6
A Ferramenta MoLIC Designer

Esse capítulo apresenta a ferramenta desenvolvida para dar suporte à construção de diagramas de interação na linguagem MoLIC, contendo ainda funcionalidades para representação de caminhos preferenciais de interação, esboços de interface e simulação da interação tal como imaginada pelo designer.

6.1
Requisitos

Os requisitos iniciais para a construção da ferramenta visaram a utilização da MoLIC em uma integração futura num ambiente de design dirigido por modelos, com entradas e saídas bem definidas, de forma a fornecer meios tanto para edição de diagramas quanto para a representação da MoLIC de uma forma processável em termos de sintaxe – para que os diagramas pudessem ser lidos por algoritmos desenvolvidos externamente. Assim, idealmente a ferramenta deve:

a. **Permitir a edição gráfica de diagramas MoLIC** – o requisito principal era o de promover a edição gráfica de diagramas na notação da MoLIC, fornecendo o acesso rápido aos elementos da linguagem e uma diagramação em uma área central, exatamente como os programas de edição gráfica;

b. **Permitir a relação entre metas e caminhos de interação** – a MoLIC permite que o designer elabore caminhos alternativos para o atingimento de uma mesma meta. Desta forma, a ferramenta deve ser capaz fornecer meios para que o designer especifique e visualize diferentes metas, relacionando-as a caminhos (sequência de elementos) diferentes;

c. **Representar o diagrama em uma notação processável** – é desejável que a ferramenta exporte algum tipo notação processável dos diagramas, para que outras ferramentas possam relacionar o que foi construído a outras notações.

d. **Validar sintaticamente o diagrama** – é desejável que a ferramenta detecte inconsistências sintáticas no diagrama e impeça o designer de construir um diagrama que não faz sentido.

6.2 Arquitetura

O *MoLIC Designer* foi desenvolvido sob a forma de plugins para a plataforma Eclipse.[1] A escolha desta plataforma parece servir ao objetivo de que tenhamos no futuro um ambiente integrado de desenvolvimento de modelos para apoiar o projeto de IHC, uma vez que sua arquitetura é voltada para IDE's (*Integrated Development Environments*).

A arquitetura da plataforma Eclipse pode ser sumarizada na Figura 6.1. Para o *MoLIC Designer*, consideram-se as duas divisões mais abaixo, classificando-o como um software *standalone* ou *Rich Client*, como são chamados pela comunidade os programas que utilizam a estrutura de plugins e interface gráfica do Eclipse. Ou seja, em softwares desse tipo geralmente não são inclusos plugins para desenvolvimento Java nem edição de texto, comuns ao Eclipse como IDE.

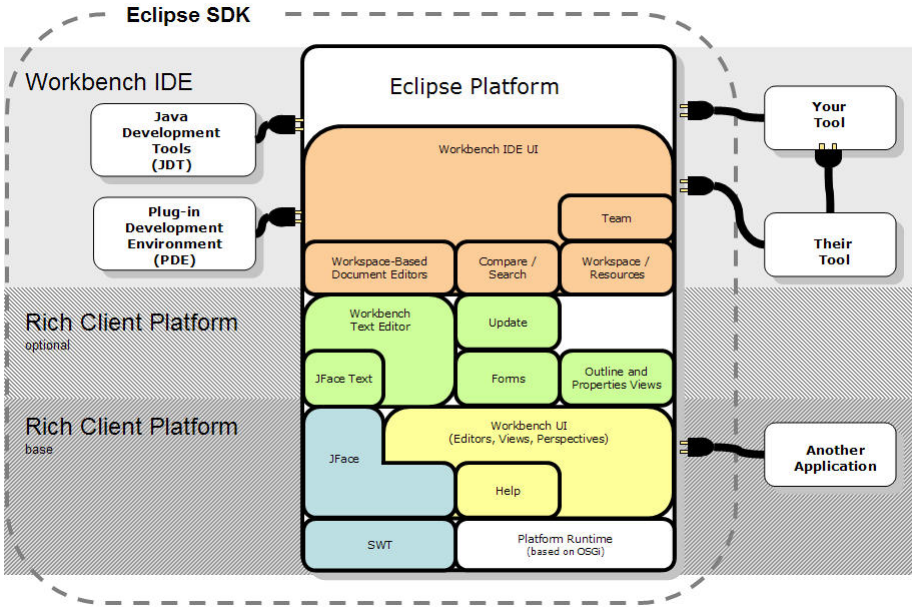

Figura 6.1: Arquitetura da plataforma Eclipse (fonte: `http://www.ibm.com/developerworks/opensource/library/os-ecjdt/`

