Protótipo de uma Linguagem de Programação de Script para elaboração de conteúdo dinâmico na WWW

Fabio Eduardo Tomaz
Orientando
José Roque Voltolini da Silva
Orientador

Roteiro

- Introdução
- Objetivos
- Funcionamento de um Script
- Especificação HTTP
- Definição da Linguagem
- Estrutura Interpretador
- Desenvolvimento
  - Requisitos
  - Especificação
  - Implementação
- Conclusões

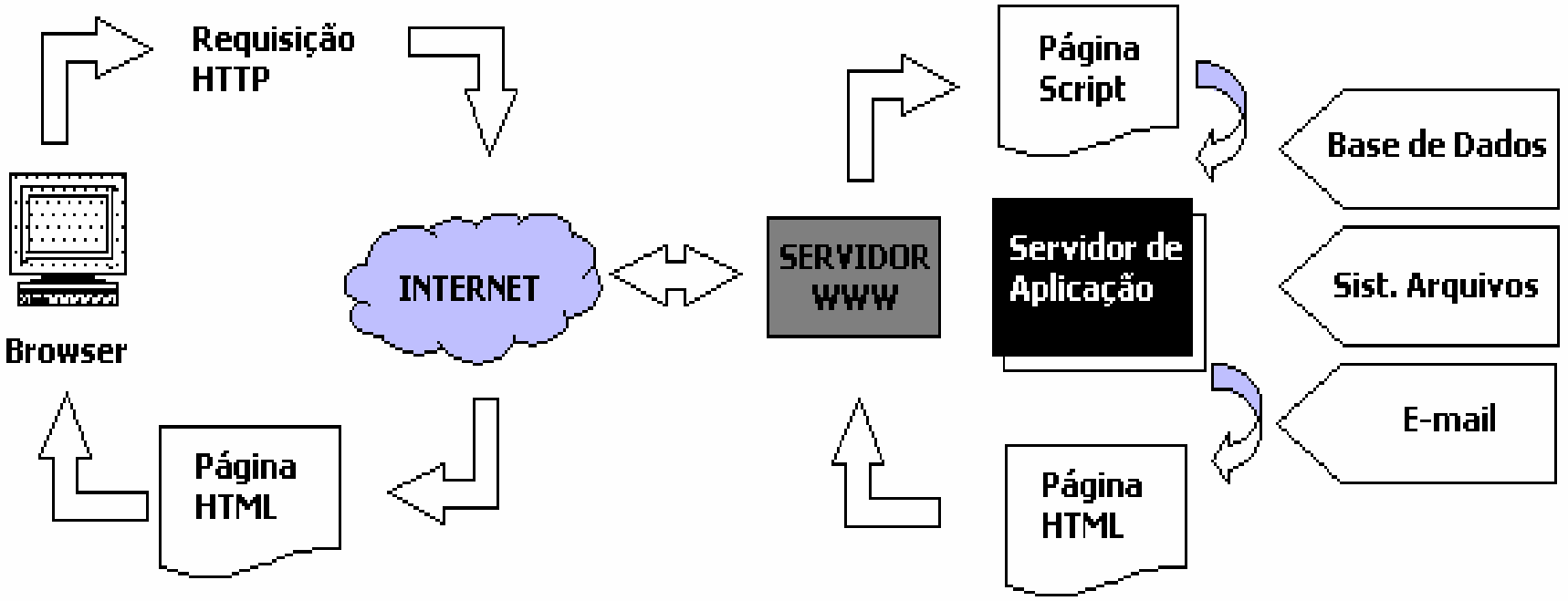
Introdução

- Documentos na WWW
- Compiladores
- Linguagem de Programação de Script incorporada em documentos WWW

Objetivos

- Criar uma linguagem de script com comandos em português
- Desenvolver um software interpretador para a linguagem

Funcionamento de um Script



Especificações HTTP

- CGI
- ISA DLL

Linguagens Script na WWW

- Active Server Pages (ASP)
- PHP
- Meta-HTML
- Cold Fusion

Gramática Livre de Contexto

- Terminais — T= {1, 2, 3}
- Não Terminais — N = {A, B}
- Produções — P = { A -> AB | 1
  B -> 2 | 3 }
- Elemento Inicial — A

