SAL:
A Service Level Agreement Language

Donna Eng Dillenberger

December 14, 2003

Document Changes

Date	Changes
9/22/03	Document Created
10/08/03	Included variables, control-flow, and functions.
10/22/03	Included Language Reference Manual
11/01/03	Included Built In identifiers, specified that tag values may appear in any order
12/14/03	Added Architecture, final sections

Contents

An Introduction to SAL

The SAL language is designed to help Information Technology (IT) professionals and Business Process Modelers quickly build metrics, prototype, evaluate and optimize the configuration of their systems using Service Level Agreements (SLAs). Service Level Agreements are legal contracts used by business organizations to specify levels of performance, availability, storage, server and network capacity that an IT organization will provide to support the business processes. When these levels are not met, IT organizations are penalized by compensating the business in dollar amounts, for the time that lack of access to the computer infrastructure causes business transactions to be missed.

Examples of Service Level Agreements rules are:

This IT organization will support the following 4 Business Processes:

1. Trading Application
2. Prospectus Browsing Application
3. Risk Portfolio Analysis
4. Maintenance, Background jobs

- The Trading Application:
  1. Will have a Mean Time to Recovery [1] of 2 seconds.
  2. All requests to the Trading Application will receive a response within 3 seconds.
  3. The profit from each Trading Application request is ten dollars for trades.
  4. The IT cost of each Trading Application request should not exceed one dollar.
  5. The response time should be less than 3 seconds
  6. If it's after 9 am and the above Mean time to Recovery and Response Time goals are not met then a penalty of 100 dollars shall be incurred for each transaction.

- The Prospectus Browsing Application:
  1. Between 9am and 10 pm will have a Mean Time to Recovery of 5 minutes
  2. All requests to this Application will receive a response within 1 minute
  3. The profit from each request for this application is two dollars
  4. The IT cost of each request to this application should not one dollar
  5. No penalty to IT if the above rules are not met for Prospectus Browsers. The above rules are best effort.

[1] Mean time to recovery entails that this application must not be inaccessible for longer than 2 seconds)

- The Risk Portfolio Analysis Application:
  1. Between 6pm and 12 pm, will have a Mean Time to Recovery of 2 hours
  2. All requests to this Application will receive a response within 8 hours.
  3. The profit from each request for this application is two hundred dollars.
  4. The cost of each request to this application should not exceed 10 dollars.
  5. If any of the above rules are missed, IT's payment to the business will be 100 dollars for:
     - Each transaction missed (because Mean Time to Recovery was not met) between 6 pm and 12 pm.
     - Each transaction that failed to get a response within 8 hours

Maintenance and Background work:

1. Will have a Mean Time to Recovery of 4 hours between 1 am and 7 am
2. All requests to this Application will receive a response within 24 hours.
3. The profit from each request for this application is 3 dollars.
4. The cost of each request to this application should not exceed 2 dollars.
5. No penalty to IT if the above rules are not met for Prospectus Browsers. The above rules are best effort.

To specify the above using SAL, one would write:

```
:LineOfBusiness Trading;
{       For Scope = 900 to 1700;
        (
                MeanTimeToRecovery = 2 seconds;
         )
        :ResponseTime = 3 seconds;
        :Profit = 10 dollars;
        TxCost = 1 dollar;
        For Scope >= 900
        (
                :Penalty = 100 dollars;
        )
}

:LineOfBusiness ProspectusBrowsing;
{
        For Scope = 900 to 2200;
        (
                :MeanTimeToRecovery = 5 minutes;
        )
        :ResponseTime = 1 minute;
        :Profit = 2 dollars;
        :TxCost = 1 dollar;
```

```
        :Penalty =  0 dollars;
}

:LineOfBusiness RiskPortfolioAnalysis;
{       For Scope = 600 to 1200;
        (
                :MeanTimeToRecovery = 2 hours;
        )
        :ResponseTime = 8 hours;
        :Profit = 200 dollars;
        :TxCost = 10 dollars;
        For Scope = 600 1200
        (
                :Penalty = 100 dollars;
        )
}

:LineOfBusiness MaintenanceBackground;
{       For Scope = 100 to 700;
        (
        :MeanTimeToRecovery = 4 hours;
        )
        :ResponseTime = 5 hours;
        :Profit = 3 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
}
```

The Interpreter for SAL will:

- Check that for each defined Application, there are MeanTimetoRecovery, ResponseTime, Profit, TxCost and Penalty values defined and specified in the right units.
- Transform the Application definitions above to java classes

The java code could then be fed into SLA deployment machines (these don't currently exist yet) that take the above definitions and:

1. Creates Monitoring Agents for each Application and measures whether the above thresholds are violated
2. In more sophisticated SLA deployment machines, forecast when the defined thresholds will be violated. Parameter passed into Forecast would request how far into the future a forecast is desired.
3. Provisions/deprovisions/reconfigures IT routers, servers and storage to prevent Threshold violation from occurring.
4. In more sophisticated SLA deployment machines, the IT adaptation would maximize profit, minimize cost, within the boundaries of not incurring IT penalties.

The programmer would invoke the above actions (2)-(4) by coding:

```
:Actions;
{
        :Forecast = yes;
        :Optimize = yes;
        :Provsion (CurrentRevenue);
}
```

For the above, the SAL Interpreter would derive/calculate the right parameters to invoke the above functions:

- Forecast
- Optimize
- Provision

The implementations for 1-4 above are meant to be pluggable Different vendors can supply different implementations to differentiate themselves. The interfaces will be generated by the compiler but the package implementations will not be provided (these would be vendor specific).

This would print the results of Creating Agents, Forecast warnings, what provisioning, configuration actions were taken, and what optimization tradeoffs were invoked.

A key initial step in writing a SLA is to identify the Business Processes, the IT infrastructure will support. IT implementations should be driven by the business needs for the measurement of progress and fulfillment of business goals (e.g. ROI, EBIT). At a fundamental level, each organization has a general goal, e.g. maximize Return On Investment (ROI) or Earnings Before Interest and Taxes (EBIT). An organization's next step is to translate those goals into Lines of Businesses that can support maximizing these business goals. For example a Financial Institution may choose to offer three Lines of Businesses to maximize ROI:

- A Trading Service
- A Risk Portfolio Analysis Service
- A Prospectus Downloading Service

These three types of services all would have quarter profit and cost targets. The IT implementations built to support these services should help a Business measure how each service is progressing (number of transactions/day, IT cost of each transaction, profits of each service per day) to help the Business make decisions on:

- Whether to increase capacity for that service
- Whether to offer some services more frequently than other dependent on time of day, day of week or season of year
- Shut down that service (not profitable)
- Offer a new service.

The code for the above would look like:

```
:ReturnOnInvestment;
{
        :LineOfBusiness Trading;
        :LineOfBusiness RiskPortfolioAnalysis;
        :LineOfBusiness  ProspectusBrowsing;
}

:ITCosts:
{
        :Routers = yes;
        :Storage = no;
        :Servers = yes;
        :Bandwidth = no;
        :Power = yes;
        :Personnel = 20;
}
```

The Interpreter would take the above code and perform the following actions:

- Check that ReturnOnInvestment had associated Lines Of Business types
- Check that each LineOfBusiness type had associated Application structures defined (previous code sample that gives us IT metrics)
- Check that ITCosts had a structure delineating which physical resources were to be monitored, and that Personnel was assigned a value.
- Deploy monitoring agents (just calls to interfaces) to get NumberOfTransactions/day for each LineOfBusiness.
- Calculate ReturnOnInvestments = Sum of Profits from each LineOfBusiness transaction.
- Obtain ITCosts for each Line of Business transaction by deploying Monitoring agents on each ITCosts member (just a call to an interface)
- Use the above data from the Monitoring agents to calculate IT costs per Application
- Support the verbs FORECAST, PROVISION, OPTIMIZE by calling the correct methods and generating the data needed by each method (from the bullets above).

In the Service Level Agreement Language for this project, I propose to offer language types and operations that help Businesses:

- Translate Business Process Goals to Business Process Metrics
- Indicate to the IT infrastructure what IT measurements should be used to report on Business Process Metrics
- Indicate to the IT infrastructure what trade offs can be made when adding/deleting storage, network and server capacity to a SLA application based on cost and profit.

SAL is an intuitive, object-oriented, portable, powerful way to specify Business Goals. These Business Goals dynamically cause agents to be created to monitor the progress

towards Business Goals and can call the right packages to optimize the IT infrastructure to support these goals.

The foremost goal in the creation of this language was to make it easy to learn and straightforward to program. The user should be left to concentrate on the structure and design of their Business Processes, not the syntax of the modeling language. SAL was created to be consistent (strong type checking) and intuitive (type definitions are in the language of Business Architects), even to users who have limited programming experience.

By supporting the concept of objects, each Application of the Business has its own attributes. By breaking down the Line Of Business into its Application components, we believe that most users will find SAL to be conceptually intuitive to use.

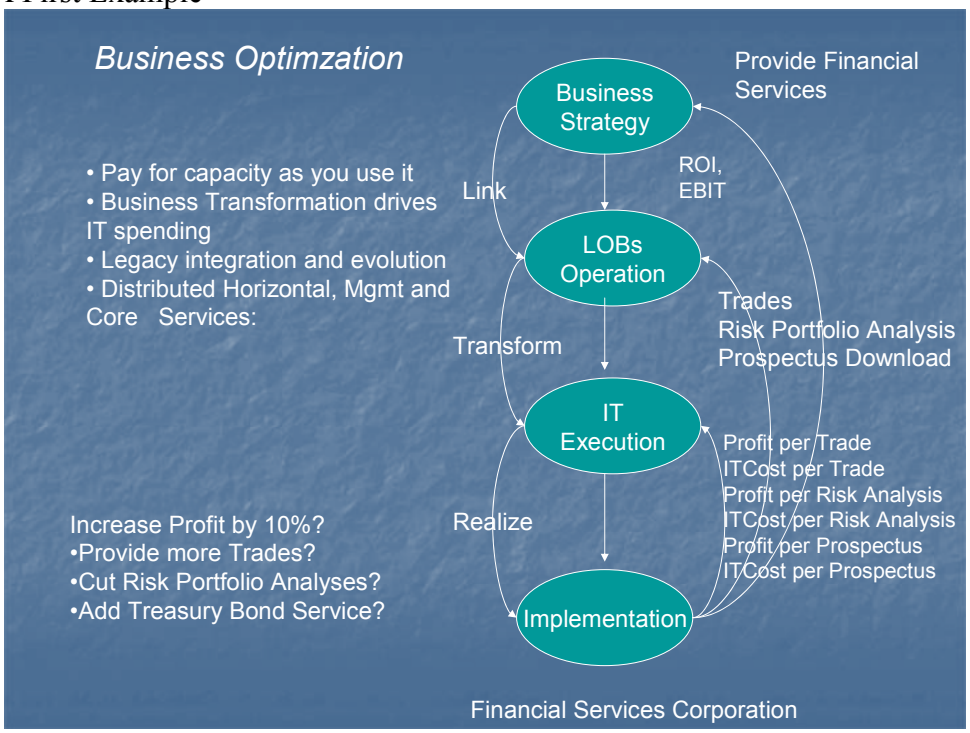
The compiler will accept a SAL specification as an input and generate Java source code. This source code can be integrated with a larger Java project and compiled with any Java compiler. Because Java is SAL's target platform, portability is limited only by the availability of a Java Virtual Machine.

With the power and simplicity of a robust SAL specification engine, the time to design, implement and test a Business Process' IT infrastructure requirements can be reduced by an order of magnitude. SAL's simple and intuitive language syntax ensures that most errors are detected at compile time and that compiled SAL code is as accurate as it can be. Monitoring Agents are automatically deployed. Profits are automatically calculated. IT Costs are automatically deduced. The deployment of SAL will focus its users on which Business Processes are the most profitable, at what time of the day, week, year, and which Business Processes are the most costly in terms of IT support to provide.

Tutorial

A Service Level Agreement specification is a sequence of definitions of Business Applications a corporation will purchase IT infrastructure to support. The IT infrastructure should measure the cost of Business Transactions, the profit generated from each Business Application and is able to signal a reconfiguration or automatically call reconfiguration packages to optimize profit and minimize cost.

A First Example



In the figure above, a Financial Institution decides that its business strategy is to provide premier Financial Services. The corporation measures how well it's providing these services based on Return On Investment and Earnings Before Interest and Taxes. The corporation decides it will create three Lines of Businesses to maximize ROI and EBIT. The Lines Of Businesses are: Trades, Portfolio Analyses and Prospectus Browsers.

To measure how each Line of Business is doing, IT deploys these applications and measures (1) Number of Transactions (2) Profit for each transaction and (3) IT Cost of each transaction for each Line of Business Application: Trades, Portfolio Analysis and Prospectus Browsers.

The code that the Business Modeler would write was documented as examples in the above section. The code to deploy Agents to measure the IT execution metrics were automatically generated by the Interpreter and would be deployed by the execution of this code. SAL's Interpreter generates syntax errors if a Business Strategy does not provide Business Metrics to measure (e.g. ROI, EBIT), Lines of Businesses to calculate the Business Metrics and Members of an ITCost structure to automatically generate the code to calculate cost and profit for the Business.

