FERRAMENTA DE AUXÍLIO AO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE INTEGRANDO TECNOLOGIAS OTIMIZADORAS

Acadêmico: Roger Anderson Schmidt

Orientador : Marcel Hugo

Supervisor : Ricardo de Freitas Becker

Empresa : Mult Sistemas Ltda

Roteiro da Apresentação

- Introdução
- Motivação
- Objetivos
- Tecnologias Otimizadoras
- Compiladores
- Especificação do Projeto
- Conclusão
- Extensões

Introdução

- Tecnologias Otimizadoras
  - Combinação sinérgica
- Ferramenta constrói um interpretador
  - Linguagem de comandos
- Mult Sistemas
  - Perfil da empresa
  - Metodologia de Desenvolvimento

Motivação

- Como tornar o conhecimento resistente às constantes evoluções tecnológicas no desenvolvimento de software ?

Objetivos

- Desenvolver uma ferramenta CASE que gere código-fonte a partir das definições de um repositório de dados complementar a ferramenta de modelagem de dados *CA ERwin*;
- Demonstrar a integração de tecnologias otimizadoras.

Tecnologias Otimizadoras

- A revolução industrial do software representa o movimento em direção a era da aplicação em massa da componentização;
- Uma revolução industrial do software não resulta de nenhuma das tecnologias otimizadoras isoladamente, mas de todas integradas em uma estrutura orientada a objetos;

Tecnologias Otimizadoras Exemplos

- Ferramentas CASE
  - Separam o projeto do programa aplicativo, da implementação do código;
- Programação visual
  - Apresenta um ambiente mais intuitivo para o ser humano;
- Geradores de código
  - Reduzem o tempo de desenvolvimento e aumentam a confiabilidade do código gerado;

Tecnologias Otimizadoras Exemplos

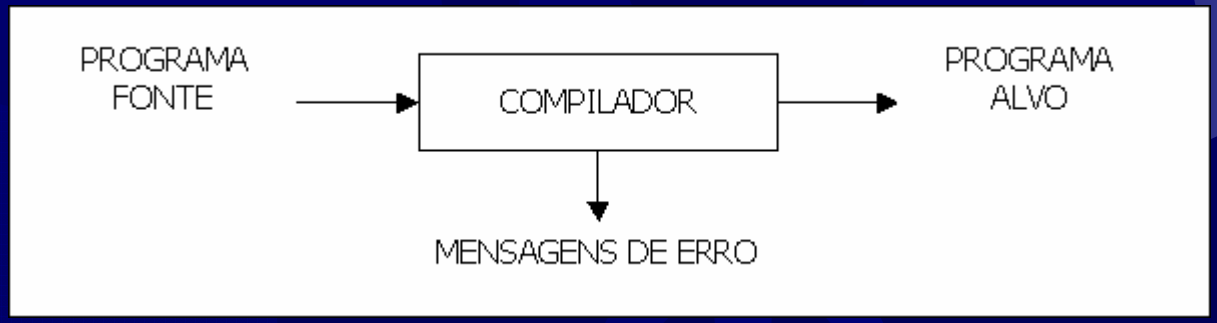
- Repositório de dados
  - Banco de dados utilizado pelas ferramentas CASE;
- Engenharia da Informação
  - Conjunto de técnicas interligadas para desenvolvimento de sistemas de informação de uma empresa;
- Bancos de dados orientados a objetos
  - Suportam o paradigma da orientação da objeto, armazenando operações além de dados;

Tecnologias Otimizadoras Exemplos

- Linguagens não-procedimentais
  - Abstraem o procedimento detalhado da execução;
- Tecnologia cliente x servidor
  - Paradigma ou modelo para interação entre processos de software em execução concorrente;
- Biblioteca de classes
  - Viabilizam o reaproveitamento através de um conjunto de classes reaproveitáveis;
- Análise e projeto orientados a objeto
  - Unificam o modelo conceitual, reduzindo a distância semântica.

Compilador - Conceito

- Software que lê um programa escrito numa linguagem – a linguagem fonte – e o traduz em programa equivalente numa outra linguagem – a linguagem alvo;
- Executa uma série de funções auxiliares, sendo que a mais importante é a detecção de erros.