Estrutura do Interpretador

- Análise Léxica
- Análise Sintática
- Análise Semântica
- Geração de Código Intermediário
- Otimizações

Análise Léxica

- Extração e classificação dos tokens
- Eliminação Comentários
- Tabela de Símbolos
- Função Hash

Análise Sintática

- Análise Top-Down
- Análise Bottom-Up

Análise Semântica

- Verificação Tipos em Expressões
- Implementação
  - Atributos
  - Regras e Ações Semânticas

Requisitos

- Linguagem com comandos em português
- Comandos Condicionais (SE..ENTÃO)
- Comandos Repetição (ENQUANTO)
- Operações Aritméticas
- Tipos Inteiro e String
- Banco de Dados

Caso de Uso

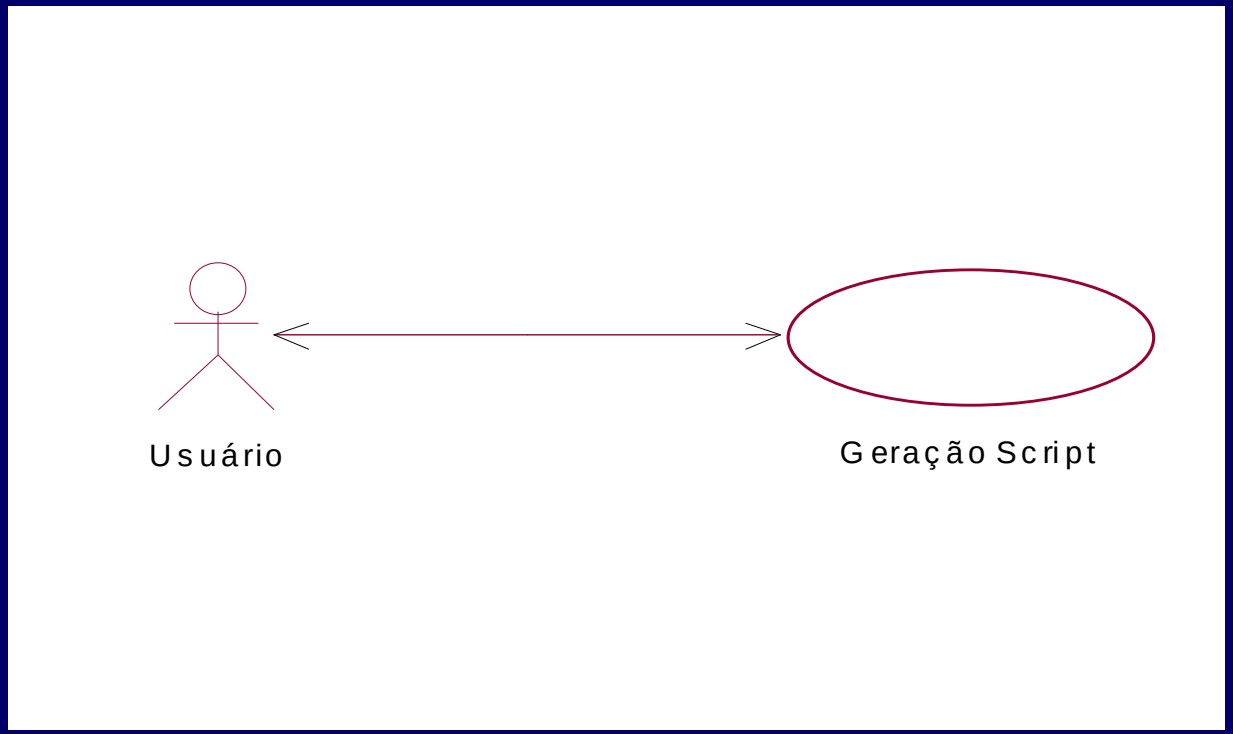
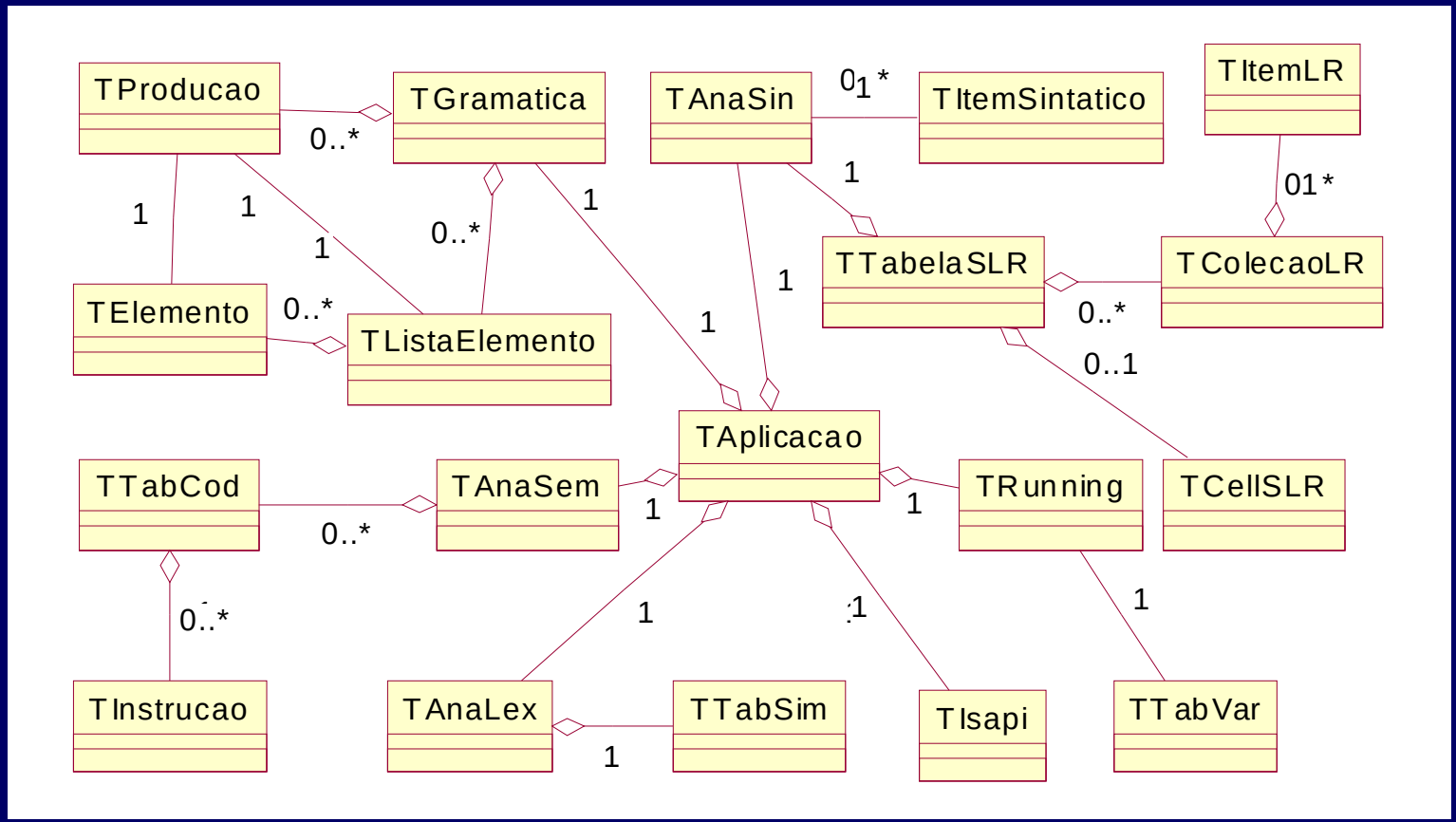


Diagrama de Classe



Expressões Regulares

```
ttIniScript    = <#
ttFimScript    = #>
ttVirgula      = ,
ttPontoVirgula = ;
ttAtribuicao   = :=
ttSomSub       = + | -
ttMulDiv       = * | /
ttID           = ($ letra | letra) (letra | digito | _)*
ttNum          = digito (digito)*
ttString       = “ (digito | letra ) “
ttAbrePar      = (
ttFechaPar     = )
ttOperadorE    = E
ttOperadorOU   = OU
ttOpRelacional = > | <  | =
ttComentario   = % ~(%) %
ttHtml         = ~(<#)
digito         = 0|1|2|3|4|5|6|7|8|9
letra          = A..Z | a..z
```