Interpreting and Running SAL Spec Files
Once you have a simple example of a Business Process that you want to run through SAL, create a .sal file with your Application, Lines of Business and ITCost structures, compile it using SAL. If you have created a Business Process named ProfitableFinancialServices in a file called ProfitableFinancialServices.sal, you would run it by specifying:
$ java SALMain ProfitableFinancialServices.sal

Results from Running SAL Spec Files
After the above command is issued for the second example in the next section (Profitable News Services), the results that will be shown on your screen are:

InternationalNews
Current Arrival Rate: 611
Total Day Profit: 611000
Total Day Cost: 183732
Current Revenue 427268
Forecasted Arrival Rate: 49491
Projected Arrival Rate change 8000%
Forecasted Profit 494910
Projected Profit change -19%
Forecasted Cost 34643
Projected Cost change -81%
Forecasted Revenue 460267
Projected Revenue change 7%

NationalNews
Current Arrival Rate: 365
Total Day Profit: 219000
Total Day Cost: 109500
Current Revenue 109500
Forecasted Arrival Rate: 14600
Projected Arrival Rate change 3900%
Forecasted Profit 87600
Projected Profit change -60%
Forecasted Cost 2920
Projected Cost change -97%
Forecasted Revenue 84680
Projected Revenue change -22%

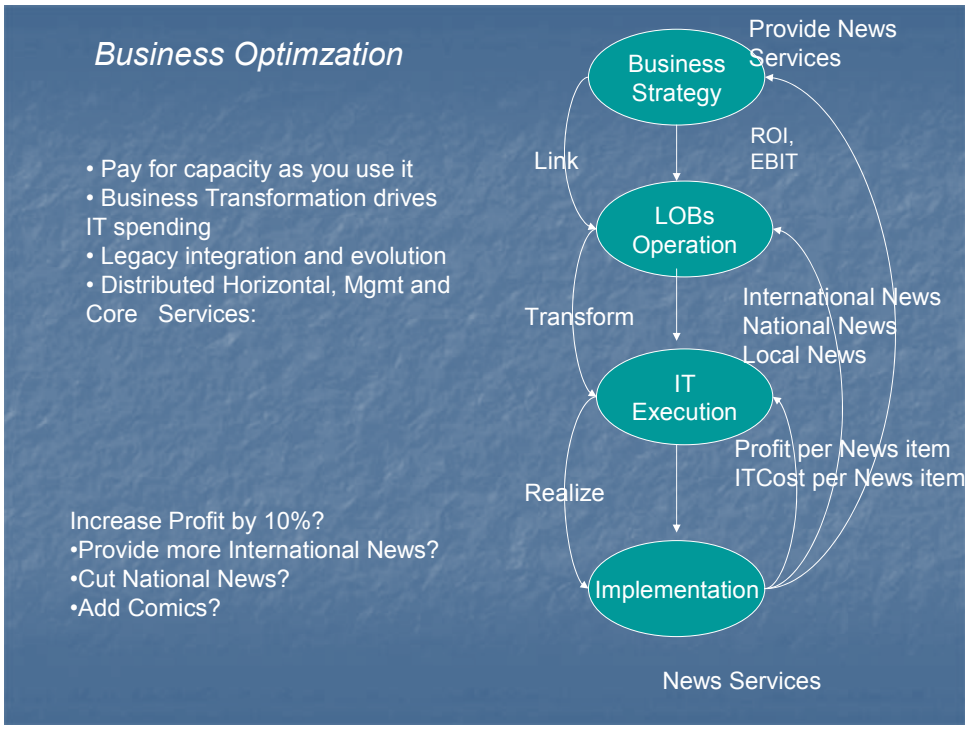
LocalNews
Current Arrival Rate: 216
Total Day Profit: 432000
Total Day Cost: 72216

Current Revenue 359784
Forecasted Arrival Rate: 6264
Projected Arrival Rate change 2800%
Forecasted Profit 125280
Projected Profit change -71%
Forecasted Cost 68904
Projected Cost change -4%
Forecasted Revenue 56376
Projected Revenue change -84%

Total Company Profit 1262000
Total Company Cost 365448
Current Company ROI: 3.4532956809176683
Forecasted Company Profit 707790
Forecasted Cost 106467
Forecasted Company ROI: 6.647975429006171
Line of Business with Forecasted Largest Revenue: InternationalNews 460267
Line of Business with Forecasted Largest Growth:InternationalNews 7%
Line of Business with Current Largest Revenue: InternationalNews 427268

Provisioning Storage, Network and Server resources for Line of Business with Largest Growth: InternationalNews

More Examples



In the figure above, a News Services Institution decides that its business strategy is to provide premier News Services. The corporation measures how well it's providing these services based on Return On Investment and Earnings Before Interest and Taxes. The corporation decides it will create three Lines of Businesses to maximize ROI and EBIT. The Lines Of Businesses are: International News, National News and Local News.

To measure how each Line of Business is doing, IT deploys these applications and measures (1) Number of News items (2) Profit for each news item (3) IT Cost of each news item for each Line of Business Application: International News, National News and Local News.

The code that the Business Modeler would write is:

```
:ReturnOnInvestment;
{
        :LineOfBusiness InternationalNews;
        :LineOfBusiness NationalNews;
        :LineOfBusiness  LocalNews;
}
```

```
:LineOfBusiness InternationalNews;
{
        :MeanTimeToRecovery = 2 minutes;
        :ResponseTime = 5 seconds;
        :Profit = 10 dollars;
        :TxCost = 5 dollars;
        :Penalty = 2 dollars;
}

:LineOfBusiness NationalNews;
{
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
}
:LineOfBusiness LocalNews;
{       :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 200 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700
        {
                :Penalty = 100 dollars;
        }
}
:ITCosts:
{
        :Routers = yes;
        :Storage = no;
        :Servers = yes;
        :Bandwidth = no;
        :Power = yes;
        :Personnel = 200;
}
:Actions;
{
        :Forecast = yes;
        :Optimize = yes;
        :Provision(Growth);
}
```

The code to deploy Agents to measure the IT execution metrics would be automatically generated by the Interpreter and would be deployed by the execution of this code. SAL's Interpreter generates syntax errors if a Business Strategy does not provide Business Metrics to measure (e.g. ROI, EBIT), Lines of Businesses to calculate the Business

Metrics and Members of a TxCost structure to automatically generate the code to calculate cost and profit for the Business and pass to forecasting, and optimization packages. If the verb PROVISION is requested, the compiler would also invoke libraries that would configure servers, storage and network to meet the Line Of Business requirements.

Reference Manual

Grammar Notation

Grammar symbols are defined as they are introduced in this document. Regular expression notation has been used to make the productions more perspicuous. Symbols in italics are nonterminals. Quoted symbols are terminals. `s?' denotes the symbol s is optional. `s*' denotes the symbol s may be occur zero or more times. `s+' denotes the symbol s may occur one or more times. `(s|t)' denotes a choice between the symbol sequences s and t. Parentheses are also used to group symbols with respect to `?', `*' and `+'.

Lexical Conventions

A program consists of one Return On Investment declaration, an ITCost declaration, an Actions declaration and one or more Line of Business declarations defined in a *.SAL file. Programs are written using the Unicode character set. The precise version of Unicode used will be determined by the Java Virtual Machine used to run the SAL compiler. For more information, refer to your Java Developer's Kit documentation.

Line Terminators

The sequence of input characters is divided into lines by line terminators. Lines are terminated by the ASCII characters CR (\carriage return"), LF (\linefeed"), or CR LF. The CR LF combination is counted as one line terminator, not two.

Comments

Only C++-style comments are supported. A C++-style comment begins with the characters // and ends with a line terminator. /* and */ have no special meaning inside comments beginning with //

Whitespace

Whitespace is defined as the ASCII space, horizontal tab and form feed characters, as well as line terminators and comments.

Tokens

There are six classes of tokens: identifiers, keywords, built in identifiers, operators, integers and separators. Whitespace is ignored except as a token separator. Whitespace is sometimes required to separate adjacent tokens that might otherwise be combined into one token (i.e., identifiers, keywords and constants). Token formation is greedy: the input is searched for the longest string of characters that could constitute a token.

Identifiers
An identifier is a sequence of letters or digits, the first of which must be a letter. There is no limit on the length of an identifier. Two identifiers are the same if they have the same Unicode character for every letter and digit. Identifiers that have the same external appearance may not by identical. For example, the Latin capital letter 'A' and the Greek capital letter `A' (\Alpha") are different Unicode characters that have the same appearance when displayed.

*Identifier -> letter (letter | digit)**

Keywords
The following identifiers are reserved for use as keywords, and may not be used otherwise:

minutes	second	For
seconds	hour	to
hours	dollar	
minute	dollars	

Built in Identifiers
The following identifiers are reserved as built in identifiers, and may not be used otherwise. They are all prefaced with a colon.

ReturnOnInvestment	Penalty	Bandwidth
LineOfBusiness	Actions	Power
ITCost	Forecast	Personnel
Scope	Provision	Growth
ResponseTime	Optimize	CurrentRevenue
MeanTimeToRecovery	Routers	ProjectedRevenue
Profit	Storage	none
TxCost	Servers	

Integers
An integer is a sequence of digits. The limit on the length of an integer is 12 digits. Two integers are the same if they have the same Unicode character for every digit.

Integer -> (digit)+

Operators
The following symbol is an operator that is used to assign an integer to the following built in identifiers: Scope, |ResponseTime, MeanTimeToRecovery , Profit , TxCost , Penalty , Personnel:

=

Separators
The following ASCII characters are separators:

{ } ;

Return On Investment declaration
A "sal" file always starts with a Return On Investment declaration. This begins with the key word ReturnOnInvesment followed by a ";". The next token is a " {" . The declaration ends with a "}". Within the braces are one or more lines that begin with the key word "LineOfBusiness" followed by an Identifier for the Line of Businesses, followed by a ";". For every line of business declared in the Return on Investment definition, a Line of Business Specification must follow.

Line Of Business declaration
A Line Of Business declaration begins with an Identifier of the Line of Business followed by a ";". The next token is a " {" . The last token for a Line of Business Specification is a "}". Within the braces are all of the following built in identifiers followed by the operator "=" followed by an integer value for the key word, followed by a ";". No built in identifier below can be missing. All of the built in identifiers must have an integer assigned to it.

MeanTimeToRecovery – The duration of time the application must be available following an outage of the Line of Business Application. This is an integer followed by the keywords minute(s)|second(s)|hour(s).

ResponseTime – The time an end user must receive a response for a transaction initiated against the line of business. This is an integer followed by the keywords minute(s)|second(s)|hour(s).

Profit – The amount of US dollars one transaction for this Line of Business will bring to the company. This is an integer followed by the keyword dollar(s).

TxCost – The amount of US dollars one transaction cost in Information Technology (IT) services. This is an integer followed by the keyword dollar(s).
Penalty – The amount of US dollars missing the Mean Time To Recovery goal or the Response Time goal would incur on the IT organization. This is an integer followed by the keyword dollar(s).

Values for Mean Time To Recovery, Response Time and Penalty can be optionally specified to apply to only certain times of the day. This is done by using the "For" keyword followed by the "Scope" keyword, followed the operator "=", followed by an Integer followed by the keyword "to" followed by an Integer, followed by a ";". Valid values for Scope are 0 to 2400. The time format is hhmm, where hh = The hour time in US Military format, and mm = minutes. The beginning time must be less than the ending time. Scopes cannot span more than 24 hours. If no time of day scope is declared for Mean Time to Recovery, Response time or Penalty, the default scope is set to the full 24 hours in a day.

ITCosts Declaration
An ITCost declaration begins with the built in identifier "ITCosts" followed by a ";". The next token is a " {" . The last token for an ITCost specification is a "}". Within the braces are at least one of the following built in identifiers followed by a ';"

Routers – Include costs of routers for the IT infrastructure supporting these Lines of Businesses.

Storage - Include costs of the storage (storage volumes, storage fabric, storage switches, storage appliances) for the IT infrastructure supporting these Lines of Businesses.

Servers - Include costs of the servers (Racks, Hardware, Operating Systems, Memory, Software) for the IT infrastructure supporting these Lines of Businesses.

Bandwidth - Include costs of the internet and intranet network bandwidth for the IT infrastructure supporting these Lines of Businesses.

Power - Include costs of the electrical power consumption required for the IT infrastructure supporting these Lines of Businesses.

Personnel – Include costs of the people required to run the IT infrastructure. This built in identifier is followed by the operator "=", followed by an integer, followed by a ";". The integer denotes how many people are required to run the IT infrastructure.

All of the costs above will be monitored and aggregated to a day's cost (24 hours).

Actions Declaration
An Action declaration begins with the built in identifier "Actions" followed by a ";". The next token is a " {". The last token of the ITCosts declaration is a "}". Within the braces are at least one of the following built in identifiers followed by a "yes|no" and then a semicolon ";"

Forecast = yes
This action will forecast the next day's arrival rate, revenue, profit, cost and penalty.

Optimize = yes
This action will calculate which Line Of Business would bring the most profit if given additional IT resources. This action cannot be issued without first issuing "Forecast".