No entanto, é importante ressaltar que, ainda que se trate de um software independente, o *MoLIC Designer* pode ser integrado a uma IDE Eclipse,

[1] http://www.eclipse.org

bastando incluir nela os plugins para edição de diagramas MoLIC. Portanto, outros plugins de editores para diferentes modelos podem ser desenvolvidos de forma separada, sem que seja necessário reescrever ou adaptar diretamente o código.

Outras vantagens quanto à utilização da plataforma Eclipse podem ser destacadas, como frameworks como o EMF (Eclipse Modeling Framework):[2] que facilita a criação e geração de código a partir de um meta-modelo Ecore, que tem uma notação parecida com MOF (Meta-Object Facility); e GEF (Graphical Editing Framework):[3] para manipulação gráfica de objetos do modelo.

A desvantagem mais evidente, no entanto, é a dificuldade de alterar a metacomunicação da plataforma Eclipse – em termos de Engenharia Semiótica – uma vez que a plataforma foi desenvolvida para dar suporte a aplicações de **qualquer** domínio, a comunicação dos designers do ambiente sobre todas as funcionalidades disponíveis (e eventualmente não desejadas) é bastante evidente nas aplicações finais. Em outras palavras, há suporte para que o "desenvolvedor-final" consiga construir o seu discurso para os seus usuários, porém há uma grande dificuldade para que a sua aplicação não converse com o usuário sobre o que ele não deseja.

6.3 Edição de Diagramas

O *MoLIC Designer* apresenta funcionalidades básicas para criação de arquivos de diagramas, e o editor conta com uma área de desenho e uma paleta com os elementos Scene, Opening Point, Closing Point, System Process, Ubiquitous Access, Utterance e Breakdown (Fig. 6.2). Os elementos podem ser arrastados a partir da paleta (através de clique e arraste para a área de edição) ou selecionados na paleta e inseridos ao se fazer clique na área de edição.

A ferramenta contém várias *visões*, que são pequenos espaços com propósitos específicos e que podem ser posicionadas livremente dentro no espaço da ferramenta. Uma visão pode servir, por exemplo, como exibição de propriedades do elemento selecionado no diagrama. Na Figura 6.2, dentro da área de desenho, a cena Search está selecionada, e em *Properties*, na parte inferior esquerda são exibidas suas propriedades. Na lateral direita inferior (*Outline*) se pode ter uma visão geral do diagrama, o que auxilia a navegação em diagramas muito grandes.