BNF

1. <S> ::= <HTML>
2. <HTML> ::= <SCRIPT>
3. <HTML> ::= <HTML> <SCRIPT>
4. <HTML> ::= ttHTML
5. <SCRIPT> ::= ttIniScript <LISTACMD> ttFimScript
6. <LISTACMD> ::= <LISTACMD> <COMANDO> ttPontoVirgula
7. <LISTACMD> ::= <COMANDO> ttPontoVirgula
8. <COMANDO> ::= ttID ttAtribuicao <EXPRE>
9. <COMANDO> ::= **”imprime”** ttAbrePar <EXPRE> ttFechaPar
10. <COMANDO> ::= **”se”** <CONDICAO> **“então”** <M> <LISTACMD> **“fimse”**
11. <COMANDO> ::= **”se”** <CONDICAO> **“então”** <M> <LISTACMD> **“senão”** <M> <LISTACMD> **“fimse”**
12. <COMANDO> ::= **“enquanto”** <M> <CONDICAO> **“faca”** <M> <LISTACMD> **“fime”**
13. <COMANDO> ::= ∈
14. <EXPRE> ::= <EXPRE> ttSomSub <TERMO>

BNF

15. <EXPRE> ::= <TERMO>
16. <TERMO> ::= <TERMO> ttMulDiv <FATOR>
17. <TERMO> ::= <FATOR>
18. <FATOR> ::= ttAbrePar <EXPRE> ttAFechaPar
19. <FATOR> ::= ttID
20. <FATOR> ::= ttNum
21. <FATOR> ::= ttString
22. <CONDICAO> ::= <CONDICAO> ttOperadorE <M> <COND_OU>
23. <CONDICAO> ::= <COND_OU>
24. <COND_OU> ::= <COND_OU> ttOperadorOU <M> <COND_LOG>
25. <COND_OU> ::= <COND_LOG>
26. <COND_LOG> ::= <EXPRE> ttOpRelacional <EXPRE>
27. <M> ::= ∈
28. <COMANDO> ::= **“db”** ttAbrePar ttNum <EXPRE> ttAFechaPar
29. <FATOR> ::= **“dbcampo”** ttAbrePar <EXPRE> ttAFechaPar