Compilador – Terminologia Básica

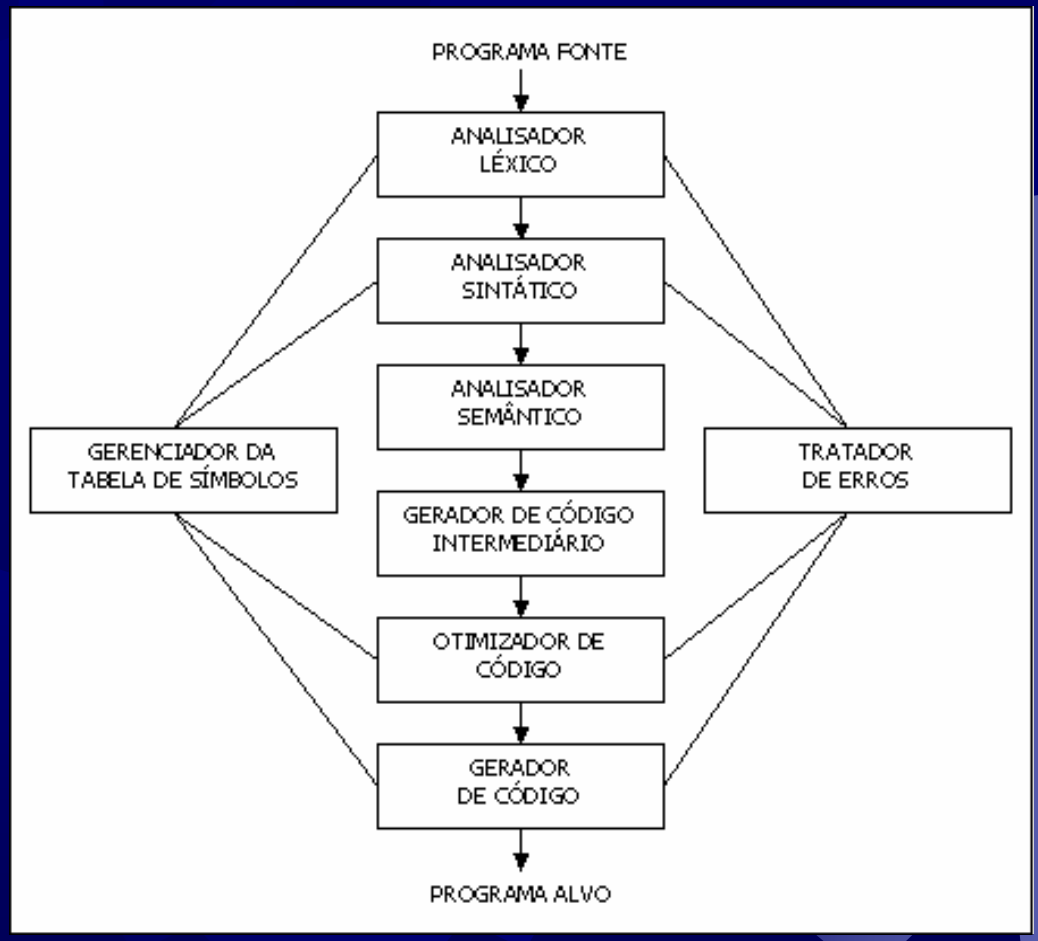
- *Token:*
  - seqüência de caracteres com um significado coletivo. É a menor unidade de informação de uma linguagem;
- *Padrão:*
  - regra que descreve o conjunto de lexemas que podem representar um *token* particular nos programas-fonte;
- *Lexema:*
  - conjunto de caracteres no programa-fonte que é reconhecido pelo padrão de algum *token*.

Compilador – Terminologia Básica

TOKEN	LEXEMAS EXEMPLO	DESCRIÇÃO INFORMAL DO PADRÃO
const	`const`	constante
if	`if`	comando condicional
relação	`<, <=, =, <>, >, >=`	operadores relacionais
id	`pi, contador, D2`	letra seguida por letras e/ou dígitos
num	`3.1416, 0, 6.02E23`	qualquer constante numérica
literal	"conteúdo da memória"	quaisquer caracteres entre aspas, exceto aspas

Compilador - Fases

- O processo de compilação é dividido em análise e síntese;
- Conceitualmente, um compilador opera em fases, cada uma das quais transforma o programa-fonte de uma representação para outra.



Compilador – Fases de Análise

- Análise léxica lê o fluxo de caracteres de entrada e produz uma seqüência de *tokens*;
- Análise sintática obtém cadeia de *tokens* e verifica se pode ser gerada pela gramática;
- Semântica consiste no exato significado que uma sentença assume dentro do texto;

Compilador – Fases de Síntese

- Gerador de código intermediário produz uma representação intermediária explícita do programa-fonte;
- Otimizador de código tenta melhorar o código intermediário, resultando um código de máquina mais rápido em tempo de execução;
- Gerador de código traduz as instruções intermediárias em uma seqüência de instruções de máquina.

Compilador – Componentes

- Gerenciador da tabela de símbolos manipula a estrutura de dados da tabela de símbolos;
- Tratador de erros registra os erros, possivelmente sem interrupção do processo de compilação.

Interpretador - Conceito

- Consiste em uma ferramenta de software que realiza as operações especificadas pelo programa fonte, ao invés de produzir um programa alvo como resultado da tradução;
- Pode ser classificado em direto e processadores de linguagem.