[2]http://www.eclipse.org/emf
[3]http://www.eclipse.org/gef

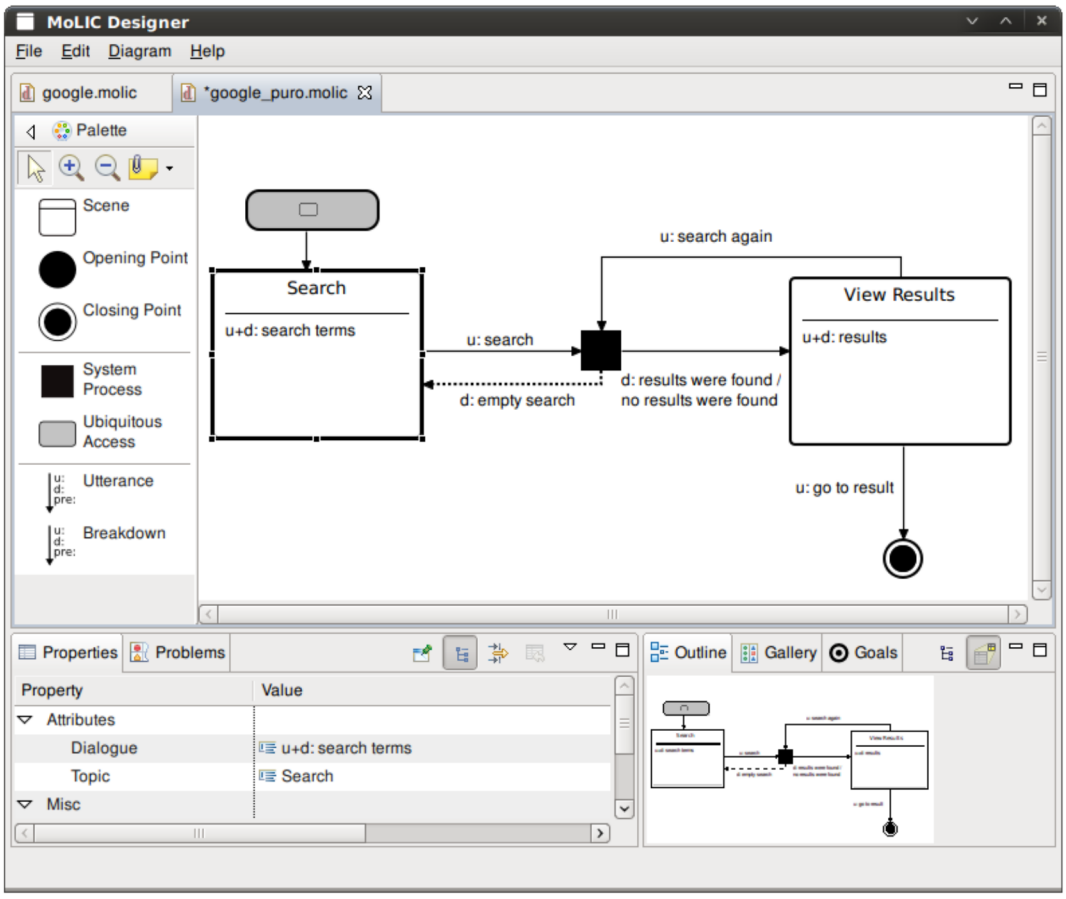

Figura 6.2: A ferramenta *MoLIC Designer*

Em comparação com ferramentas puramente gráficas para o desenho de diagramas, o *MoLIC Designer* tem a vantagem de fornecer uma notação textual, em XMI (*XML Metadata Interchange*), em vez de somente desenhos, o que pode propiciar no futuro a integração com outras linguagens, em outros ambientes.

6.4 Representação de Metas

Ao projetar um sistema, o designer escolhe alguns caminhos como preferenciais para o atingimento de determinadas metas. A MoLIC permite a especificação de todas as metas através da lista de metas [Barbosa e Paula, 2003], o que não é feito pelo *MoLIC Designer*.

Araujo (2008) estendeu a notação para representar o início e fim de conversas sobre metas preferenciais. Porém, para preservar a legibilidade do

diagrama, foi preferível permitir ao designer que visualizasse os caminhos preferenciais como um *overlay*, como mostrado na Figura 6.3.