Analisador Sintático LR

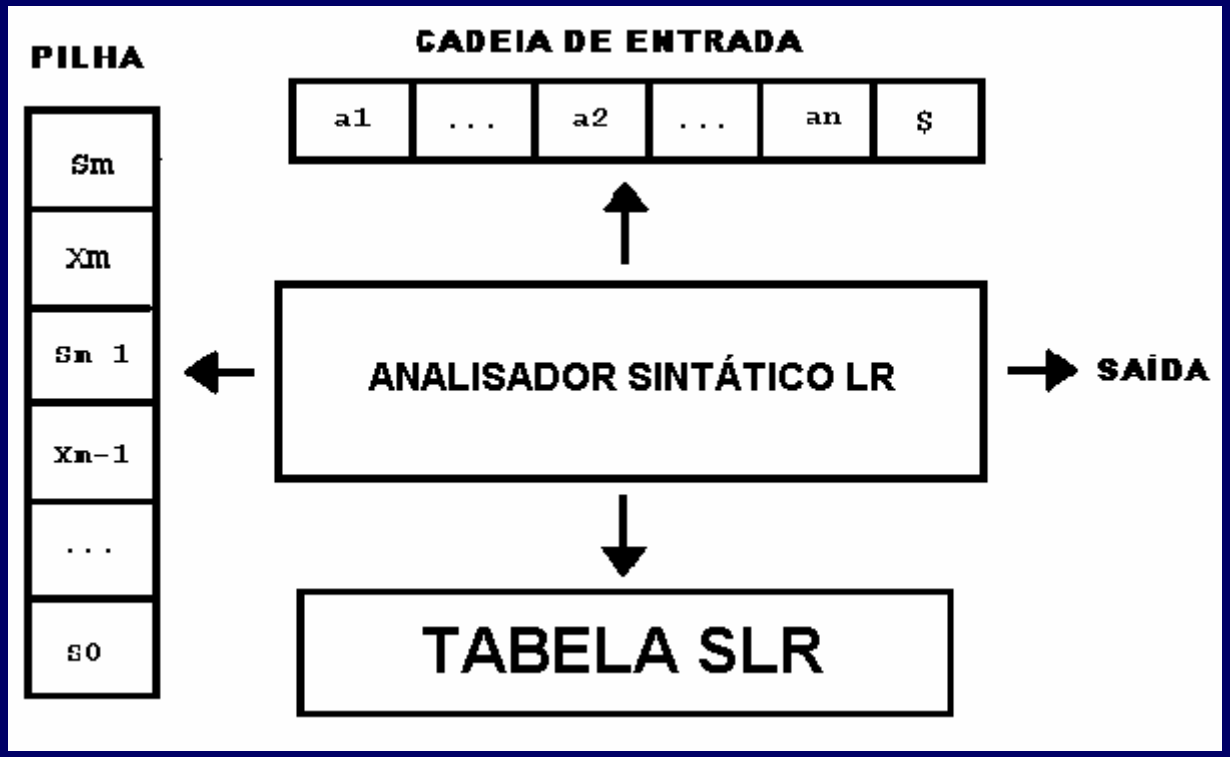


Tabela Sintática SLR

	E	**T**	**F**	**id**	**+**	*****	**(**	**)**	**#**
0	1	2	3	s5			s4		
1					s6				acc
2					r2	s7		r2	r2
3					r4	r4		r4	r4
4	8	2	3	s5			s4		
5					r6	r6		r6	r6
6		9	3	s5			s4		
7			10	s5			s4		
8					s6			s11	
9					r1	s7		r1	r1
10					r3	r3		r3	r3
11					r5	r5		r5	r5

Regras Semânticas

```
<EXPRE>    ::= <EXPRE> ttSomSub <TERMO>


{ EndVar := Pilha[2].EndVar;
  TipVar := Pilha[2].TipVar;
  Se  Pilha[1].NomLex   =  +
    AddInstrucao(tiSOMA, 0, 0, 0);
  Senao
    AddInstrucao(tiSUBT, 0, 0, 0);
}
```

Geração Código

- Quadruplas
- Código Curto Circuito

Tradução da Expressão Lógica "a > b OU c = d E a > d"

```
020  => DSMA 000 000 024 // desvia se a > b para linha 24
021  => DSSE 000 000 022 // desvia sempre para linha 22
022  => DSIG 000 000 024 // desvia se c = d para linha 24
023  => DSSE 000 000 027 // desvia sempre para linha 27
024  => DSMA 000 000 026 // desvia se a > d para linha 26
025  => DSSE 000 000 027 // desvia sempre para linha 27
```