Provision (Growth|CurrentRevenue|ProjectedRevenue|none)
This action will deploy new IT resources (servers, storage, network) needed to instantiate a new instance of the Optimal Line of Business supporting applications. If the options: ":Growth" is specified, IT resources with be deployed to the Line Of Business with the largest growth.

:CurrentRevenue is specified, IT resources will be deployed to the Line Of Business with the largest current revenue.

:ProjectedRevenue is specified, IT resources will be deployed to the Line Of Business with the largest forecasted revenue.

These actions cannot be issued without first issuing "Optimize" and "Forecast".

:none is specified, no IT resources will be provisioned.

Service Level Agreement Specifications

A Service level Agreement specification is a fille that contains a Return On Investment declaration followed by any number of Line of Business declarations, followed by an ITCost and Action declaration. A specification is compiled into a Java class. Executing the generated class will begin execution of the Service Level Agreement.

SLASpecification -> ReturnOnInvestment declaration
(LineOfBusiness declaration)+
ITCost declaration
Actions declaration

Names

A name is bound by a declaration and is available at any point in the specification that follows the declaration. A name must be unique within the specification.

Return On Investment Declaration

A specification must begin with a Return On Investment declaration. There may be only one Return On Investment declaration in a specification. A Return On Investment declaration takes the form:

ReturnOnInvestment Declaration -> ':ReturnOnInvestment' ';'
'{'
(':LineOfBusiness' Identifier)+
'}'

Line of Business Declaration

One or more Line of Business declarations must follow a Return On Investment declaration. A Line of Business declaration takes the form:

Line of Business declaration -> Identifier ';'
':MeanTimeToRecovery' = Integer ';'
':Responsetime' = Integer 'second' 's?' ';'
';'?
':Profit' = Integer 'dollar' 's?' ';'
':TxCost' = Integer 'dollar' 's?' ';'
('For Scope' = Integer 'to' Integer ';)?
'(' ?
':Penalty' = Integer 'dollar' 's?' ';'
')' ?
'}'

The tags: MeanTimeToRecovery, Responsetime, Profit, TxCost, Penalty and their values can appear in any order.

IT Costs

Following the Line of Business declarations, is the IT Costs declaration. There may be only one IT Costs declaration in a specification. An IT Costs declaration takes the form:

IT Costs Declaration -> *':ITCosts' ';'*
'{'
':Servers' = 'yes|no' ';'
':Routers' = 'yes|no' ';'
':Storage' = 'yes|no' ';'
':Bandwidth' = 'yes|no' ';'
':Power' = 'yes|no' ';'
':Personnel' = Integer ';'
'}'

The above tags: Servers, Routers, Storage, Bandwidth, Personnel can appear in any order.

Action

Following the IT Costs declaration, is the Action declaration. There may be only one IT Costs declaration in a specification. An Action declaration takes the form:

Action Declaration -> *':Action' ';'*
'{'
':Forecast' = 'yes|no' ';'
':Optimize;' = 'yes|no' ';'
':Provision;' = '('
':Growth'
|':ProjectedRevenue'
|':CurrentRevenue'
|':none'
')' ';'
'}'

Project Plan

Antlr coding style
If an Antlr rule is short and contains only one choice without any action, then it will be written in one line. The colon ":" always starts at the ninth column unless the string in front of :" is too long. In the one line mode, ;" is at the end of this line.

If an Antlr rule has multiple choice or actions, the ;" is placed in a separated line at the ninth column. "|" is also at ninth column. Short actions are in the same line of its rule and at the right half of the screen. Long action lines start at the thirteenth column of the next line. Code in actions follows the Java coding style.

Lexem (token) names are in upper cases and syntactic items (non-terminals) are in lower cases, which may contain the underscore "_".

Java coding style
Indentation of each level is 4 spaces. The left brace "{" occupies a full line and is at the same column as the first character of the previous text. The right brace "}" occupies a full line and at the same column of its corresponding "{ ". One space between arguments and their parentheses "(" ")", but no space between function name and "(". Add spaces between operators and operands for outer expressions. Use spaces, do not use tabs. The then-part and else-part of an "if" statement must not be at the same line of "if" or "else". Similar for "for" statement.

Names:
Class names, variables and Method names are English words or phrases with each word starting with a capital letter.

The lowercase letter "i" is used to control "for" loops.

Project Timeline
The following deadlines were set for key project development goals.
10-08-2003 Language whitepaper, core language features defined
10-22-2003 Development environment and code conventions defined
11-07-2003 Language reference manual, grammar complete
11-14-2003 Parser complete
11-21-2003 Code generation complete
11-28-2003 Error recovery complete
12-02-2003 Code freeze, project feature complete

Software Development Environment
This project will be developed on Windows using Java SDK 1.3.1. The parser and scanner will be developed using ANTLR. The project will be tested using a test harness written in Perl.

Software project environment
Most of the programs are written in Java. The lexer, parser and walker are written in Antlr, and will be translated to java code. The test harness was written in Perl. A makefile was created for build.

Operating systems
Since this project is developed in pure Java, it can be used anywhere that a Java VM is running. Testing requires an environment that can support Perl. Build and compilation uses a Makefile. Both Perl and Makefiles are supported in UNIX and Intel Cygwin environments.

Java 1.3
Java is a simple, object-oriented, architecture neutral, portable, multi threaded language.

Antlr
The language lexer, parser and walker are written in Antlr, "a language tool that provides a framework for constructing recognizers, compilers, and translators from grammatical descriptions containing C++ or Java actions."

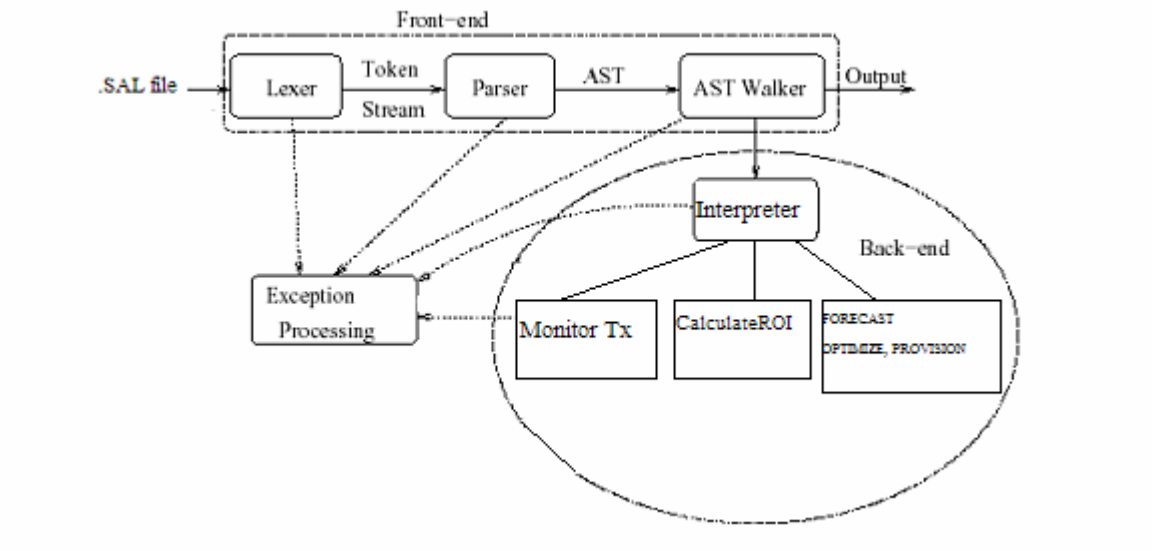
Perl

Perl is a general-purpose programming language originally developed for text manipulation and now used for system administration, web development, network programming, and GUI development. The language is intended to be practical (easy to use, efficient, complete) rather than beautiful (tiny, elegant, minimal). It supports both procedural and object-oriented (OO) programming, and has built-in support for text processing. I chose this language for a test harness due to its text processing capabilities (because I do a lot of pattern matching), file manipulation and its ability to invoke and monitor java processes.

Project log
Here are the dates of significant project milestones. Deviations from the timeline initially proposed are highlighted in italics. They are all the December dates.
10-08-2003 Language whitepaper, core language features defined
10-22-2003 Development environment and code conventions defined
11-07-2003 Language reference manual, grammar complete
11-14-2003 Parser complete
12-01-2003 Interpreter complete
12-13-2003 Walker complete
12-09-2003 Code generation complete
12-10-2003 Error recovery complete
12-15-2003 Test Harness and testing complete
12-15-2003 Code freeze, project feature complete

Architecture Design

The SAL interpreter consists of these components: a lexer that reads the user input or program files to tokens, a parser that analyzes the syntactic structure of the program and converts it to an abstract syntax tree, a tree walker that travels the abstract syntax tree and calls corresponding functions, a executor that looks up symbol tables and (1) deploys agents to monitor transactions for the specified Lines of Businesses (2) Calculates Profits, Costs and Return on Investments based on the measurements received (3) Forecast future transaction growth, profits, costs and Revenue and optimizes the best Line of Business to provision additional IT (storage, server, network, personnel) resources. If there are errors with the initial .sal file, a simple error and exception processing mechanism is used.



The lexer, parser and tree walker are implemented by Antlr. The lexer, parser and tree walker are implemented in Antlr file sal.g,. The tree walker stores the values that are parsed in data structures and calls the Interpreter in class SALInterpreter.

The main() method is in the class SALMain. It reads the filename of the .sal file to be processed, then initializes the lexer, the parser and the tree walker . The class SALMain also handles exceptions and errors occur in other classes, and prints corresponding messages.

The back end (SA Interpreter) is called from the tree walker when the file has been completely parsed. For each Line of Business defined in the input file, the SAL Interpreter:

- Obtains the number of transactions for each Line of Business
- Checks whether a Line Of Business' goals have been breached
- Calculates the cost to run the Information Technology infrastructure required for this line.
- Calculates the penalty incurred for not meeting the LOB's IT goals

- Calculates the profits for the LOB
- If Forecast was requested then the Interpreter calculates the forecasted values for all of the items above.
- If Optimize was requested then the Interpreter calculates which LOB had the largest:
    - Current Revenue
    - Projected Revenue
    - Growth
- If Provision was requested, then the Interpreter will provision more IT resources to the LOB that had the most Current Revenue, Projected Revenue or Growth based on the inputted Provision parameter.
- At the end of these actions a listing with each LOB's profits, costs, projected profits, and costs (if Forecast was requested) is listed.
- Also the total company's profits, costs and forecasted earnings are also printed on the screen.

The Interpreter was written so that the classes to:

- Deploy agents to collect monitoring data
- Forecast future revenue
- Obtain IT costs

would be pluggable so that these modules can readily be replaced and plugged into by advanced systems management systems from vendors (e.g. Tivoli, Computer Associates, BEA, SAP). By using late-binding methods different functions for the above methods can be tried.

Testing Plan

Testing is evolved throughout the compiler development, not only at the final stage, but also from the very beginning when the grammar was implemented. The approach, is to test each piece of our compiler and sew them together to conduct the final test.

Unit Testing

I tested the lexer by generating *.sal files that didn't conform to each rule specified in the lexer. Test variations:

- Left out braces, parentheses, semicolons
- Left out keywords, built-ins, IDs
- Repeated keywords, built-ins, IDs, parentheses, braces, semicolons
- Input a blank file
- Put in comments
- Misspellings

I tested the parser by generating *.sal files that didn't conform to each rule specified in the parser. Test variations:

- Re order of braces, parentheses, semicolons, keywords, built-ins, IDs
- Misspellings
- Same variations as above

For the Walker, I used the above *.sal files variations and checked the values of the data structures that represented the information required. I included new variations that focused on:

- Valid values for Lines of Businesses
- IT Costs
- Actions

For the Interpreter, I used the valid variations generated above and checked for correct values of:

- Profits
- Costs
- Return on investments
- Projected Forecasts
- Penalties
- Negative and zero values to check boundary conditions

Automated Regression Testing

All programs in the tutorial chapter as well as the above programs were run as part of an automated Regression bucket. This regression harness was written in Perl: RegressionTest.pl. Perl was chosen as a test harness for its superior text processing and system invocation capabilities. The harness generates 150 test cases and checks for correct results. Since the number of transactions for a Line of Business vary depending on the agents that send back different numbers based on actual values (when system

packages are used) or random values (when systems mgmt packages are not used), and forecasted numbers are different based on actual numbers (systems mgmt packages plugged in) or random numbers (no systems mgmt packages) the test harness generates new test cases simply by invoking different types of test cases repeatedly. This causes new arrival rates to be fed into the test harness. The different types of test cases fall into the categories of provisioning for the Line of Business with the largest:

1. Revenue growth or
2. Current Revenue or
3. Forecasted Revenue or
4. Also where no provisioning is specified

Example of a Test case:

```
:ReturnOnInvestment;
{
  :Revenue;
  {
      :LineOfBusiness InternationalNews;
      {
        :MeanTimeToRecovery = 2 minutes;
        :ResponseTime = 5 seconds;
        :Profit = 10 dollars;
        :TxCost = 5 dollars;
        :Penalty = 2 dollars;
      } //End International News

      :LineOfBusiness NationalNews;
      {
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
       } //End National News

      :LineOfBusiness LocalNews;
      {
        :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 20 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700;
        (
            :Penalty = 100 dollars;
        )
       } //End Local News
   } //End Profit

   :ITCosts;
   {
      :Routers = no;
      :Storage = yes;
      :Servers = yes;
```

```
      :Bandwidth = no;
      :Power = yes;
      :Personnel = 200;
   }

   :Actions;
   {
      :Forecast = yes;
      :Optimize = yes;
      :Provision (:Growth);
   }
} //End ROI
```