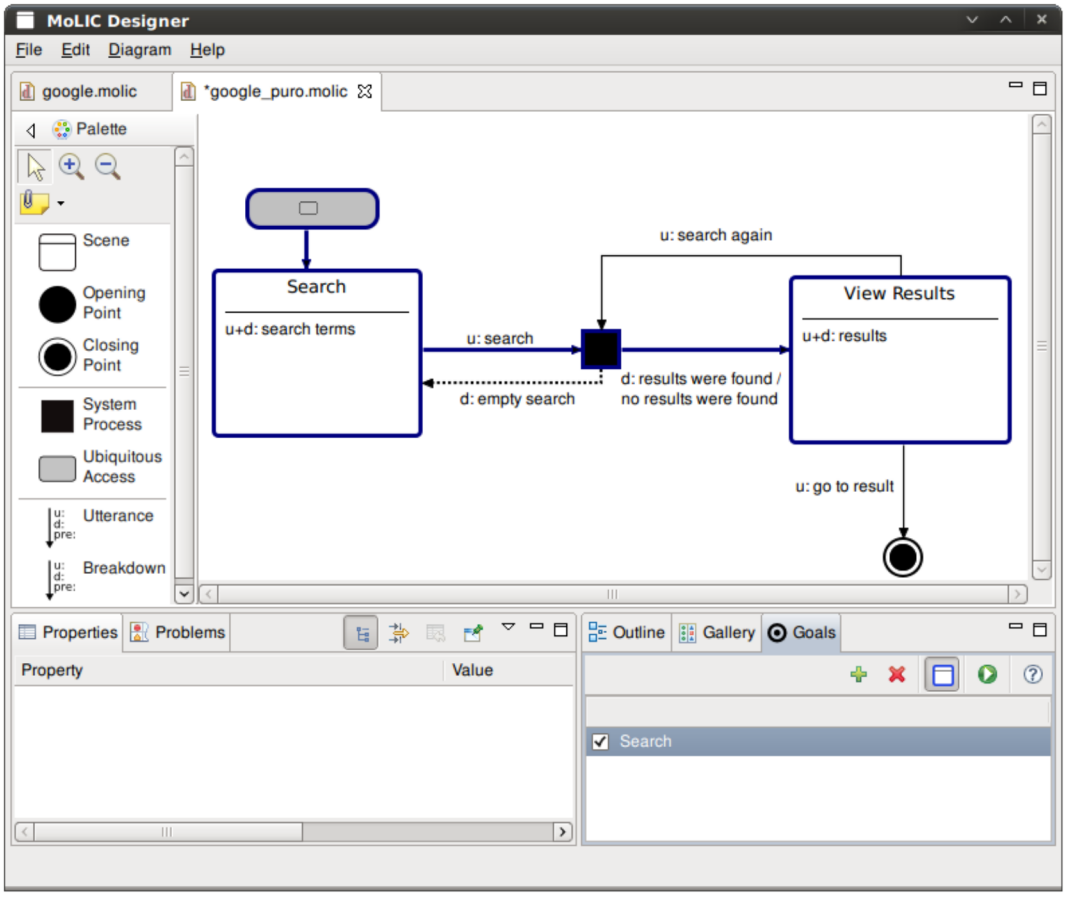

Figura 6.3: Realce gráfico (linhas mais espessas) da meta selecionada

Desta forma, o designer pode alternar entre a visualização do diagrama com/sem as metas. A Figura 6.3 mostra uma pequena área na parte inferior direita, onde é possível adicionar novas metas, associar elementos do diagrama às metas e visualizá-las em destaque no diagrama. Quando uma meta é selecionada, uma área é destacada em torno dos elementos a ela associados.

Além de permitir o destaque, facilitando a comparação de duas ou mais metas, a especificação dessas sequências de elementos permitirá que, por exemplo, apenas os caminhos em foco no momento possam ser testados em um eventual simulador da interação.

6.5 Esboços de Interface

É importante ressaltar que tanto a ferramenta quanto a própria extensão da linguagem MoLIC não visam a especificação de sistemas, mas sim a

discussão em torno do artefato sendo projetado. Dessa maneira, os esboços de interface podem ser feitos com tanto detalhe quanto se queira, sendo preferencialmente em um baixo nível.