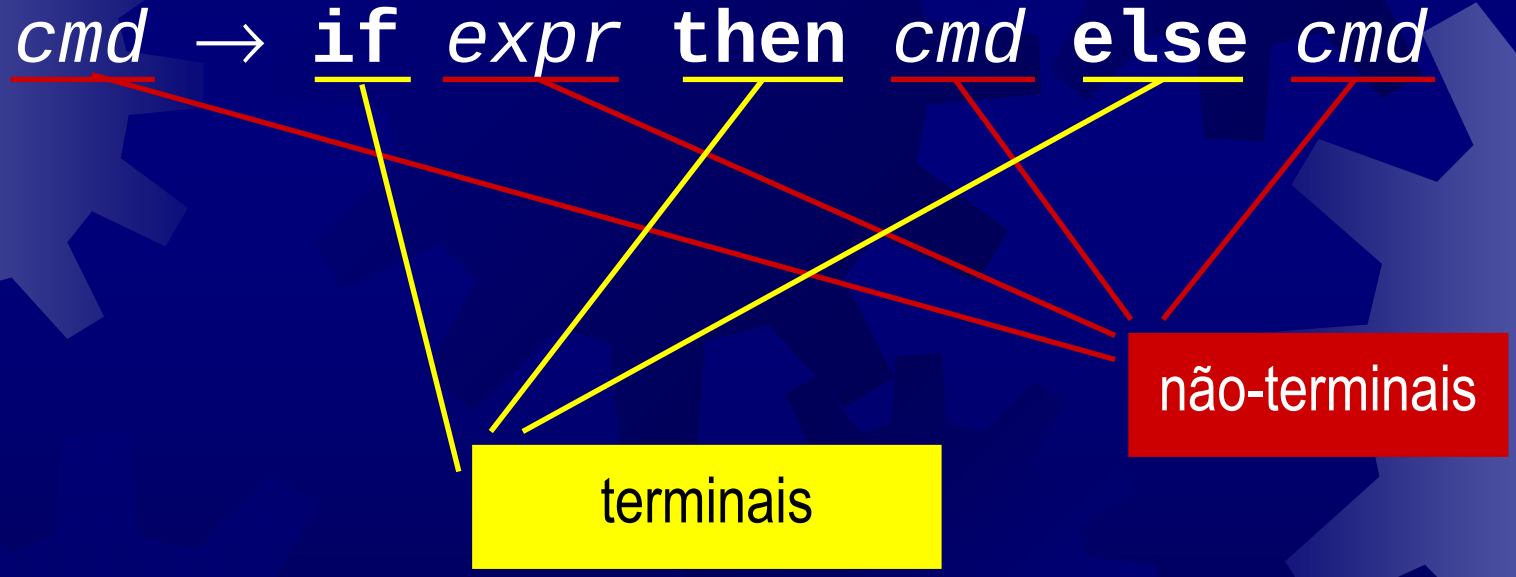
Gramática x Linguagem

- Uma gramática corresponde ao conjunto de leis de formação que definem o modo de formar textos em uma linguagem;
- Uma linguagem consiste no conjunto de todos os textos que podem ser gerados a partir da gramática que a define;
- Numa linguagem de comandos, cada operador é usualmente uma invocação de uma rotina complexa.

Gramática Livre de Contexto Componentes

- Terminais:
  - Símbolos básicos a partir dos quais as cadeias são formadas;
- Não-terminais:
  - variáveis sintáticas que geram novas produções;
- Símbolo de partida:
  - Não-terminal que denota o conjunto de cadeias da linguagem;
- Conjunto de produções:
  - Forma pela qual os terminais e não-terminais podem ser combinados para formar cadeias.

Gramática Livre de Contexto Exemplo

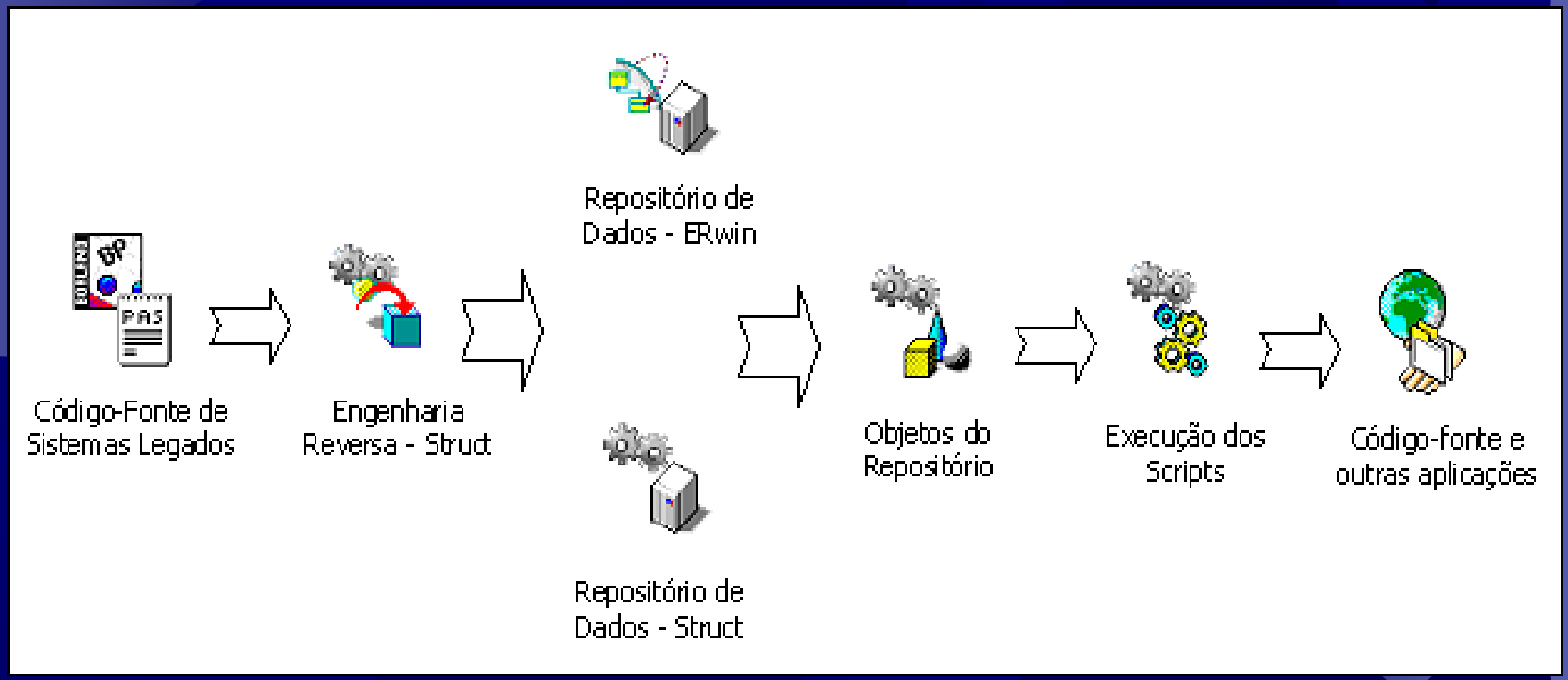


BNF (*Backus Naur Form*)

- Uma metalinguagem é uma linguagem utilizada como forma de representação ou definição de outras linguagens;
- A BNF é uma metalinguagem comumente utilizada para especificação da sintaxe de linguagens de programação;
- Notação recursiva de formalização da sintaxe de linguagens através de produções gramaticais, permitindo assim a criação de dispositivos de geração de sentenças.