Example of Results expected:

```
Line Of Business: InternationalNews
Current Arrival Rate: 588
Total Day Profit: 588000
Total Day Cost: 176988
Current Revenue 411012
Forecasted Arrival Rate: 22932
Projected Arrival Rate change 3800%
Forecasted Profit 229320
Projected Profit change -61%
Forecasted Cost 16052
Projected Cost change -90%
Forecasted Revenue 213268
Projected Revenue change -48%

Line Of Business: NationalNews
Current Arrival Rate: 75
Total Day Profit: 45000
Total Day Cost: 22575
Current Revenue 22425
Forecasted Arrival Rate: 1050
Projected Arrival Rate change 1300%
Forecasted Profit 6300
Projected Profit change -86%
Forecasted Cost 210
Projected Cost change -99%
Forecasted Revenue 6090
Projected Revenue change -72%

Line Of Business: LocalNews
Current Arrival Rate: 276
Total Day Profit: 552000
Total Day Cost: 90276
Current Revenue 461724
Forecasted Arrival Rate: 3864
Projected Arrival Rate change 1300%
Forecasted Profit 77280
Projected Profit change -86%
Forecasted Cost 42504
Projected Cost change -52%
Forecasted Revenue 34776
```

```
Projected Revenue change -92%

Total Company Profit 1185000
Total Company Cost 289839
Current Company ROI: 4.088476706033349
Forecasted Company Profit 312900
Forecasted Cost 58766
Forecasted Company ROI: 5.324507368206105
Line of Business with Forecasted Largest Revenue: InternationalNews
213268
Line of Business with Forecasted Largest Growth: InternationalNews -48%
Line of Business with Current Largest Revenue: LocalNews 461724

Provisioning Storage, Network and Server resources for Line of Business
with Largest Growth: InternationalNewsLesson Learned
```

Example of a Test case with Optimization and Forecasting not requested:

```
:ReturnOnInvestment;
{
  :Revenue;
  {
      :LineOfBusiness InternationalNews;
      {
        :MeanTimeToRecovery = 2 minutes;
        :ResponseTime = 5 seconds;
        :Profit = 10 dollars;
        :TxCost = 5 dollars;
        :Penalty = 2 dollars;
      } //End International News

      :LineOfBusiness NationalNews;
      {
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
       } //End National News

      :LineOfBusiness LocalNews;
      {
        :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 20 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700;
        (
            :Penalty = 100 dollars;
        )
       } //End Local News
   } //End Profit

   :ITCosts;
   {
      :Routers = no;
```

```
      :Storage = yes;
      :Servers = yes;
      :Bandwidth = no;
      :Power = yes;
      :Personnel = 200;
   }

   :Actions;
   {
      :Forecast = no;
      :Optimize = no;
      :Provision (:none);
   }
} //End ROI
```

Example of Results expected:

```
Line Of Business: InternationalNews
Current Arrival Rate: 519
Total Day Profit: 519000
Total Day Cost: 156882
Current Revenue 362118

Line Of Business: NationalNews
Current Arrival Rate: 365
Total Day Profit: 219000
Total Day Cost: 109500
Current Revenue 109500

Line Of Business: LocalNews
Current Arrival Rate: 592
Total Day Profit: 1184000
Total Day Cost: 182992
Current Revenue 1001008

Total Company Profit 1922000
Total Company Cost 449374
Current Company ROI: 4.27706097816073
```

Lessons Learned

Usability
Service level agreements are defined to monitor and provide cost savings for customers using existing Information Technology infrastructures. Prior to SAL, customers would have to determine what metrics to measure and where to place monitoring agents to see if they were getting a maximal return on their IT investment. To do this, customers would need detailed knowledge of their IT architectures, application deployment, number of servers, type of network, storage connections. With SAL, customers no longer have to know what their IT infrastructure is composed of. SAL brings the objectives customers would like to achieve up a level, from the IT resource level up to the business requirements. Customers need only state what are the "Business goals" they're trying to achieve (i.e. profit per transaction, response time desired, Businesses to measure) rather than having to determine how these business goals could be translated into paging rates, CPU utilization, I/O rate, etc. The lesson learned here is that a new language can help solve a problem (measuring the cost and value of your IT infrastructure) that could only have been done previously using complex manipulations of agents that obfuscated the goals desired (return on investment) with terms that the only the agents could measure (throughput, CPU utilization).

Performance
SAL could have been designed to calculate and deploy agents as the SAL file was parsed, invoking the functions (1) Deploy Agents (2) Calculate ROI (3) Optimize and Provision as the necessary parameters were parsed. Instead, the design taken was to parse the entire input file first, save the values parsed in the data structures (A) Line Of Business (B) ITCost and (C) Actions To Call. Once these data structures were populated, SAL executed functions 1-3 without any delay due to file reads, and parsing.

Pluggability
All customers with over a hundred servers have already bought into systems management applications to monitor, debug and help plan for future capacity. The data from these systems management applications can be plugged into SAL to help augment values that should be a part of the Return On Investment calculations. This drove the design of SAL to enable (1) Deploy Agents (2) Calculate ROI and (3) Optimize and Provision to be pluggable so that customers could provide their own class implementations or agents to perform these functions.

Test
I had started out using JUNIT an open source java framework for unit testing. However, because the input into SAL is not controlled, transaction rate is based on the actual rate of work in a system at the time or simulated by random number generators, regression testing had to be able to check the results without any dependence on the input values into SAL during real time. The approach taken was to write a Perl program that invokes SAL in real time and checks for the correctness of the values calculated and optimized. Perl was chosen because of its builtin text processing, system and file manipulation capabilities. 4 types of SAL testcases were written and the Perl program invokes these

four types up to 50 times generating 200 test cases. Each test case receives different transaction rates due to the real transaction rate in the system (when pluggable agents are used) or with random number generators (to simulate pluggable agents). The four types of test cases are (1) With no optimization (2) Provisioning for the Line of Business with the largest current revenue (3) Provisiong for the LOB with the largest forecasted revenue (4) Provisioning for the Line of Business with the largest projected growth. Using this test harness, both logic, arithmetic, and syntax bugs were able to be fleshed out.

Appendix A

SAL Grammar
The following lists all of the grammar productions described in the Language Reference section.

SLASpecification -> ReturnOnInvestment declaration
(LineOfBusiness declaration)+
ITCost declaration
Actions declaration

ReturnOnInvestment Declaration -> ':ReturnOnInvestment' ';'
'{'
(':LineOfBusiness' Identifier)+
'}'

Line of Business declaration -> Identifier ';'
':MeanTimeToRecovery' = Integer ';'
':Responsetime' = Integer 'second' 's?' ';'
';'?
':Profit' = Integer 'dollar' 's?' ';'
':TxCost' = Integer 'dollar' 's?' ';'
('For Scope' = Integer 'to' Integer ';)?
'(' ?
':Penalty' = Integer 'dollar' 's?' ';'
')' ?
'}'

IT Costs Declaration -> ':ITCosts' ';'
'{'
':Servers' = 'yes|no' ';'
':Routers' = 'yes|no' ';'
':Storage' = 'yes|no' ';'
':Bandwidth' = 'yes|no' ';'
':Power' = 'yes|no' ';'
':Personnel' = Integer ';'
'}'

Action Declaration -> ':Action' ';'
'{'
':Forecast' = 'yes|no' ';'
':Optimize;' = 'yes|no' ';'
':Provision;' = '('

```
        ':Growth'
        |':ProjectedRevenue'
        |':CurrentRevenue'
        |':none'
    ')' ';'
'}'
```