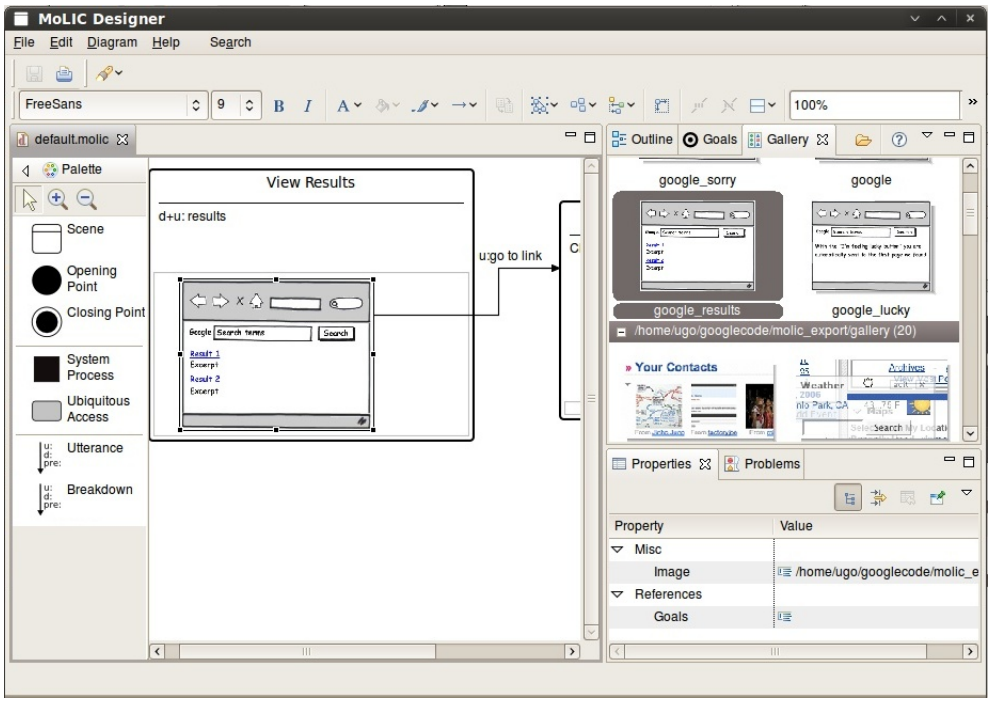

Figura 6.4: MoLIC Designer exibindo a adição de esboços de interface nas cenas

De modo a dar suporte à representação de esboços de interface, o *MoLIC Designer* apresenta a visão *Gallery* onde é possível adicionar os esboços que posteriormente podem ser arrastados para dentro das cenas. Esses esboços devem ser feitos utilizando-se outros softwares de desenho, como por exemplo o Balsamiq.[4]

A Figura 6.4 mostra a visão *Gallery* exibindo alguns esboços de tela desenhados para o cenário do estudo de caso, exportados como arquivos de imagem e importados na ferramenta. O diagrama ao centro exibe uma cena contendo um dos esboços.

Os esboços de tela podem ser adicionados e conectados respectivamente com falas do usuário, que têm sua origem em esboços de tela; e falas do designer, que têm o seu destino nos esboços.

As imagens dentro da cena podem ser re-dimensionadas para ficarem tão pequenas quanto o designer queira, podendo ser visualizadas com a ferramenta de *zoom*, se desejar. Essa característica pode evoluir para represen-

[4]http://www.balsamiq.com

tar o diagrama com um conceito semelhante ao de *Zoomable User Interfaces* [Beard e Walker, 1990].

6.6 Geração de Protótipos

Uma vez que os esboços de tela estejam ligados a falas de transição, que consequentemente ligam-se a outras cenas e esboços, é possível se imaginar a simulação da interação sendo feita sob a forma de telas em sequência, como um protótipo guiado pelas falas de *u:* e *d:*. Assim, o simulador exibiria sequencialmente os esboços contidos nas cenas, reagindo às ações de quem o utiliza, mais ou menos como um *storyline* interativo, ou um protótipo não-funcional.

Um módulo de criação de simulações foi desenvolvido de forma preliminar, onde a sequência de telas é exportada sob a forma de páginas HTML, que exibem uma imagem por vez. Na ferramenta, o designer pode especificar áreas sensíveis ao clique, em cada esboço, que quando acionadas representam a fala do usuário, podendo levar a uma outra cena (um outro esboço) ou a um processo. O designer ou o usuário podem navegar no protótipo, escolhendo o próximo passo a partir de cada tela.

As áreas sensíveis ao clique são especificadas em cada esboço como mapas de imagem HTML, através de um atributo correspondente. Neste mapa podem ser especificadas áreas especiais que, quando clicadas, levam a um endereço indicado em um atributo. A Figura 6.5 mostra um exemplo deste mapeamento para um esboço de tela relativa à busca no *Google*.

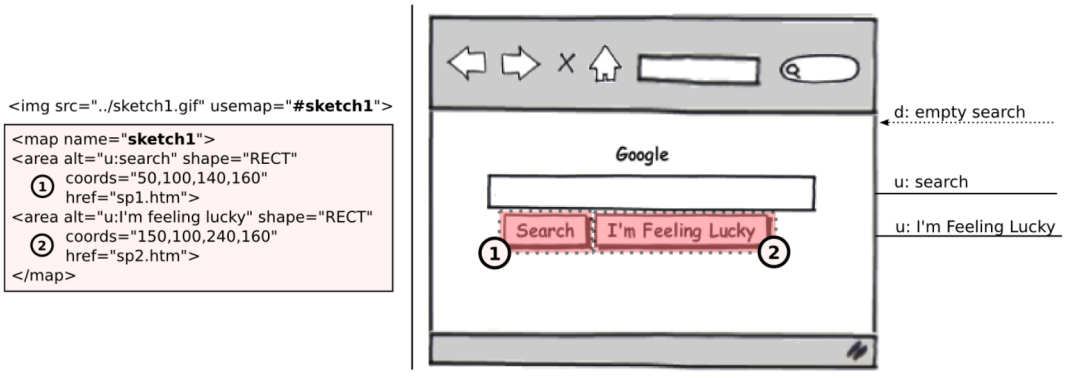

Figura 6.5: Mapa da imagem, utilizado para associar regiões da imagem a falas.

