções semiautomáticas (cf. linha 7, *fix*). Uma correção pode ter uma pré-condição (cf. linha 8, *guard*), um título (cf. linha 9, *title*) e ações (cf. linha 10, *do*). Um exemplo prático de como implementar um *feedback* na EERCASE usando EVL é mostrado na Listagem 2.

```
1.  context Element {
2.      constraint ElementoPossuiNome {
3.          check : self.name.isDefined()
4.          message : self.eClass().name +  ' sem nome não é permitido.
                      Informe um nome para o '+ self.eClass().name + '.'
5.      }
6.      critique NomeIniciaComCapital {
7.          guard : self.satisfies('ElementoPossuiNome')
8.          check : self.name.firstToUpperCase() <> self.name
9.          message : self.eClass().name +  '    '    +    self.name    +'    deve
                      começar com uma letra maiúscula.'
10.         fix {
11.             title : 'Renomeie para ' + self.name.firstToUpperCase()
12.             do { self.name := self.name.firstToUpperCase();}
13.         }
14.     }
15. }
```

Listagem 2- Exemplo de validação EVL

Na Listagem 2, tem-se a especificação de *feedbacks* cujo contexto refere-se a qualquer elemento a ser diagramado na EERCASE (cf. linha 1). O primeiro *feedback* especificado é sobre um erro semântico e exige que todo elemento tenha um nome (cf. linhas 2-4). O segundo é um aviso que será exibido para todo elemento que tiver o seu nome iniciado com letra minúscula e, além de informar esta violação (cf. linhas 6-9), sugere uma correção (cf.10-12).

5. Prova de conceito

Nesta seção, mostra-se como os *feedbacks* propostos são apresentados na EERCASE. Além disso, dado que o tipo "contingente-à-resposta" é o mais avançado da dimensão Resposta, os *feedbacks* da EERCASE são avaliados neste nível. Ressalta-se que 1) a dimensão Apresentação não foi avaliada, pois a escolha de um ou mais itens desta dimensão é uma questão de preferência e 2) na EERCASE, um *feedback* imediato é apresentado no instante em que uma ação equivocada ocorre e um atrasado quando o diagrama é salvo (**Arquivo** → **Salvar**) ou validado (**Editar** → **Validar**). Na Figura 5, alguns exemplos de *feedbacks* e como um erro pode ser corrigido são ilustrados. Nesta Figura, são destacadas cinco áreas relacionadas, as quais mostram que: a) um erro atrasado é simbolizado com "⛒"; b) um erro imediato é representada com "Ø"; c) um aviso é marcada com "⚠"; d) os erros e avisos são apresentados como um relatório; e e) a interface para corrigir, de forma semiautomática, um erro apontado por um *feedback*. Ressalta-se que as mensagens dos *feedbacks* são apresentadas quando o mouse é passado sobre os marcadores "⛒" ou "⚠".

A avaliação da EERCASE busca identificar os seus pontos fortes e fracos considerando o nível mais avançado da dimensão Resposta (i.e., "contingente-à-resposta"). Como resultados desta avaliação constatou-se que: 1) *feedbacks* imediatos (i.e., sobre erros sintáticos capturados exclusivamente pelo metamodelo) impedem a modelagem de erros primários/básicos sem exibir mensagens explicativas, o que pode ser desfavorável para aprendizes iniciantes, mas interessante para aprendizes iniciados, pois evita sobrecarga visual com mensagens óbvias e 2) *feedbacks* atrasados (i.e., sobre avisos e erros semânticos ou sintáticos capturados via EVL) ocorrem de forma atrasada, apresentam mensagens elaboradas sobre o erro e como corrigi-lo, incluindo a opção de corrigir o erro de forma semiautomática.