Code Listings:

```
/*
 * sal.g : the lexer, parser and tree walker, in ANTLR grammar.
 *
 * @author Donna Eng Dillenberger
 *
 */

{
import java.io.*;
import java.util.*;
}

class SALAntlrLexer extends Lexer;

options{
    charVocabulary = '\3'..'\377';
    testLiterals = false;
    exportVocab = SALAntlr;
    k=5;
}

{
    int nr_error = 0;
    public void reportError( String s ) {
        super.reportError( s );
        nr_error++;
    }
    public void reportError( RecognitionException e ) {
        super.reportError( e );
        nr_error++;
    }
}

protected
ALPHA   : 'a'..'z' | 'A'..'Z' | '_' ;

protected
DIGIT   :   '0'..'9';

WS      : (' ' | '\t')+          { $setType(Token.SKIP); }
        ;

NL      : ('\n' | ('\r' '\n') => '\r' '\n' | '\r')
            { $setType(Token.SKIP); newline(); }
        ;

COMMENT : ( "/*" (
                    options {greedy=false;} :
                    (NL)
                    | ~( '\n' | '\r' )
                )* "*/"
            | "//" (~( '\n' | '\r' ))* (NL)
          )                        { $setType(Token.SKIP); }
        ;
```

```
ROI     : ":ReturnOnInvestment";
PROFIT  : ":Profit";
LOB     : ":LineOfBusiness";
MTR     : ":MeanTimeToRecovery";
RT      : ":ResponseTime";
REVENUE : ":Revenue";
TXCOST  : ":TxCost";
PENALTY : ":Penalty";
FSCOPE  : "For Scope";
ITCOSTS : ":ITCosts";
ROUTERS : ":Routers";
STORAGE : ":Storage";
SERVERS : ":Servers";
BANDW   : ":Bandwidth";
POWER   : ":Power";
PERS    : ":Personnel";
ACTIONS : ":Actions";
FCAST   : ":Forecast";
OPTIMIZE  : ":Optimize";
PROVISION : ":Provision";

LPAREN  : '(';
RPAREN  : ')';
SEMI    : ';';
LBRACE  : '{';
RBRACE  : '}';
ASGN    : '=';


ID  options { testLiterals = true; }
        : ALPHA(ALPHA|DIGIT)*
        ;

PROVACT options {testLiterals = true; }
        :
          ( ":CurrentRevenue"
            |":ProjectedRevenue"
            |":Growth"
            |":none"
          )
         ;


/* NUMBER example:
 * 1, 1e, 1., 1.e10, 1.1, 1.1e10
 */

NUMBER  : (DIGIT)+ ('.' (DIGIT)*)? (('E'|'e') ('+'|'-')? (DIGIT)+)? ;

STRING  : '"'!
              (  ~('"' | '\n')
               | ('"'!'"')
              )*
          '"'!
        ;
```

```
/*************************************************************************/
class SALAntlrParser extends Parser;

options{
    buildAST = true;
    k=5;
}


{
    int nr_error = 0;
    public void reportError( String s ) {
        super.reportError( s );
        nr_error++;
    }
    public void reportError( RecognitionException e ) {
        super.reportError( e );
        nr_error++;
    }
}

roi_def
        : ROI^ SEMI!
          LBRACE!
          revenue_def
          itcosts_def
          actions_def
          RBRACE!
        ;

revenue_def
        : REVENUE^  SEMI!
          LBRACE!
          (lob_def)*
          RBRACE!
        ;


lob_def:  LOB^ ID SEMI!
          LBRACE!
          mtr_stmt
          response_stmt
          profit_stmt
          txcost_stmt
          (for_stmt |
           penalty_stmt)
          RBRACE!
        ;
mtr_stmt
        : MTR^ ASGN! NUMBER time! SEMI!
        ;

response_stmt
        : RT^ ASGN! NUMBER time! SEMI!
        ;
```

```
profit_stmt
        : PROFIT^ ASGN! NUMBER currency! SEMI!
        ;

txcost_stmt
        : TXCOST^ ASGN! NUMBER currency! SEMI!
        ;

penalty_stmt
        : PENALTY^ ASGN! NUMBER currency! SEMI!
        ;

for_stmt
        : FSCOPE^ ASGN! NUMBER "to"! NUMBER SEMI! LPAREN!
          penalty_stmt
          RPAREN!
        ;

time    : "minutes"
        | "minute"
        | "seconds"
        | "second"
        ;

currency:
         "dollar"
        |"dollars"
        ;

itcosts_def
        :
          ITCOSTS^  SEMI!
          LBRACE!
          router_stmt
          storage_stmt
          servers_stmt
          bandw_stmt
          power_stmt
          pers_stmt
          RBRACE!
        ;

router_stmt
        : ROUTERS^ ASGN! ID SEMI!
        ;

storage_stmt
        : STORAGE^ ASGN! ID SEMI!
        ;

servers_stmt
        : SERVERS^ ASGN! ID SEMI!
        ;
bandw_stmt
        : BANDW^ ASGN! ID SEMI!
```

```
        ;

power_stmt
        : POWER^ ASGN! ID SEMI!
        ;

pers_stmt
        : PERS^ ASGN! NUMBER SEMI!
        ;

actions_def
        :
          ACTIONS^  SEMI!
          LBRACE!
          forecast_stmt
          optimize_stmt
          provision_stmt
          RBRACE!
        ;

forecast_stmt
        : FCAST^ ASGN! ID SEMI!
        ;

optimize_stmt
        : OPTIMIZE^ ASGN! ID SEMI!
        ;

provision_stmt
        : PROVISION^  LPAREN! PROVACT
         RPAREN! SEMI!
        ;


/**********************************************************************
 * walker.g : the AST walker.
 *
 * @author Donna Eng Dillenberger
 */


class SALAntlrWalker extends TreeParser;
options{
    importVocab = SALAntlr;
}

{
   public static LineOfBusiness [] SpecificLOB;    //Save parsed values
in Line of Business data structures
   public static int LOBInstance;
   public static ITCost IT;                         //Save parsed values
in IT components data structures
   public static int ITInstance;
   public static String [] ActionsToCall;          //Save parsed values
in Actions data structures
   public static int ActInstance;
   public void initialize()
```

```
    //Initialize above data structures to prepare for parsing
    {
      LOBInstance = -1;
      SpecificLOB = new LineOfBusiness[3];
      for (int i=0; i<3; i++)
      {  SpecificLOB[i] = new LineOfBusiness();  }

      IT = new ITCost();
      ITInstance = 0;

      ActionsToCall = new String[3];
      ActInstance = 0;


   } // end initialize
} // end global definitions


roi_def returns [String s]
{
  s = null;
  String mc,rc,pc,tc,l1,l2,l3,pec,si,sa,sc,stc,bc,poc,perc,sac,fc,oc =
null;
}
       // Parsed values have all been collected, call the Interpreter
       // to run the user's program
       : #(ro:ROI s= roi_def si=roi_def  sac=roi_def  )
       {
         SALInterpreter.Interpretmain (SpecificLOB, IT, ActionsToCall);
       }

       | #(re:REVENUE l1=roi_def l2=roi_def l3=roi_def)


       | #(lo:LOB l:ID mc=roi_def rc=roi_def pc=roi_def tc=roi_def
pec=roi_def)
                                // Save Line of Business name
                                {  SpecificLOB[LOBInstance].LOBName = l.getText();

                                }


      | #(m:MTR mn:NUMBER)
                                 // Save Mean Time To Recovery goals
                                 { LOBInstance = LOBInstance + 1;
                                    SpecificLOB[LOBInstance].MeanTimeToRecoveryGoal =
                                   (int)(Integer.parseInt(mn.getText()));
                                 }

       | #(r:RT rn:NUMBER)       // Save Response time Goals
                                 {  SpecificLOB[LOBInstance].ResponseTimeGoal =
                                  (int)(Integer.parseInt(rn.getText()));

                                 }

       | #(pr:PROFIT pn:NUMBER) // Save Profit per transaction
                                 { SpecificLOB[LOBInstance].Profit =
                                        (int)(Integer.parseInt(pn.getText()));

                                 }
```

```
| #(tx:TXCOST tn:NUMBER) //Save Cost per transaction
                         {
                           SpecificLOB[LOBInstance].TxCost =
                                (int)(Integer.parseInt(tn.getText()));
                         }

| #(pe:PENALTY pen:NUMBER) //Save Penalty
                         {
                           SpecificLOB[LOBInstance].Penalty =
                                (int)(Integer.parseInt(pen.getText()));
                         }

| #(fo:FSCOPE f1n:NUMBER f2n:NUMBER s=roi_def)
                         { // Save time scope for penalty
                           SpecificLOB[LOBInstance].PenaltyBegin =
                                (int)(Integer.parseInt(f1n.getText()));

                           SpecificLOB[LOBInstance].PenaltyEnd =
                                (int)(Integer.parseInt(f2n.getText()));
                          }



| #(it:ITCOSTS rc=roi_def sc=roi_def stc=roi_def bc=roi_def
    poc=roi_def perc=roi_def)


| #(rou:ROUTERS rid:ID)  { // Charge for Routers
                           IT.Expense[ITInstance] = rid.getText();
                           ITInstance = ITInstance +1;

                         }

| #(sto:STORAGE stid:ID)
                         { // Charge for Storage
                           IT.Expense[ITInstance] = stid.getText();
                           ITInstance = ITInstance +1;
                         }


| #(ser:SERVERS serid:ID)
                         { // Charge for servers
                           IT.Expense[ITInstance] = serid.getText();
                           ITInstance = ITInstance +1;
                         }

| #(ba:BANDW bid:ID)
                         { // Charge for Bandwidth
                           IT.Expense[ITInstance] = bid.getText();
                           ITInstance = ITInstance +1;


                         }
| #(pow:POWER pid:ID)
                         { //Charge for Power
                           IT.Expense[ITInstance] = pid.getText();
                           ITInstance = ITInstance +1;

                         }

| #(per:PERS pers:NUMBER)
                         { //Charge for Personnel
```

```
                            IT.Personnel =
                               (int)(Integer.parseInt(pers.getText()));
                          }

| #(ac:ACTIONS fc=roi_def oc=roi_def pc=roi_def)


| #(fcastid:FCAST fid:ID)
                          { //Save Forecasting option
                            ActionsToCall[ActInstance] = fid.getText();
                            ActInstance = ActInstance +1;

                          }

| #(opt:OPTIMIZE oid:ID)
                          { //Save Optimize option
                            ActionsToCall[ActInstance] = oid.getText();
                            ActInstance = ActInstance +1;
                          }


| #(prov:PROVISION prid:PROVACT)
                          { //Save Provisioning option
                            ActionsToCall[ActInstance] = prid.getText();

                          }
;
```

```
/**
 * SALMain
 *
 * @author Donna Eng Dillenberger
 */

import java.io.*;
import antlr.CommonAST;
import antlr.collections.AST;
import antlr.RecognitionException;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;


class SALMain {

    public static void execFile( String filename ) {
        try
        {
            InputStream input = ( null != filename ) ?
                (InputStream) new FileInputStream( filename ) :
                (InputStream) System.in;

            SALAntlrLexer lexer = new SALAntlrLexer( input );

            SALAntlrParser parser = new SALAntlrParser( lexer );
            parser.setFilename(filename);

            // Parse the input program
            parser.roi_def();

            if ( lexer.nr_error > 0 || parser.nr_error > 0 )
            {
                System.err.println( "Parsing errors found. Stop." );
                return;
            }

            CommonAST tree = (CommonAST)parser.getAST();


            SALAntlrWalker walker = new SALAntlrWalker();
            // Traverse the tree created by the parser

            walker.initialize();
            String r = walker.roi_def( tree );


        } catch( IOException e ) {
            System.err.println( "Error: I/O: " + e );
        } catch( RecognitionException e ) {
            System.err.println( "Error: Recognition: " + e );
        } catch( TokenStreamException e ) {
            System.err.println( "Error: Token stream: " + e );
        } catch( Exception e ) {
            System.err.println( "Error: " + e );
        }
```

```
    }


    public static void main( String[] args ) {


        if ( args.length >= 1 && args[args.length-1].charAt(0) != '-' )
            execFile( args[args.length-1] );

        System.exit( 0 );
    }
}
```

```
/*************************************************************************/
/* Donna Eng Dillenberger, class: COMSW4115
/*
/* Name: SALInterpreter.java
/*
/* Tested with: Javasoft JDK 1.3.1 on Windows XP
/*
/* To compile: javac SALInterpreter.java
/* Must have the following files in your classpath:  LineOfBusiness.java
/*                                                   ITCost.java
/*
/*
/*
/*
/* Function: This program is called by SALAntlrWalker.java after
/*           SALAntlrWalker.java parses a Service Agreement Language (SAL) file
/*           A SAL file specifies the Lines of Businesses a company has and the
/*           profits and costs per transaction for each Line of Business (LOB).
/*           Adminstrators would use the syntax, operators and built in methods
/*           of the SAL language to define Service Level Agreements and
/*           Information Technology (IT) monitoring, forecasting and
/*           optimization agents that would be deployed in the IT
/*           infrastructure to monitor whether a Business is achieving
/*           their business goals.  SALAntlrLexer.java then scans the *.sal
/*           file performing lexical checking and parsing(SALAntlrParser.java).
/*           SALAntlrWalker then stores the parsed values in the Lines of
/*           Businesses, Actions and ITCost data structures and passes them to
/*           this program.  This program will:
/*           •      Deploy monitoring agents (calls to interfaces) to get
/*              NumberOfTransactions/day for each LineOfBusiness.
/*           •      Calculate ReturnOnInvestments = Sum of Profits from each
/*              LineOfBusiness transaction.
/*           •      Obtain ITCosts for each Line of Business transaction by
/*              deploying Monitoring agents on each ITCosts member
/*              (call to an interface)
/*           •      Use the above data from the Monitoring agents to calculate
/*              IT costs per Application
/*           •      Support the verbs FORECAST, PROVISION, OPTIMIZE by calling
/*              the correct methods and generating the data needed by each
/*              method
/*
 */

import java.io.*;


/***************** Main Program ****************************************/
public class SALInterpreter
{

    /***************** Global Variables ******************************/
   public static ITCost IT;                         //Class holding parsed IT
components to charge
   public static LineOfBusiness [] SpecificLOB;    //Classes holding lines of
businesses costs and profits
   public static String [] ActionsToCall;          //Class holding parsed
actions to execute
   public static int AllLOBProfit;                 //Total Company's profits
   public static int AllLOBCost;                   //Total Company's costs
   public static int AllLOBPenalty;                //Total Company's penalties
   public static double ReturnOnInvestment;        //ROI =
Profits/(Costs+Penalties)
```

```
   public static int OptimalLOB;                    //The LOB with the largest
revenue
   public static int OptimalGain;                   //The largest current
revenue amongst all LOBs
   public static int ForecastAllProfit;             //Forecasted profit for
entire company
   public static int ForecastAllCost;               //Forecasted cost for entire
company
   public static double ForecastROI;                //Forecasted Return On
Investment
   public static int ForecastOptimalLOB;            //LOB with largest
forecasted revenue
   public static int ForecastOptimalGain;           //Largest forecasted revenue
   public static double OptimalChange;              //Largest forecasted revenue
growth
   public static int OptimalChangeLOB;              //LOB with largest
forecasted revenue growth


   /****************** Global Constants *****************************/
   public final static int LOBMax = 3;              //Lines of Businesses
   public final static int ITComponentMax = 5;      //Maximum IT components
   public final static int ITSalary = 15;           //Cost of IT personnel per
month
   public final static int ForecastAction = 0;      //Constant for forecast
action
   public final static int OptimizeAction = 1;      //Constant for optimize
action
   public final static int ProvisionAction = 2;     //Constant for provision
action


    /******************** Start of Main ***************************/
    /* This method is called by SALAntlrWalker.java after SALAntlrWalker.java
has
    /* parsed a *.sal file
     */
    public static void Interpretermain (LineOfBusiness [] SpecificLOB, ITCost
IT, String [] ActionsToCall)
    {

      InitializeGlobalValues();
      DeployAgents(SpecificLOB, IT);
      CalculateROI (SpecificLOB, ActionsToCall);
      ProvisionOptimize (SpecificLOB, ActionsToCall);

    } // end ProfMain

    /*****************************************************************/



    /*****************************************************************/
    private static void InitializeGlobalValues()
    {
      AllLOBProfit = 0;                   //Total Company's profits
      AllLOBCost = 0;                     //Total Company's costs
      AllLOBPenalty = 0;                  //Total Company's penalties
      ReturnOnInvestment = 0;             //ROI = Profits/(Costs+Penalties)
      OptimalLOB = 0;                     //The LOB with the largest revenue
      OptimalGain = -999999999;           //The largest current revenue amongst
all LOBs
```

```
      ForecastAllProfit = 0;              //Forecasted profit for entire
company
      ForecastAllCost = 0;                //Forecasted cost for entire company
      ForecastROI = 0;                    //Forecasted Return On Investment
      ForecastOptimalLOB = 0;             //LOB with largest forecasted revenue
      ForecastOptimalGain = -999999999;   //Largest forecasted revenue
      OptimalChange = -999999999;         //Largest forecasted revenue growth
      OptimalChangeLOB = 0;               //LOB with largest forecasted revenue
growth
    }




   /***************************** DeployAgents *****************************/