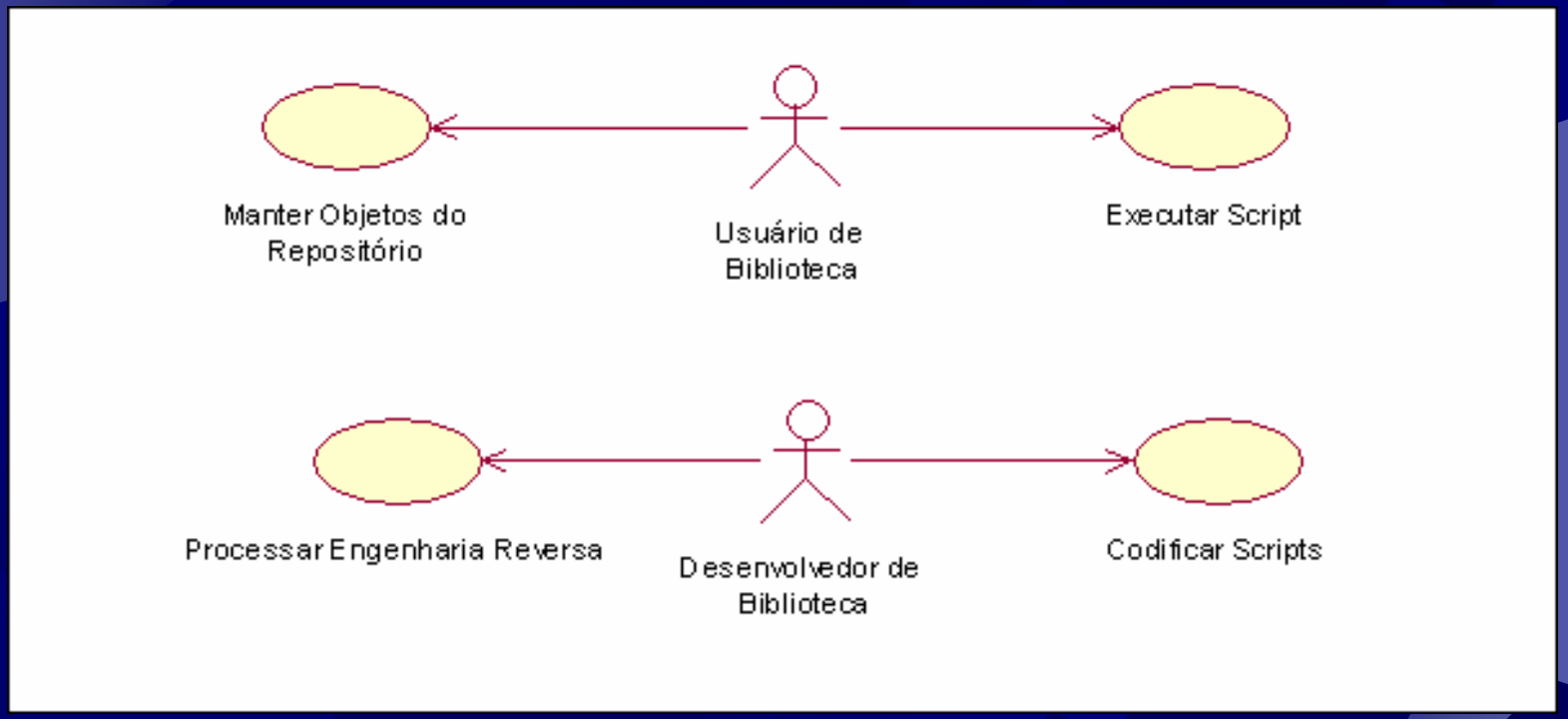
Especificação do Projeto

- Esquema funcional



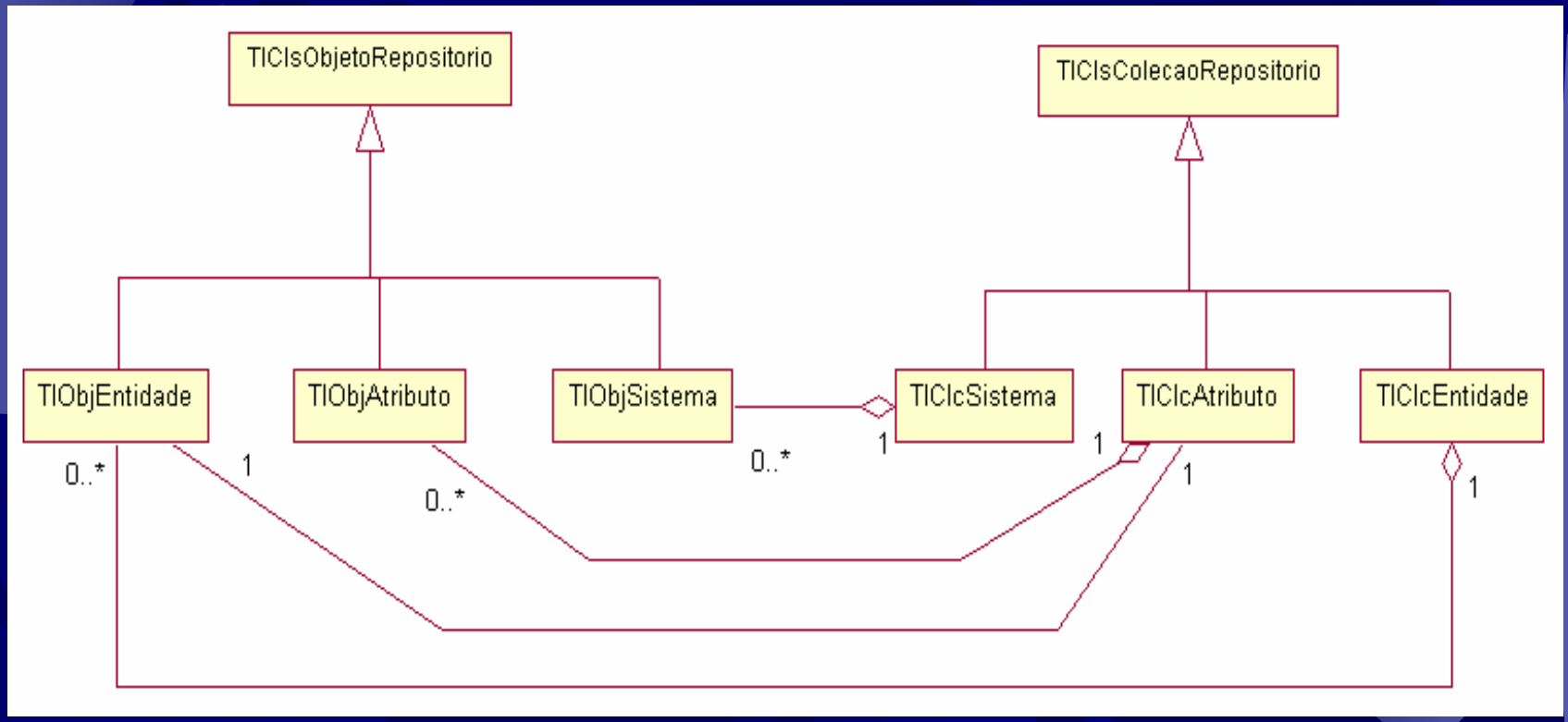
Especificação do Projeto

- Diagrama de casos de uso



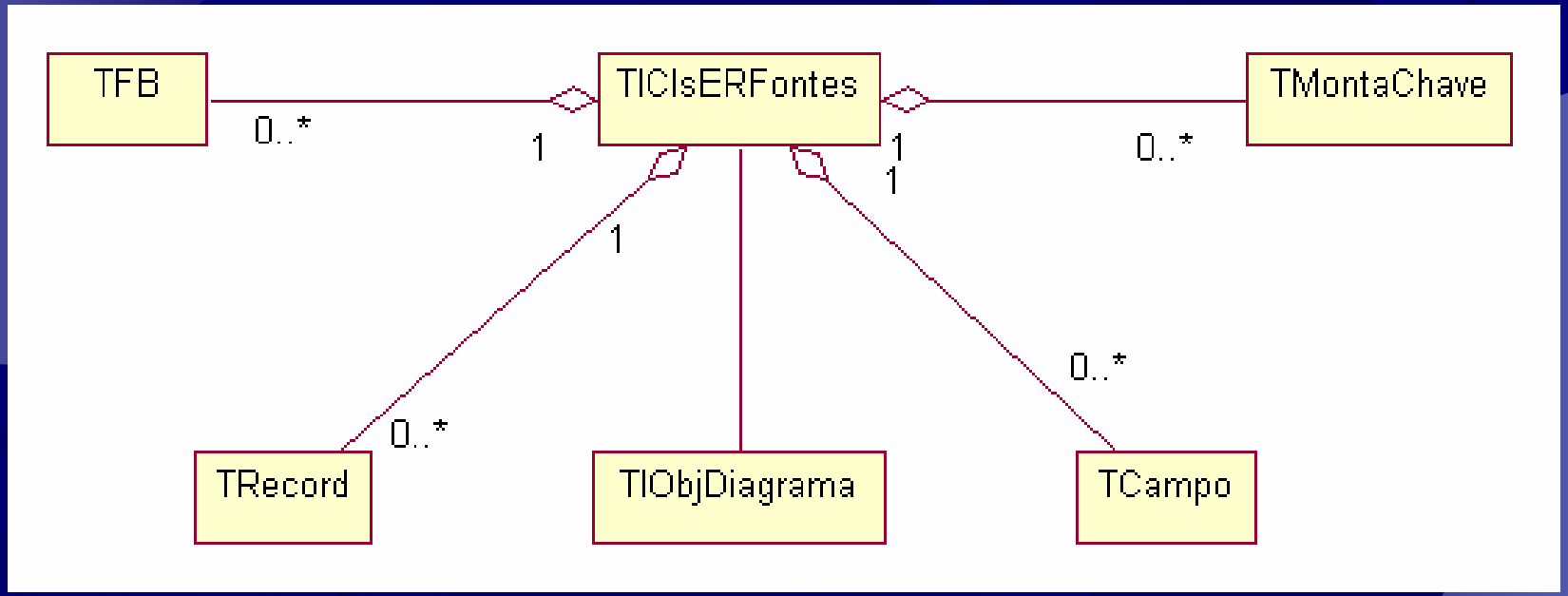
Especificação do Projeto

- Diagrama de Classes – Coordenador de Repositório



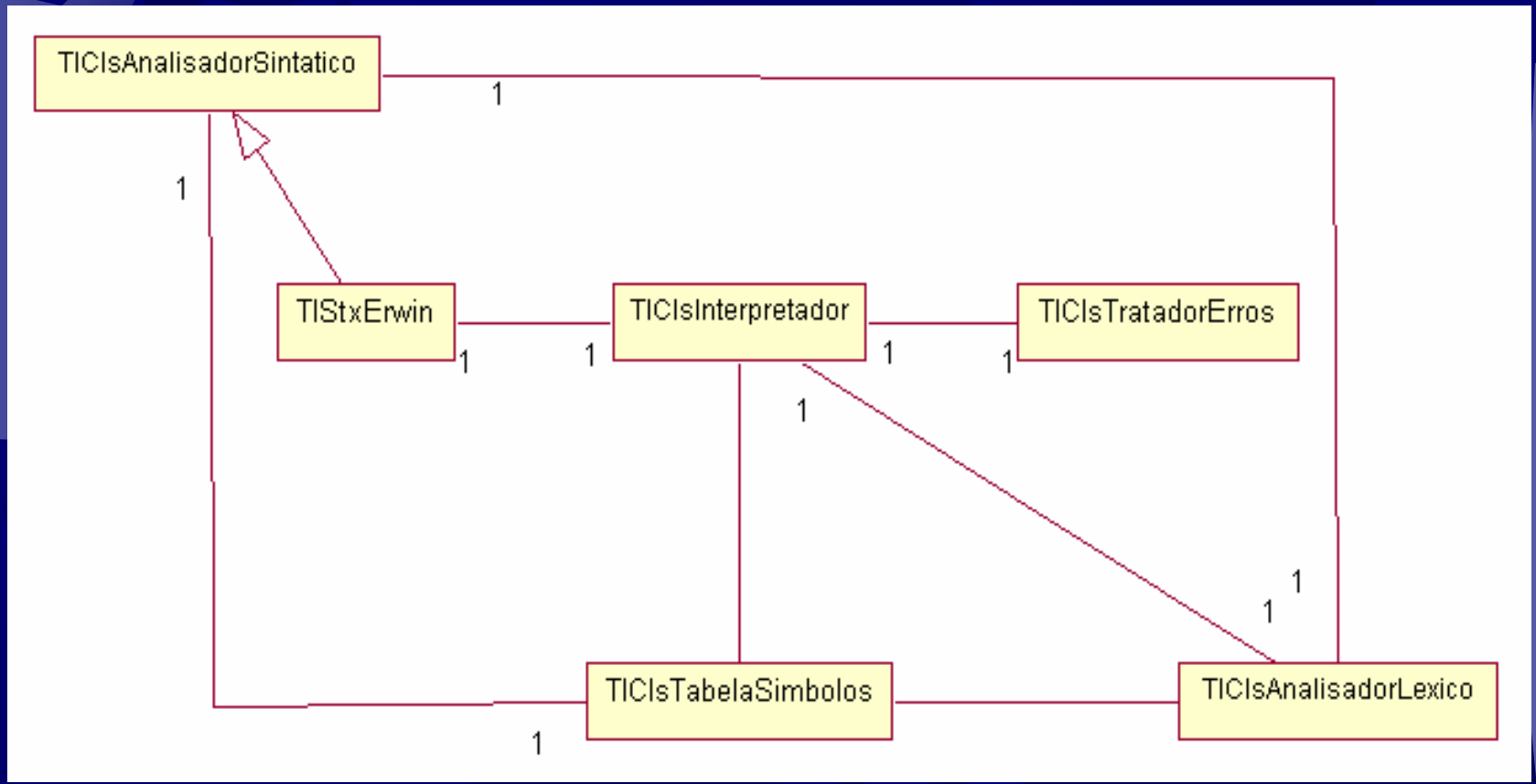
Especificação do Projeto

- Diagrama de Classes – Engenharia Reversa



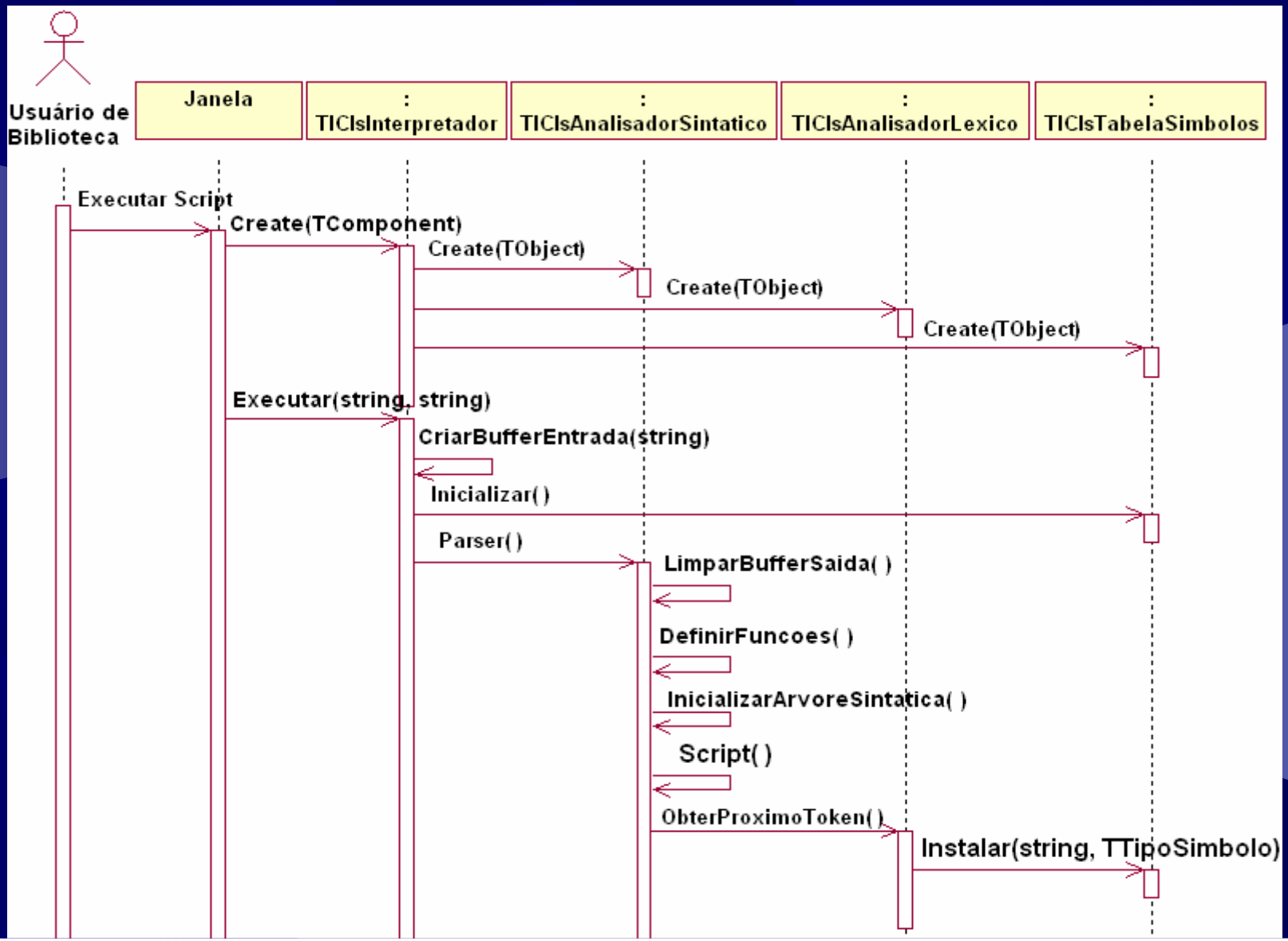
Especificação do Projeto

- Diagrama de Classes – Interpretador



Especificação do Projeto

- Diagrama de Seqüência – Executar *Script*



Especificação do Projeto

- Gramática da Linguagem LCS (Linguagem de Comandos Struct)

```
                    script -> script id;
      enunciado_composto -> Inicio
                              enunciados_opcionais
                            Fim
enunciados_opcionais -> lista_de_enunciados | E
lista_de_enunciados' -> ;
                        enunciado
                        lista_de_enunciados'
                        | E
```

Especificação do Projeto

- Gramática da Linguagem LCS (Linguagem de Comandos Struct)

```
        enunciado -> enunciado_composto
                   | Se expressao Entao enunciado enunciado'
                   | Enquanto expressao Faca enunciado
                   | chamada_de_funcao
       enunciado' -> Senao enunciado | E
 chamada_de_funcao -> id chamada_de_funcao'
chamada_de_funcao' -> (lista_de_expressoes) | E
lista_de_expressoes -> expressao
                       lista_de_expressoes'
```