Geração Código

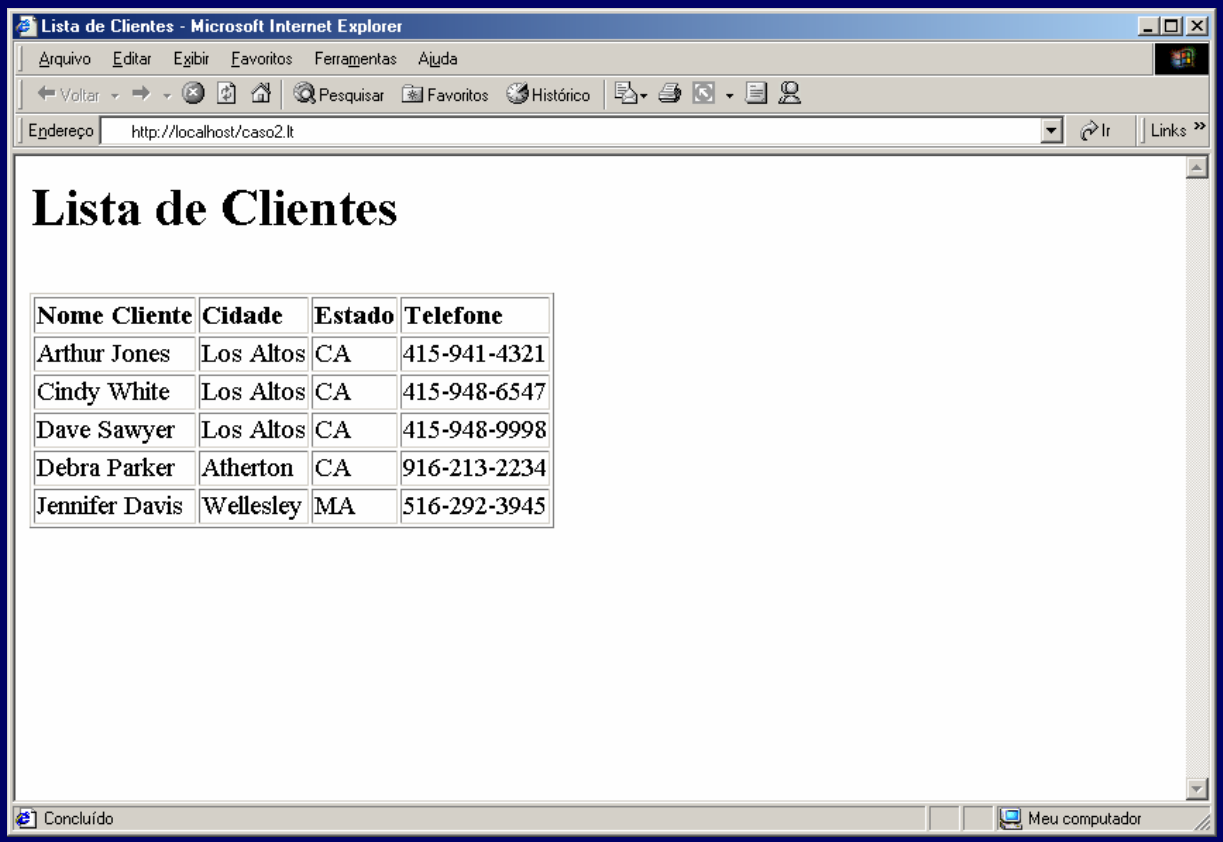
(Técnica Backpatching)
Tradução Comando "SE A > 3 ENTÃO.. SENAO .. FIMSE"

```
008  => DSMA 000 000 010 // se a > 3 desvia (linha 10)
009  => DSSE 000 000 013 // desvia (linha 13)
010  => CRVL 004 000 000 // verdadeiro
011  => IMPR 004 020 000
012  => DSSE 000 000 016
013  => CRVL 005 000 000 // falso
014  => IMPR 005 020 000
015  => DSSE 000 000 016 // próximo comando
```

Estudo Caso

```
DB(10, "DBDEMOS");
DB(30, "SELECT * FROM CLIENTS ORDER BY FIRST_NAME");
imprime("<table border>");
imprime("<tr>");
imprime("  <td><b>Nome Cliente</b></td>");
imprime("  <td><b>Cidade</b></td>");
imprime("  <td><b>Estado</b></td>");
imprime("  <td><b>Telefone</b></td>");
imprime("</tr>");
ENQUANTO DBSTATUS > 0 FACA
  imprime("<tr>");
  imprime("  <td>" + DBCAMPO("FIRST_NAME") + "</td>");
  imprime("  <td>" + DBCAMPO("CITY")  + "</td>");
  imprime("  <td>" + DBCAMPO("STATE")  + "</td>");
  imprime("  <td>" + DBCAMPO("TELEPHONE")  + "</td>");
  imprime("</tr>");
  DB(40, 0);   % Proximo %
FIME;
imprime("</table>");
```

Estudo Caso



Lista de Clientes

Nome Cliente	Cidade	Estado	Telefone
Arthur Jones	Los Altos	CA	415-941-4321
Cindy White	Los Altos	CA	415-948-6547
Dave Sawyer	Los Altos	CA	415-948-9998
Debra Parker	Atherton	CA	916-213-2234
Jennifer Davis	Wellesley	MA	516-292-3945

Conclusões

- A linguagem implementada atingiu os objetivos de criação da linguagem e do interpretador
- A implementação do algoritmo de construção da tabela SLR trouxe flexibilidade para modificação e criação de extensões na gramática
- Comparada com as linguagens de script estudas a linguagem desenvolvida apresenta limitações que podem ser superadas futuras extensões

Extensões

- Funções e procedimentos
- Novos tipos de dados
- Ambiente desenvolvimento integrado
- Banco de dados
- Gerador sintático automático