   /* For each Line of Business, call a pluggable class to obtain transactions
per day,
    * TotalDayProfit, TotalDayCost and Penalties incurred by missed SLA rules.
    *
    * Parameters: Lines Of Business, IT Costs
    */

    private static void DeployAgents(LineOfBusiness[] SpecificLOB,ITCost IT)
    {
      int ActualCost;

      try {

        for (int i= 0; i < LOBMax; i++)
         {
              SpecificLOB[i].TransactionsPerDay = GetTxPerLOB(i);
              SpecificLOB[i].TotalDayProfit = SpecificLOB[i].TransactionsPerDay
*
                                              SpecificLOB[i].Profit * 100;

              ActualCost = GetTxCostLOB(IT);
              SpecificLOB[i].TotalDayCost = SpecificLOB[i].TransactionsPerDay *
                                              ActualCost;
              GetPenaltyPayment(SpecificLOB, i);
              SpecificLOB[i].CurrentRevenue = SpecificLOB[i].TotalDayProfit -
                                              SpecificLOB[i].TotalDayCost;


         }  // end for Deploy Agents for all Lines of Businesses

      } //end try
    catch (Exception e)
    {       System.err.println(e);  }


    } // end Deploy Agents

    /********************** end Deploy Agents*******************************/


    /****************** GetTxPerLOB ************************************
     * Return number of transactions for this Line of Business
     *
     * parameter = int: the Line of Businesss to obtain transactins for
     *
     *
```

```
     * return value = int: Total number of tx in the time range requested for
     *                     the inputed Line of business
     */

    private static int GetTxPerLOB (int LOBInstance)
                                            throws IllegalArgumentException
    {
      int TotalTx =0;
      return (TotalTx = (int)(Math.random() * 1000));

      // For a product, instead of returning a Random number, this method
      // would call on pluggable classes that deploys Monitoring agents for
      // specific Lines of Businesses.  These Monitoring agents would collect
      // number of transactions executed for a Line of Business during a
      // certain time or time range of a business day.


    } //end GetTxPerLOB
    /************************ End of GetTxPerLOB ************************/


    /***************** GetTxCostPerLOB ***************************************
     * Return Information Technology (IT) cost for a transaction for each Line
of Business
     *
     * parameter = int: the Line of Businesss to obtain IT cost for
     *
     *
     * return value = int: Total IT cost of tx
     */

    private static int GetTxCostLOB (ITCost IT)
                                            throws IllegalArgumentException
    {
      int TotalTxCost =0;

      for (int i=0; i<ITComponentMax; i++)
      {
          if (IT.Expense[i].equals("yes"))
          { TotalTxCost = TotalTxCost + DeployITAgent(i); }
      }
      if (IT.Personnel > 0) {TotalTxCost = TotalTxCost + (IT.Personnel *
ITSalary);}
      TotalTxCost = TotalTxCost/10;
      return (TotalTxCost);


    } //end GetTxCostPerLOB


    /****************** DeployITAgent ************************************
     * Install and Activate Agent that will return the cost of the IT resource
and
     * how much of that IT resource is being used for a Line of Business
     *
     * parameter = int: the Resource to measure
     *
     *
     *
     * return value = int: Total IT cost of
     */

    private static int DeployITAgent (int Resource)
```

```
                                            throws IllegalArgumentException
    {
      int ITComponentCost =0;

      ITComponentCost = (int) (Math.random()*10);
      return (ITComponentCost);

      // For a product, instead of returning a Random number, this method
      // would call on pluggable classes that deploys Monitoring agents for
      // the specific resource to measure the amount a Line of Businesses is
using.
      // These Monitoring agents would collect the IT Cost for this resource
for the
      // inputed Line of Business during a certain time or time range of a
business day.


    } //end DeployITAgent

    /***************** GetPenaltyPayment  ********************************
     * Obtain data from agents on when Mean Time to Recovery goals were missed
     * and for how long.  Calculate the penalty incurred if Mean time to
recovery
     * goals were missed.
     *
     * parameter = int: the Line Of Business
     */

    private static void GetPenaltyPayment (LineOfBusiness [] SpecificLOB, int
LOBIndex)
                                            throws IllegalArgumentException
    {
      int Penalty =0;
      int TimeScope = 0;

      TimeScope = SpecificLOB[LOBIndex].PenaltyEnd -
SpecificLOB[LOBIndex].PenaltyBegin;
      if (TimeScope <= 0) {TimeScope = 24;}
      else {TimeScope = TimeScope/100;}
      SpecificLOB[LOBIndex].MeanTimeToRecoveryMissed = (int)(Math.random()*10);
      SpecificLOB[LOBIndex].ResponseTimeMissed = (int)(Math.random()*10);
      SpecificLOB[LOBIndex].TotalDayPenalty =
            (SpecificLOB[LOBIndex].MeanTimeToRecoveryMissed +
             SpecificLOB[LOBIndex].ResponseTimeMissed) *
            (SpecificLOB[LOBIndex].Penalty * TimeScope);
      SpecificLOB[LOBIndex].TotalDayCost = SpecificLOB[LOBIndex].TotalDayCost +

SpecificLOB[LOBIndex].TotalDayPenalty;

      // For a product, instead of returning a Random number, this method
      // would call on pluggable classes that deploys Monitoring agents for
      // the specific resource to measure the amount of time a goal is missed.
      // These Monitoring agents would check the time of day and note when
specific
      // time ranges a goal applies to.

    } //end GetPenaltyPayment

    /*************************************************************/

    /***************** Forecast ******************************
     * Project from the current arrival rate what the future arrival rate,
     * future costs, profits and penalty will be.
```

```
     *
     * parameter = int: the Line Of Business
     *
     */

    private static void Forecast (LineOfBusiness [] SpecificLOB, int LOBIndex)
throws IllegalArgumentException
    {
      SpecificLOB[LOBIndex].ForecastArrivalRate =
SpecificLOB[LOBIndex].TransactionsPerDay
                                                * (int)(Math.random()*100) ;
      SpecificLOB[LOBIndex].ForecastTotalDayPenalty =
SpecificLOB[LOBIndex].Penalty *

SpecificLOB[LOBIndex].ForecastArrivalRate/10;
      SpecificLOB[LOBIndex].ForecastTotalDayProfit =
SpecificLOB[LOBIndex].Profit *

SpecificLOB[LOBIndex].ForecastArrivalRate;
      SpecificLOB[LOBIndex].ForecastTotalDayCost =
(SpecificLOB[LOBIndex].TxCost *

SpecificLOB[LOBIndex].ForecastArrivalRate/10)+

SpecificLOB[LOBIndex].ForecastTotalDayPenalty;
      SpecificLOB[LOBIndex].ForecastRevenue =
SpecificLOB[LOBIndex].ForecastTotalDayProfit -

SpecificLOB[LOBIndex].ForecastTotalDayCost;



      // For a product, instead of using a Random number, to forecast
      // an arrival rate, this method would use Time Series, M5 model
      // trees, or Multiple Regression methods to project future arrival rates.
      // These methods would be vendor supplied pluggable classes.

    } //end Forecast



    /****************** Provision  ********************************
     * Provision the resources needed for another instantiation of a
     * Line of Business' applications.
     *
     * parameter = int: the Line Of Business to provision storage, server,
     *             and network resources for
     *
     */

    private static void Provision (LineOfBusiness[] SpecificLOB, String Value,
int LOBIndex)
                                                        throws
IllegalArgumentException
    {
      System.out.println(
      "Provisioning Storage, Network and Server resources for Line of Business
with Largest "
             + Value + ": " + SpecificLOB[LOBIndex].LOBName);


      // For a product, instead of printing a statement, this method would
      // call on vendor supplied pluggable classes to configure and deploy
```

```
      // new Servers, Storage and Network resources to instantiate additional
      // IT resources required to support this Line of Business

    } //end Provision

    /******************** end provision ************************************/


    /************************ GetGrowth ***********************************/
    private static double GetGrowth (int x, int y)
    {
      double growth = 0;

      growth = y-x;
      if (x <0) {x = x * -1;}
      growth = (growth/(double) x) * 100;
      return (growth);

    }

/************************** End of GetGrowth*****************************/



/************************** Calculate ROI ******************************/
private static void CalculateROI(LineOfBusiness [] SpecificLOB, String[]
ActionsToCall)
{
      // Calculate Current and Forecasted Profit, Cost, Penalty for each
      // Line of Business
      for (int i=0; i< LOBMax; i++)  //For all Lines of Businesses
      {
             //Calculate company's profit and cost
        AllLOBProfit = AllLOBProfit + SpecificLOB[i].TotalDayProfit;
             AllLOBCost = AllLOBCost + SpecificLOB[i].TotalDayCost;

             //If SAL file specified Forecast = yes
             if (ActionsToCall[ForecastAction].equals("yes"))
             {
               Forecast(SpecificLOB,i);    //get projected arrival rates,
profits, costs.
              ForecastAllProfit = ForecastAllProfit +
SpecificLOB[i].ForecastTotalDayProfit;
              ForecastAllCost = ForecastAllCost +
SpecificLOB[i].ForecastTotalDayCost;
              SpecificLOB[i].ProjectedChange = GetGrowth
(SpecificLOB[i].CurrentRevenue,

SpecificLOB[i].ForecastRevenue);
              //Calculate largest forecasted revenue
              if (SpecificLOB[i].ForecastRevenue > ForecastOptimalGain)
              {
                  ForecastOptimalGain = SpecificLOB[i].ForecastRevenue;
                 ForecastOptimalLOB = i;
              } // end look for Optimal Gain

              //Calculate largest forecasted growth in revenue
              if (SpecificLOB[i].ProjectedChange > OptimalChange)
              {
                 OptimalChange = SpecificLOB[i].ProjectedChange;
                OptimalChangeLOB = i;
              } // end look for Optimal Change
             } // end if Forecast = yes
```

```
         //This code is put in here if the *.sal file contained the Action word
         // "Optimize = yes"
         if (ActionsToCall[OptimizeAction].equals("yes")) {
             //Calculate line of business with largest revenue
             if (SpecificLOB[i].CurrentRevenue > OptimalGain)
             {
             OptimalGain = SpecificLOB[i].CurrentRevenue;
             OptimalLOB = i;
             }
         } //If Optimize = yes

         //Print results
         System.out.println("Line Of Business: " + SpecificLOB[i].LOBName);
         System.out.println("Current Arrival Rate: " +
SpecificLOB[i].TransactionsPerDay);
         System.out.println("Total Day Profit: " +
SpecificLOB[i].TotalDayProfit);
         System.out.println("Total Day Cost: " + SpecificLOB[i].TotalDayCost);
         System.out.println("Current Revenue " +
SpecificLOB[i].CurrentRevenue);
         if (ActionsToCall[ForecastAction].equals("yes"))
         {
             System.out.println("Forecasted Arrival Rate: " +
SpecificLOB[i].ForecastArrivalRate);
             System.out.println("Projected Arrival Rate change " +
(int)GetGrowth(SpecificLOB[i].TransactionsPerDay,

SpecificLOB[i].ForecastArrivalRate)+"%");

             System.out.println("Forecasted Profit " +
SpecificLOB[i].ForecastTotalDayProfit);
             System.out.println("Projected Profit change " +
(int)GetGrowth(SpecificLOB[i].TotalDayProfit,

SpecificLOB[i].ForecastTotalDayProfit)+"%");

             System.out.println("Forecasted Cost " +
SpecificLOB[i].ForecastTotalDayCost);
             System.out.println("Projected Cost change " +
(int)GetGrowth(SpecificLOB[i].TotalDayCost,

SpecificLOB[i].ForecastTotalDayCost)+"%");
             System.out.println("Forecasted Revenue " +
SpecificLOB[i].ForecastRevenue);
             System.out.println("Projected Revenue change " +
(int)SpecificLOB[i].ProjectedChange+"%");
         } // end if Forecast
         System.out.println();
      }  // end For loop to calculate Profits, Cost and Penalty for all Lines
of Businesses

      ReturnOnInvestment = (double)AllLOBProfit/(double)AllLOBCost;


      //Print ROI
      System.out.println("Total Company Profit " + AllLOBProfit);
      System.out.println("Total Company Cost " + AllLOBCost);
      System.out.println("Current Company ROI: " + ReturnOnInvestment);
      if (ActionsToCall[ForecastAction].equals("yes"))
      {
       System.out.println("Forecasted Company Profit " + ForecastAllProfit);
       System.out.println("Forecasted Cost " + ForecastAllCost);
```

```
       ForecastROI = (double)ForecastAllProfit/(double)ForecastAllCost;
       System.out.println("Forecasted Company ROI: " + ForecastROI);
            System.out.println("Line of Business with Forecasted Largest
Revenue: "
                                  + SpecificLOB[ForecastOptimalLOB].LOBName + "
"
                                  +
SpecificLOB[ForecastOptimalLOB].ForecastRevenue);
            System.out.println("Line of Business with Forecasted Largest
Growth: "
                                  + SpecificLOB[OptimalChangeLOB].LOBName + " "
                                  +
(int)SpecificLOB[OptimalChangeLOB].ProjectedChange+"%");
     } // end forecast actions

} //end Calculate ROI
/*************************************************************************/

/*************************************************************************/

private static void ProvisionOptimize(LineOfBusiness[] SpecificLOB, String[]
ActionsToCall)
{
// Print Optimal LOB
      if (ActionsToCall[OptimizeAction].equals("yes")) {
            System.out.println("Line of Business with Current Largest Revenue:
"
                                  + SpecificLOB[OptimalLOB].LOBName + " "
                                  + SpecificLOB[OptimalLOB].CurrentRevenue);
      }

      System.out.println();

      //This code is put in here if the *.sal file contained the Action word
      // "Provision"

      if (ActionsToCall[OptimizeAction].equals("yes"))
      {  if (ActionsToCall[ProvisionAction].equals(":CurrentRevenue"))
            //Provision to LOB with the largest current revenue
            {Provision(SpecificLOB,"Current Revenue",OptimalLOB);}

         if (ActionsToCall[ProvisionAction].equals(":Growth"))
            //Provision to LOB with largest revenue growth
            {Provision(SpecificLOB,"Growth",OptimalChangeLOB);}

         if (ActionsToCall[ForecastAction].equals("yes"))
         {   if (ActionsToCall[ProvisionAction].equals(":ForecastedRevenue"))
            //Provision to LOB with largest forecasted revenue
            {Provision(SpecificLOB,"Forecasted Revenue",ForecastOptimalLOB);}
         }

      }// end if Optimize

}// end Provision and Optimize


} // end SALInterpreter
/****************************** end program **********************/
```

```
/*************************************************************************
*/
/* Donna Eng Dillenberger, class: COMSW4115
/*
/* Tested with: Windows XP
/*              JDK Level Tested with: JDK 1.3.1
/*
/* To compile this program, issue: javac LineOfBusiness.java
/*
/* The following files must be in your classpath for this class to
compile:
/*     - JDK 1.3.1
/*
/* Function: This class is used to calculate costs and profits for a
Line of Business.
/*
/*/

public class LineOfBusiness
{
          int TransactionsPerDay;
          int Profit;
          int TxCost;
          int TotalDayProfit;
          int TotalDayCost;
          int MeanTimeToRecoveryGoal;
          int MeanTimeToRecoveryMissed;
          int ResponseTimeGoal;
          int ResponseTimeMissed;
          int Penalty;
          int TotalDayPenalty;
          int PenaltyBegin;
          int PenaltyEnd;
          int ForecastArrivalRate;
          int ForecastTotalDayPenalty;
          int ForecastTotalDayProfit;
          int ForecastTotalDayCost;
          int ForecastRevenue;
          int CurrentRevenue;
          double ProjectedChange;
          String LOBName;

   // Constructor
   LineOfBusiness ()
   {
          TransactionsPerDay = 0;
          Profit = 0;
          TxCost = 0;
          TotalDayProfit = 0;
          TotalDayCost = 0;
          MeanTimeToRecoveryGoal = 0;
          MeanTimeToRecoveryMissed = 0;
          ResponseTimeGoal = 0;
          ResponseTimeMissed = 0;
          Penalty = 0;
          TotalDayPenalty = 0;
          PenaltyBegin = 0;
```

```
            PenaltyEnd = 0;
            ForecastArrivalRate = 0;
            ForecastTotalDayPenalty = 0;
            ForecastTotalDayProfit = 0;
            ForecastTotalDayCost =0 ;
            ForecastRevenue = 0;
            CurrentRevenue = 0;
            ProjectedChange = 0;
            LOBName = null;
    }
}
```

```
/*************************************************************************
*/
/* Donna Eng Dillenberger,  class: COMS W4115
/*
/* Tested with: Windows XP
/*              JDK Level Tested with: JDK 1.3.1
/*
/*
/* The following files must be in your classpath for this class to
compile:
/*     - JDK 1.3.1
/*
/* Function: This class consists of an array, where each element
/*           is a component of IT Costs as defined in the *.SAL file.
/*
public class ITCost
{
        String [] Expense;
        int Personnel;

        //Constructor
        ITCost ()
        {
            Expense = new String[5];

            //Each of the elements below are obtained from parsing .sal
Adminstrator written file
            Expense [0] = null;     // Routers
            Expense [1] = null;     // Storage
            Expense [2] = null;     // Servers
            Expense [3] = null;     // Bandwidth
            Expense [4] = null;     // Power
            Personnel = 0;     // Personnel

        }
}
```