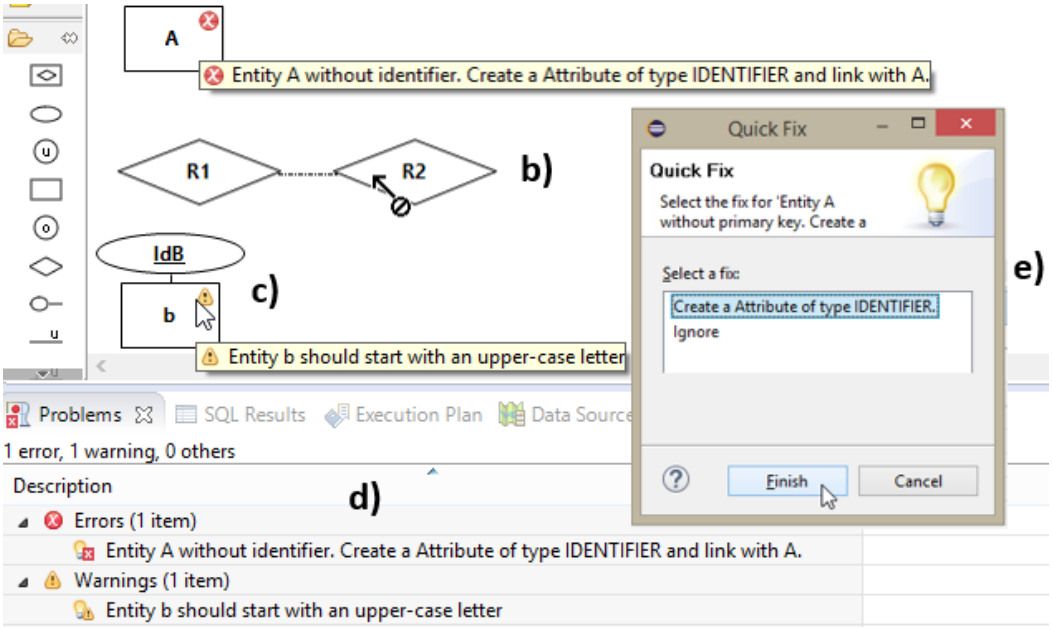

Figura 5: Exemplo de *feedbacks* de restrição, critica e correção na EERCASE

6. Considerações finais

O aprendizado ativo é uma prática de estudo que exige dos aprendizes uma postura mais participativa em sua formação, pois este método busca envolver os aprendizes no processo de aprendizagem. O uso de *feedback* é um recurso importante para exercício do aprendizado ativo, pois informa aos aprendizes o que e onde eles estão acertando ou errando, gerando uma conscientização valiosa para o processo de aprendizagem, pois confirma ou corrige o que foi realizado. Neste contexto, o uso de ferramentas CASE com bons *feedbacks* enriquece o aprendizado e prove uma visão prática, o que ajuda a consolidar o conhecimento adquirido.

Dadas as vantagens do uso educacional de ferramentas CASE e que o tema "*feedback* para ferramentas CASE" não tem recebido atenção recente da academia, este trabalho avança o estado da arte em questão, pois apresenta uma classificação de *feedbacks* para ferramentas CASE que consegue unir a teoria de *feedback* com os tipos de erros básicos em programação. Ademais, também mostra, de forma prática, como estes *feedbacks* podem ser implementados em uma ferramenta CASE baseada no paradigma MDD. Como trabalhos futuros, sugere-se usar a classificação proposta como uma guia para realizar uma análise comparativa entre diferentes ferramentas CASE, a investigação de outras tecnologias para implementar *feedbacks* em ferramentas CASE e um estudo mais aprofundado (i.e., revisão sistemática da literatura) do estado da arte de *feedbacks* e ferramentas CASE.

Referencias

ACM/IEEE (2008). Computer Science Curriculum 2008: an interim revision of CS 2001.

ACM/AIS (2010). "Curriculum Guidelines for Undergraduate Degree Programs in Information Technology".

Alves, E., Franco, N., Nascimento, A., N. Fidalgo, R. (2014). EERCASE: Uma Ferramenta para Apoiar o Estudo do Projeto Conceitual de Banco de Dados. In WCBIE 2014, p. 98-105.

Alves, F., Jaques, P. (2014). Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação. In CBIE 2014, p. 1078-1082.

Bavota, G., Gravino, C., Oliveto, R., De Lucia, A., Tortora, G., Genero, M., and Cruz-Lemus, J. (2011). Identifying the weaknesses of UML class diagrams during data model comprehension. In Model Driven Engineering Languages and Systems, volume 6981 of LNCS, pages 168–182. Springer Berlin Heidelberg.

Brambilla, M., Cabot, J., Wimmer, M. (2012). Model-Driven Software Engineering in Practice, Morgan & Claypool Publishers.

Chen, P. (976). The Entity-Relationship Model - toward a unified view of data. In ACM Transactions on Database Systems 1, pages 9–36.

Elmasri, R. and Navathe, S. (2010), Fundamentals of Database Systems, Addison-Wesley Publishing Company, 6th edition.

Fidalgo, R. D. N., Alves, E., España, S., Castro, J., and Pastor, O. (2013). Metamodeling the enhanced entity-relationship model. JIDM, 4(3):406–420.

Fidalgo, R. D. N., De Souza, E., España, S., De Castro, J., and Pastor, O. (2012). EERMM: A metamodel for the enhanced entity-relationship model. In Conceptual Modeling, volume 7532 of LNCS, pages 515–524. Springer Berlin Heidelberg.

Foundation, E. (2015). Graphical Modeling Project (GMP).

Jankowski, D. (1995): Case feedback in support of learning a systems development methodology. Journal of Information Systems Education 7(3):88-90

Jankowski, D. (1997). Computer-aided systems engineering methodology support and its effect on the output of structured analysis. Empirical Software Engineering, 2(1):11–38.

Kasprzak, J. (2005). Providing Students Feedback in Distance Education Courses. An Online Learning Magazine for UMUC Faculty.

Kolovos, D., Rose, L., García-Domínguez, A., and Paige, R. (2015). The Epsilon Book.

McKeachie, W. J., Pintrich, P. R., Lin, Y. G., & Smith, D. A. (1987). Teaching and learning in the college classroom: A review of the literature. Ann Arbor: National Center for Research to Improve Postsecondary Teaching and Learning, The University of Michigan.

Mory, E. (2004). Feedback Research Revisited. In Handbook of research on educational communications and technology, pages 745-783.

Saul, C., Runardotter, M., and Wuttke, H.-D. (2010). Towards feedback personalisation in adaptive assessment. In Sixth EDEN Research Workshop, Boook of Abstracts pp. 142-143, European Distance and E-Learning Network, Budapest.

Shute, V. (2007). Focus on formative feedback. ETS Research e Development. Princeton.

White, C. (2003). Language Learning in Distance Education. Cambridge: Cambridge University Press.

Vasilyeva, E., Puuronen, S., Pechenizkiy, M., and Rasanen, P. (2007). Feedback adaptation in web-based learning systems. In IJCEELL, 17(4/5):337–357.