Assim é possível se fazer, por exemplo, o mapeamento do botão *Search* com a fala *u:search*, o que leva a um processo e à possibilidade de três falas do designer em resposta, segundo o cenário de *Busca na Web*: *d: empty search* –

que leva o usuário de volta para a mesma cena, na mesma tela; *d: results were found* – que leva o usuário para a cena de exibição de resultados, e exibe uma tela com os resultados; e *d: no results* – que leva o usuário para a mesma cena de exibição de resultados, numa tela indicando que não foram encontrados resultados.

Caso alguma fala leve para um processo com várias falas (resultados) possíveis, uma página é exibida para que seja escolhida uma delas, o que exibe a próxima tela, e assim por diante. Esta abordagem pode evoluir para a utilização da técnica Wizard of Oz [Kelley, 1984], onde o usuário interage, sem saber, com um software "oco" – sem lógica ou banco de dados. No lugar, está uma outra pessoa que escolhe qual tela será exibida para o usuário, reagindo ao que foi informado por ele.

No caso da interação simulada com esboços, o designer pode fazer o papel de sistema, guiando o usuário por caminhos que ele irá escolher, através das falas do sistema. Evidentemente, essa abordagem irá requerer a representação de elementos de interface "reais", em vez de esboços, onde o usuário possa preencher caixas de texto, por exemplo.

6.7 Representação Lógica

O objetivo da ferramenta é permitir o processamento de diagramas MoLIC em implementações diversas, de forma que a utilização dos diagramas não dependa da própria ferramenta. Dessa forma, o *MoLIC Designer* foi projetado de modo a processar um arquivo aberto, que possa ser lido (e escrito) por rotinas desenvolvidas em outras linguagens ou outras plataformas.

A representação em XMI é apresentada na listagem de código 6.1, representando o diagrama da figura 6.2. Neste arquivo (que possui a extensão *.molic*) são escritos e lidos os elementos do diagrama construído, de forma que possa também ser processado por ferramentas externas.

A estrutura do metamodelo está refletida no arquivo *.molic*, de modo que, se o arquivo for alterado por alguma ferramenta externa (adicionando novas cenas e falas, por exemplo), o *MoLIC Designer* detecta a mudança e atualiza o diagrama exibido. Desta forma, a ferramenta também pode servir como um visualizador de diagramas feitos em outros programas, bastando que o programa escreva o arquivo com uma estrutura semelhante à do código 6.1.

Código 6.1: Código do diagrama MoLIC

```
<xmi:XMI xmi:version='2.0' xmlns:xmi='http://www.omg.org/XMI'
    xmlns:molic='molic' xmlns:notation='http://www.eclipse.org/gmf/
    runtime/1.0.2/notation'>
 <molic:Diagram xmi:id='d1'>
   <!-- ELEMENTS -->
   <element xmi:type='molic:Scene' xmi:id='sc1' topic='Search'
       dialogue='u+d: search terms'/>
   <element xmi:type='molic:Scene' xmi:id='sc2' topic='View
       Results' dialogue='u+d: results'/>
   <element xmi:type='molic:SystemProcess' xmi:id='sp1'/>
   <element xmi:type='molic:ClosingPoint' xmi:id='cp1'/>
   <element xmi:type='molic:UbiquitousAccess' xmi:id='ua1'/>

   <!-- UTTERANCES AND BREAKDOWNS -->
   <utterance xmi:type='molic:Utterance' xmi:id='u1' label='u:
       search' source='sc1' target='sp1'/>
   <utterance xmi:type='molic:Utterance' xmi:id='u2' label='d:
       results were found /&#xA;no results were found' source='sp1'
        target='sc2'/>
   <utterance xmi:type='molic:Utterance' xmi:id='u3' label='u:
       search again' source='sc2' target='sp1'/>
   <utterance xmi:type='molic:Utterance' xmi:id='u4' label='u: go
       to result' source='sc2' target='cp1'/>
   <utterance xmi:type='molic:BRTUtterance' xmi:id='u5' label='d:
       empty search' source='sp1' target='sc1'/>
   <utterance xmi:type='molic:Utterance' xmi:id='u6' source='ua1'
       target='sc1'/>

 </molic:Diagram>
</xmi:XMI>
```