Generated Code from Antlr:

```
// $ANTLR 2.7.2: "sal.g" -> "SALAntlrLexer.java"$

public interface SALAntlrWalkerTokenTypes {
	int EOF = 1;
	int NULL_TREE_LOOKAHEAD = 3;
	int ALPHA = 4;
	int DIGIT = 5;
	int WS = 6;
	int NL = 7;
	int COMMENT = 8;
	int ROI = 9;
	int PROFIT = 10;
	int LOB = 11;
	int MTR = 12;
	int RT = 13;
	int REVENUE = 14;
	int TXCOST = 15;
	int PENALTY = 16;
	int FSCOPE = 17;
	int ITCOSTS = 18;
	int ROUTERS = 19;
	int STORAGE = 20;
	int SERVERS = 21;
	int BANDW = 22;
	int POWER = 23;
	int PERS = 24;
	int ACTIONS = 25;
	int FCAST = 26;
	int OPTIMIZE = 27;
	int PROVISION = 28;
	int LPAREN = 29;
	int RPAREN = 30;
	int SEMI = 31;
	int LBRACE = 32;
	int RBRACE = 33;
	int ASGN = 34;
	int ID = 35;
	int PROVACT = 36;
	int NUMBER = 37;
	int STRING = 38;
	int LITERAL_to = 39;
	int LITERAL_minutes = 40;
	int LITERAL_minute = 41;
	int LITERAL_seconds = 42;
	int LITERAL_second = 43;
	int LITERAL_dollar = 44;
	int LITERAL_dollars = 45;
}
// $ANTLR 2.7.2: "sal.g" -> "SALAntlrLexer.java"$

import java.io.InputStream;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;
import antlr.TokenStreamRecognitionException;
```

```
import antlr.CharStreamException;
import antlr.CharStreamIOException;
import antlr.ANTLRException;
import java.io.Reader;
import java.util.Hashtable;
import antlr.CharScanner;
import antlr.InputBuffer;
import antlr.ByteBuffer;
import antlr.CharBuffer;
import antlr.Token;
import antlr.CommonToken;
import antlr.RecognitionException;
import antlr.NoViableAltForCharException;
import antlr.MismatchedCharException;
import antlr.TokenStream;
import antlr.ANTLRHashString;
import antlr.LexerSharedInputState;
import antlr.collections.impl.BitSet;
import antlr.SemanticException;

import java.io.*;
import java.util.*;

public class SALAntlrLexer extends antlr.CharScanner implements
SALAntlrTokenTypes, TokenStream
 {

    int nr_error = 0;
    public void reportError( String s ) {
        super.reportError( s );
        nr_error++;
    }
    public void reportError( RecognitionException e ) {
        super.reportError( e );
        nr_error++;
    }
public SALAntlrLexer(InputStream in) {
      this(new ByteBuffer(in));
}
public SALAntlrLexer(Reader in) {
      this(new CharBuffer(in));
}
public SALAntlrLexer(InputBuffer ib) {
      this(new LexerSharedInputState(ib));
}
public SALAntlrLexer(LexerSharedInputState state) {
      super(state);
      caseSensitiveLiterals = true;
      setCaseSensitive(true);
      literals = new Hashtable();
      literals.put(new ANTLRHashString("minutes", this), new
Integer(40));
      literals.put(new ANTLRHashString("seconds", this), new
Integer(42));
      literals.put(new ANTLRHashString("dollar", this), new
Integer(44));
```

```
      literals.put(new ANTLRHashString("minute", this), new
Integer(41));
      literals.put(new ANTLRHashString("dollars", this), new
Integer(45));
      literals.put(new ANTLRHashString("second", this), new
Integer(43));
      literals.put(new ANTLRHashString("to", this), new Integer(39));
}

public Token nextToken() throws TokenStreamException {
      Token theRetToken=null;
tryAgain:
      for (;;) {
            Token _token = null;
            int _ttype = Token.INVALID_TYPE;
            resetText();
            try {   // for char stream error handling
                  try {   // for lexical error handling
                        switch ( LA(1)) {
                        case '\t':  case ' ':
                        {
                              mWS(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '\n':  case '\r':
                        {
                              mNL(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '/':
                        {
                              mCOMMENT(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '(':
                        {
                              mLPAREN(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case ')':
                        {
                              mRPAREN(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case ';':
                        {
                              mSEMI(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '{':
                        {
```

```
                              mLBRACE(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '}':
                        {
                              mRBRACE(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '=':
                        {
                              mASGN(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '0':  case '1':  case '2':  case '3':
                        case '4':  case '5':  case '6':  case '7':
                        case '8':  case '9':
                        {
                              mNUMBER(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        case '"':
                        {
                              mSTRING(true);
                              theRetToken=_returnToken;
                              break;
                        }
                        default:
                              if ((LA(1)==':') && (LA(2)=='P') &&
(LA(3)=='r') && (LA(4)=='o') && (LA(5)=='f')) {
                                    mPROFIT(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') && (LA(2)=='P') &&
(LA(3)=='r') && (LA(4)=='o') && (LA(5)=='v')) {
                                    mPROVISION(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') &&
(_tokenSet_0.member(LA(2))) && (LA(3)=='o'||LA(3)=='r'||LA(3)=='u') &&
(LA(4)=='n'||LA(4)=='o'||LA(4)=='r') && (_tokenSet_1.member(LA(5)))) {
                                    mPROVACT(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') && (LA(2)=='R') &&
(LA(3)=='e') && (LA(4)=='t')) {
                                    mROI(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') && (LA(2)=='R') &&
(LA(3)=='e') && (LA(4)=='s')) {
                                    mRT(true);
                                    theRetToken=_returnToken;
                              }
```

```
                                else if ((LA(1)==':') && (LA(2)=='R') &&
(LA(3)=='e') && (LA(4)=='v')) {
                                        mREVENUE(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='P') &&
(LA(3)=='e') && (LA(4)=='n')) {
                                        mPENALTY(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)=='F') && (LA(2)=='o') &&
(LA(3)=='r') && (LA(4)==' ')) {
                                        mFSCOPE(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='P') &&
(LA(3)=='o') && (LA(4)=='w')) {
                                        mPOWER(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='P') &&
(LA(3)=='e') && (LA(4)=='r')) {
                                        mPERS(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='R') &&
(LA(3)=='o')) {
                                        mROUTERS(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='S') &&
(LA(3)=='t')) {
                                        mSTORAGE(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='S') &&
(LA(3)=='e')) {
                                        mSERVERS(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='L')) {
                                        mLOB(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='M')) {
                                        mMTR(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='T')) {
                                        mTXCOST(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='I')) {
                                        mITCOSTS(true);
                                        theRetToken=_returnToken;
                                }
                                else if ((LA(1)==':') && (LA(2)=='B')) {
```

```
                                    mBANDW(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') && (LA(2)=='A')) {
                                    mACTIONS(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') && (LA(2)=='F')) {
                                    mFCAST(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((LA(1)==':') && (LA(2)=='O')) {
                                    mOPTIMIZE(true);
                                    theRetToken=_returnToken;
                              }
                              else if ((_tokenSet_2.member(LA(1))) &&
(true) && (true) && (true)) {

                                    mID(true);
                                    theRetToken=_returnToken;
                              }
                        else {
                              if (LA(1)==EOF_CHAR) {uponEOF();
_returnToken = makeToken(Token.EOF_TYPE);}
                        else {throw new
NoViableAltForCharException((char)LA(1), getFilename(), getLine(),
getColumn());}

                        }
                        }
                        if ( _returnToken==null ) continue tryAgain; //
found SKIP token
                        _ttype = _returnToken.getType();
                        _returnToken.setType(_ttype);
                        return _returnToken;
                  }
                  catch (RecognitionException e) {
                        throw new TokenStreamRecognitionException(e);
                  }
            }
            catch (CharStreamException cse) {
                  if ( cse instanceof CharStreamIOException ) {
                        throw new
TokenStreamIOException(((CharStreamIOException)cse).io);
                  }
                  else {
                        throw new
TokenStreamException(cse.getMessage());
                  }
            }
      }
}

      protected final void mALPHA(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ALPHA;
```

```
            int _saveIndex;

            switch ( LA(1)) {
            case 'a':  case 'b':  case 'c':  case 'd':
            case 'e':  case 'f':  case 'g':  case 'h':
            case 'i':  case 'j':  case 'k':  case 'l':
            case 'm':  case 'n':  case 'o':  case 'p':
            case 'q':  case 'r':  case 's':  case 't':
            case 'u':  case 'v':  case 'w':  case 'x':
            case 'y':  case 'z':
            {
                  matchRange('a','z');
                  break;
            }
            case 'A':  case 'B':  case 'C':  case 'D':
            case 'E':  case 'F':  case 'G':  case 'H':
            case 'I':  case 'J':  case 'K':  case 'L':
            case 'M':  case 'N':  case 'O':  case 'P':
            case 'Q':  case 'R':  case 'S':  case 'T':
            case 'U':  case 'V':  case 'W':  case 'X':
            case 'Y':  case 'Z':
            {
                  matchRange('A','Z');
                  break;
            }
            case '_':
            {
                  match('_');
                  break;
            }
            default:
            {
                  throw new NoViableAltForCharException((char)LA(1),
getFilename(), getLine(), getColumn());
            }
            }
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      protected final void mDIGIT(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = DIGIT;
            int _saveIndex;

            matchRange('0','9');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
```

```
	}

	public final void mWS(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = WS;
		int _saveIndex;

		{
		int _cnt5=0;
		_loop5:
		do {
			switch ( LA(1)) {
			case ' ':
			{
				match(' ');
				break;
			}
			case '\t':
			{
				match('\t');
				break;
			}
			default:
			{
				if ( _cnt5>=1 ) { break _loop5; } else {throw
new NoViableAltForCharException((char)LA(1), getFilename(), getLine(),
getColumn());}
			}
			}
			_cnt5++;
		} while (true);
		}
		if ( inputState.guessing==0 ) {
			_ttype = Token.SKIP;
		}
		if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
			_token = makeToken(_ttype);
			_token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
		}
		_returnToken = _token;
	}

	public final void mNL(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
		int _ttype; Token _token=null; int _begin=text.length();
		_ttype = NL;
		int _saveIndex;

		{
		boolean synPredMatched9 = false;
		if (((LA(1)=='\r') && (LA(2)=='\n') && (true) && (true) &&
(true))) {
			int _m9 = mark();
			synPredMatched9 = true;
			inputState.guessing++;
```

```
                  try {
                        {
                        match('\r');
                        match('\n');
                        }
                  }
                  catch (RecognitionException pe) {
                        synPredMatched9 = false;
                  }
                  rewind(_m9);
                  inputState.guessing--;
            }
            if ( synPredMatched9 ) {
                  match('\r');
                  match('\n');
            }
            else if ((LA(1)=='\n')) {
                  match('\n');
            }
            else if ((LA(1)=='\r') && (true) && (true) && (true) &&
(true)) {
                  match('\r');
            }
            else {
                  throw new NoViableAltForCharException((char)LA(1),
getFilename(), getLine(), getColumn());
            }

            }
            if ( inputState.guessing==0 ) {
                  _ttype = Token.SKIP; newline();
            }
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mCOMMENT(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = COMMENT;
            int _saveIndex;

            {
            if ((LA(1)=='/') && (LA(2)=='*')) {
                  match("/*");
                  {
                  _loop15:
                  do {
                        // nongreedy exit test
                        if ((LA(1)=='*') && (LA(2)=='/') && (true))
break _loop15;
```

```
                        if ((_tokenSet_3.member(LA(1))) && ((LA(2) >=
'\u0003' && LA(2) <= '\u00ff')) && ((LA(3) >= '\u0003' && LA(3) <=
'\u00ff'))) {
                              {
                              match(_tokenSet_3);
                              }
                        }
                        else if ((LA(1)=='\n'||LA(1)=='\r')) {
                              {
                              mNL(false);
                              }
                        }
                        else {
                              break _loop15;
                        }

                  } while (true);
                  }
                  match("*/");
            }
            else if ((LA(1)=='/') && (LA(2)=='/')) {
                  match("//");
                  {
                  _loop18:
                  do {
                        if ((_tokenSet_3.member(LA(1)))) {
                              {
                              match(_tokenSet_3);
                              }
                        }
                        else {
                              break _loop18;
                        }

                  } while (true);
                  }
                  {
                  mNL(false);
                  }
            }
            else {
                  throw new NoViableAltForCharException((char)LA(1),
getFilename(), getLine(), getColumn());
            }

            }
            if ( inputState.guessing==0 ) {
                  _ttype = Token.SKIP;
            }