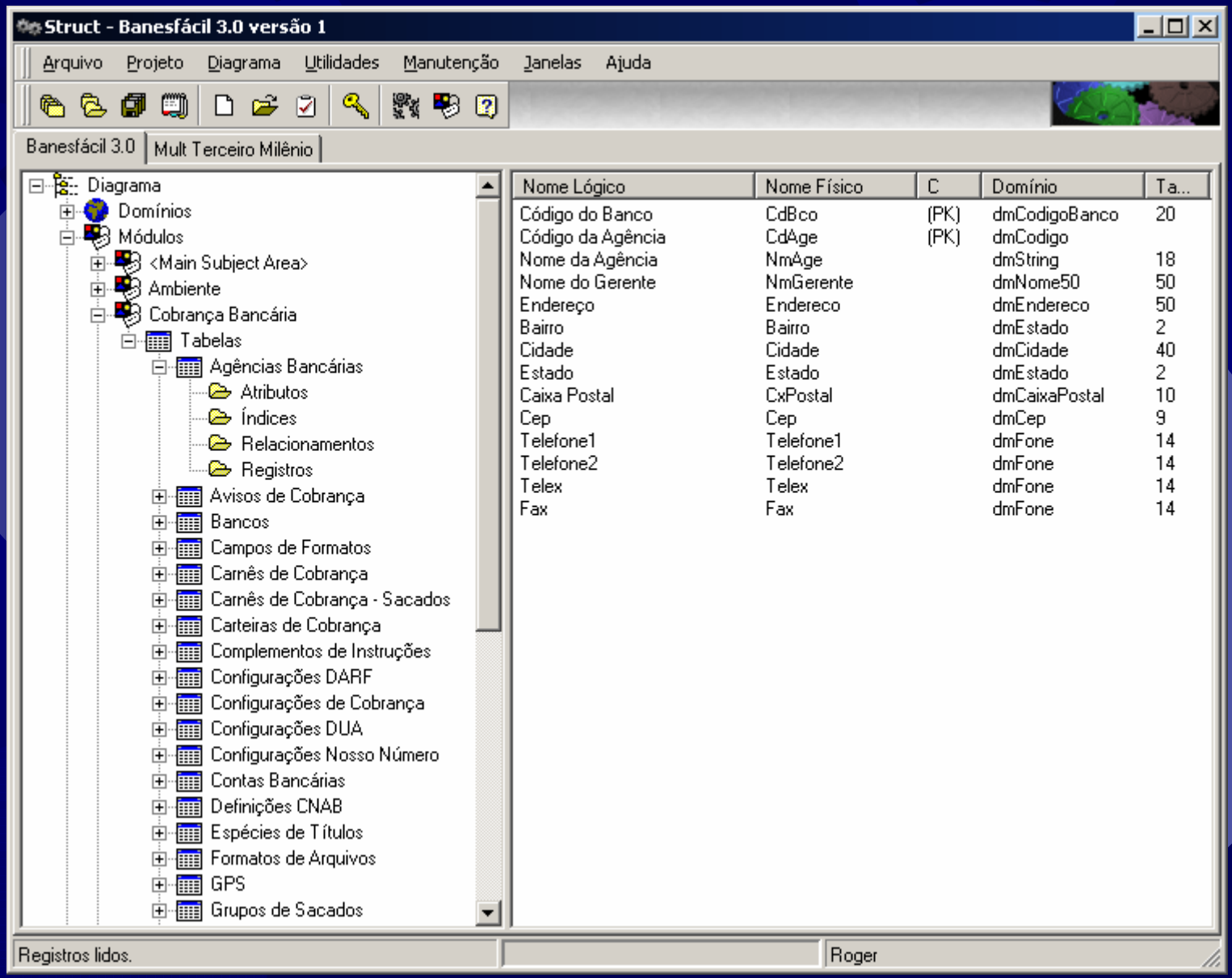
Especificação do Projeto

- Gramática da Linguagem LCS (Linguagem de Comandos Struct)

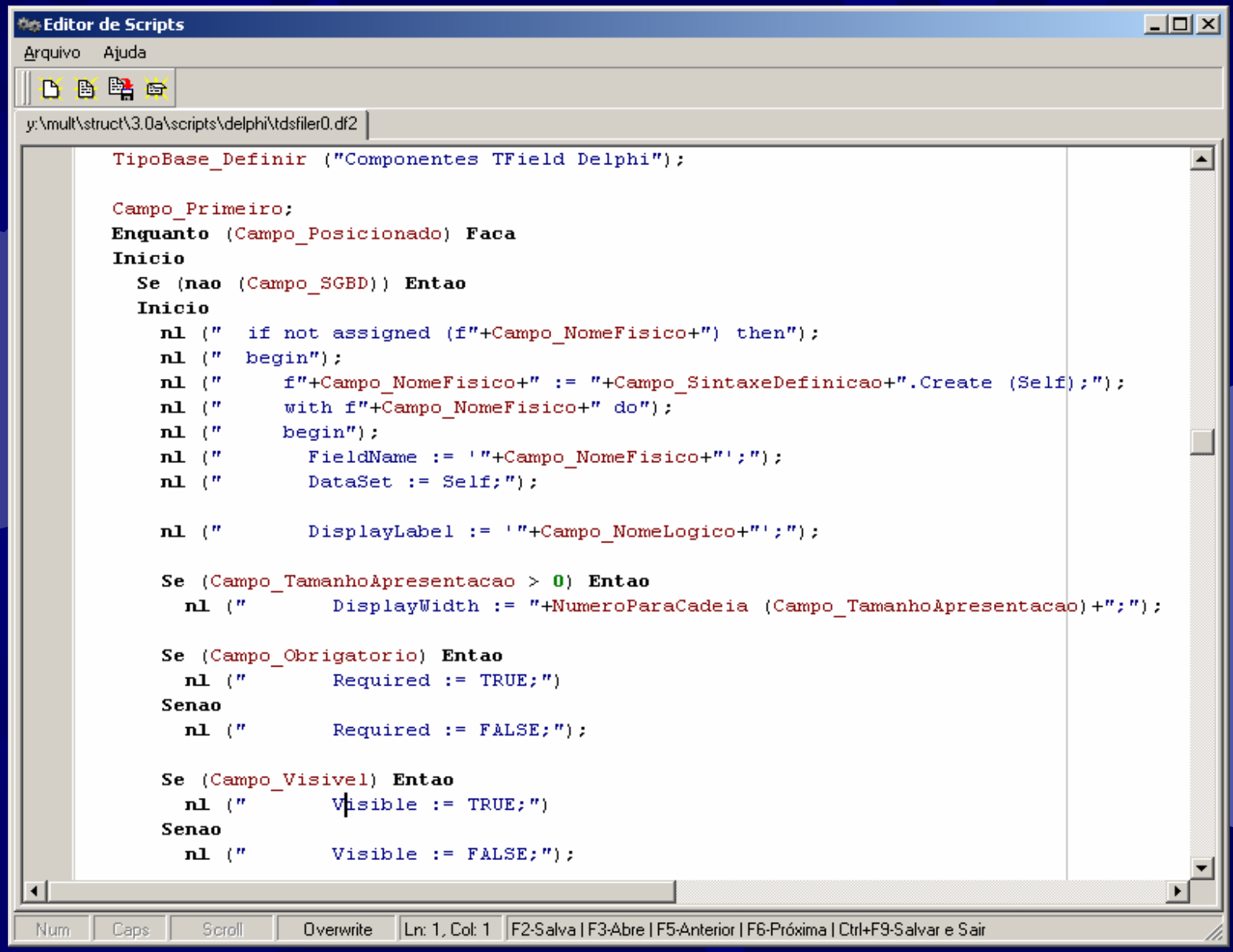
```
lista_de_expressoes -> ,
                       expressao
                       lista_de_expressoes'
                       | E
          expressao -> CE
```

Produções do Analisador de Expressões

Janela Principal



Editor de *Scripts*



Conclusão

- Os objetivos propostos foram atingidos;
- Importância do emprego das técnicas de compiladores para guiar o desenvolvimento de recursos afins;
- Dificuldade de adequação do paradigma relacional x orientado a objetos;
- Grande potencial obtido a partir da combinação das tecnologias de automação do processo de desenvolvimento.

Extensão

- Agregar ao repositório objetos das camadas de lógica e apresentação;
- Adicionar recursos e otimizar aspectos do interpretador direto;
- Integrar ao repositório de dados da ferramenta CASE *Rational Rose.*