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }
```

```
      public final void mROI(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ROI;
            int _saveIndex;

            match(":ReturnOnInvestment");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mPROFIT(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = PROFIT;
            int _saveIndex;

            match(":Profit");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mLOB(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = LOB;
            int _saveIndex;

            match(":LineOfBusiness");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mMTR(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = MTR;
            int _saveIndex;

            match(":MeanTimeToRecovery");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
```

```
            _returnToken = _token;
      }

      public final void mRT(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = RT;
            int _saveIndex;

            match(":ResponseTime");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mREVENUE(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = REVENUE;
            int _saveIndex;

            match(":Revenue");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mTXCOST(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = TXCOST;
            int _saveIndex;

            match(":TxCost");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mPENALTY(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = PENALTY;
            int _saveIndex;

            match(":Penalty");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
```

```
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mFSCOPE(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = FSCOPE;
            int _saveIndex;

            match("For Scope");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mITCOSTS(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ITCOSTS;
            int _saveIndex;

            match(":ITCosts");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mROUTERS(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ROUTERS;
            int _saveIndex;

            match(":Routers");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mSTORAGE(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = STORAGE;
            int _saveIndex;
```

```
            match(":Storage");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mSERVERS(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = SERVERS;
            int _saveIndex;

            match(":Servers");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mBANDW(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = BANDW;
            int _saveIndex;

            match(":Bandwidth");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mPOWER(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = POWER;
            int _saveIndex;

            match(":Power");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mPERS(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
```

```
            _ttype = PERS;
            int _saveIndex;

            match(":Personnel");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mACTIONS(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ACTIONS;
            int _saveIndex;

            match(":Actions");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mFCAST(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = FCAST;
            int _saveIndex;

            match(":Forecast");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mOPTIMIZE(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = OPTIMIZE;
            int _saveIndex;

            match(":Optimize");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }
```

```
      public final void mPROVISION(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = PROVISION;
            int _saveIndex;

            match(":Provision");
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mLPAREN(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = LPAREN;
            int _saveIndex;

            match('(');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mRPAREN(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = RPAREN;
            int _saveIndex;

            match(')');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {

                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mSEMI(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = SEMI;
            int _saveIndex;

            match(';');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
```

```
            }
            _returnToken = _token;
      }

      public final void mLBRACE(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = LBRACE;
            int _saveIndex;

            match('{');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mRBRACE(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = RBRACE;
            int _saveIndex;


            match('}');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mASGN(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ASGN;
            int _saveIndex;

            match('=');
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mID(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = ID;
            int _saveIndex;

            mALPHA(false);
```

```
            {
            _loop48:
            do {
                  switch ( LA(1)) {
                  case 'A':  case 'B':  case 'C':  case 'D':
                  case 'E':  case 'F':  case 'G':  case 'H':
                  case 'I':  case 'J':  case 'K':  case 'L':
                  case 'M':  case 'N':  case 'O':  case 'P':
                  case 'Q':  case 'R':  case 'S':  case 'T':
                  case 'U':  case 'V':  case 'W':  case 'X':
                  case 'Y':  case 'Z':  case '_':  case 'a':
                  case 'b':  case 'c':  case 'd':  case 'e':
                  case 'f':  case 'g':  case 'h':  case 'i':
                  case 'j':  case 'k':  case 'l':  case 'm':
                  case 'n':  case 'o':  case 'p':  case 'q':
                  case 'r':  case 's':  case 't':  case 'u':
                  case 'v':  case 'w':  case 'x':  case 'y':
                  case 'z':
                  {
                        mALPHA(false);
                        break;
                  }
                  case '0':  case '1':  case '2':  case '3':
                  case '4':  case '5':  case '6':  case '7':
                  case '8':  case '9':
                  {
                        mDIGIT(false);
                        break;
                  }
                  default:
                  {
                        break _loop48;
                  }
                  }
            } while (true);
            }
            _ttype = testLiteralsTable(_ttype);
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mPROVACT(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();

            _ttype = PROVACT;
            int _saveIndex;

            {
            if ((LA(1)==':') && (LA(2)=='C')) {
                  match(":CurrentRevenue");
            }
            else if ((LA(1)==':') && (LA(2)=='P')) {
```

```
                  match(":ProjectedRevenue");
            }
            else if ((LA(1)==':') && (LA(2)=='G')) {
                  match(":Growth");
            }
            else if ((LA(1)==':') && (LA(2)=='n')) {
                  match(":none");
            }
            else {
                  throw new NoViableAltForCharException((char)LA(1),
getFilename(), getLine(), getColumn());
            }

            }
            _ttype = testLiteralsTable(_ttype);
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mNUMBER(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = NUMBER;
            int _saveIndex;

            {
            int _cnt53=0;
            _loop53:
            do {
                  if (((LA(1) >= '0' && LA(1) <= '9'))) {
                        mDIGIT(false);
                  }
                  else {
                        if ( _cnt53>=1 ) { break _loop53; } else {throw
new NoViableAltForCharException((char)LA(1), getFilename(), getLine(),
getColumn());}
                  }

                  _cnt53++;
            } while (true);
            }
            {
            if ((LA(1)=='.')) {

                  match('.');
                  {
                  _loop56:
                  do {
                        if (((LA(1) >= '0' && LA(1) <= '9'))) {
                              mDIGIT(false);
                        }
                        else {
                              break _loop56;
```

```
                        }

                  } while (true);
                  }
            }
            else {
            }

            }
            {
            if ((LA(1)=='E'||LA(1)=='e')) {
                  {
                  switch ( LA(1)) {
                  case 'E':
                  {
                        match('E');
                        break;
                  }
                  case 'e':
                  {
                        match('e');
                        break;
                  }
                  default:
                  {
                        throw new
NoViableAltForCharException((char)LA(1), getFilename(), getLine(),
getColumn());
                  }
                  }
                  }
                  {
                  switch ( LA(1)) {
                  case '+':
                  {
                        match('+');
                        break;
                  }
                  case '-':
                  {
                        match('-');
                        break;
                  }
                  case '0':  case '1':  case '2':  case '3':
                  case '4':  case '5':  case '6':  case '7':
                  case '8':  case '9':
                  {
                        break;
                  }
                  default:
                  {
                        throw new
NoViableAltForCharException((char)LA(1), getFilename(), getLine(),
getColumn());
                  }
                  }
                  }
```

```
                  {
                  int _cnt61=0;
                  _loop61:
                  do {
                        if (((LA(1) >= '0' && LA(1) <= '9'))) {
                              mDIGIT(false);
                        }
                        else {
                              if ( _cnt61>=1 ) { break _loop61; } else
{throw new NoViableAltForCharException((char)LA(1), getFilename(),
getLine(), getColumn());}
                        }

                        _cnt61++;
                  } while (true);
                  }
            }
            else {
            }

            }
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }

      public final void mSTRING(boolean _createToken) throws
RecognitionException, CharStreamException, TokenStreamException {
            int _ttype; Token _token=null; int _begin=text.length();
            _ttype = STRING;
            int _saveIndex;

            _saveIndex=text.length();
            match('"');
            text.setLength(_saveIndex);
            {
            _loop66:
            do {
                  if ((LA(1)=='"') && (LA(2)=='"')) {
                        {
                        _saveIndex=text.length();
                        match('"');
                        text.setLength(_saveIndex);
                        match('"');
                        }
                  }
                  else if ((_tokenSet_4.member(LA(1)))) {
                        {
                        match(_tokenSet_4);
                        }
                  }
                  else {
                        break _loop66;
                  }
```

```
            } while (true);
            }
            _saveIndex=text.length();
            match('"');
            text.setLength(_saveIndex);
            if ( _createToken && _token==null && _ttype!=Token.SKIP ) {
                  _token = makeToken(_ttype);
                  _token.setText(new String(text.getBuffer(), _begin,
text.length()-_begin));
            }
            _returnToken = _token;
      }


      private static final long[] mk_tokenSet_0() {
            long[] data = { 0L, 70368744243336L, 0L, 0L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_0 = new
BitSet(mk_tokenSet_0());
      private static final long[] mk_tokenSet_1() {
            long[] data = { 0L, 37159232411271168L, 0L, 0L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_1 = new
BitSet(mk_tokenSet_1());
      private static final long[] mk_tokenSet_2() {
            long[] data = { 0L, 5764607459951902 70L, 0L, 0L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_2 = new
BitSet(mk_tokenSet_2());
      private static final long[] mk_tokenSet_3() {
            long[] data = new long[8];
            data[0]=-9224L;
            for (int i = 1; i<=3; i++) { data[i]=-1L; }
            return data;
      }
      public static final BitSet _tokenSet_3 = new
BitSet(mk_tokenSet_3());
      private static final long[] mk_tokenSet_4() {
            long[] data = new long[8];
            data[0]=-17179870216L;
            for (int i = 1; i<=3; i++) { data[i]=-1L; }
            return data;
      }
      public static final BitSet _tokenSet_4 = new
BitSet(mk_tokenSet_4());

      }
// $ANTLR 2.7.2: "sal.g" -> "SALAntlrParser.java"$

import antlr.TokenBuffer;
import antlr.TokenStreamException;
import antlr.TokenStreamIOException;
import antlr.ANTLRException;
```

```
import antlr.LLkParser;
import antlr.Token;
import antlr.TokenStream;
import antlr.RecognitionException;
import antlr.NoViableAltException;
import antlr.MismatchedTokenException;
import antlr.SemanticException;
import antlr.ParserSharedInputState;
import antlr.collections.impl.BitSet;
import antlr.collections.AST;
import java.util.Hashtable;
import antlr.ASTFactory;
import antlr.ASTPair;
import antlr.collections.impl.ASTArray;

/*****************************************************************/
public class SALAntlrParser extends antlr.LLkParser       implements
SALAntlrTokenTypes
 {

    int nr_error = 0;
    public void reportError( String s ) {
        super.reportError( s );
        nr_error++;
    }
    public void reportError( RecognitionException e ) {
        super.reportError( e );
        nr_error++;
    }

protected SALAntlrParser(TokenBuffer tokenBuf, int k) {
  super(tokenBuf,k);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public SALAntlrParser(TokenBuffer tokenBuf) {
  this(tokenBuf,5);
}

protected SALAntlrParser(TokenStream lexer, int k) {
  super(lexer,k);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
}

public SALAntlrParser(TokenStream lexer) {
  this(lexer,5);
}

public SALAntlrParser(ParserSharedInputState state) {
  super(state,5);
  tokenNames = _tokenNames;
  buildTokenTypeASTClassMap();
  astFactory = new ASTFactory(getTokenTypeToASTClassMap());
```

```
}

      public final void roi_def() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST roi_def_AST = null;

            try {      // for error handling
                  AST tmp1_AST = null;
                  tmp1_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp1_AST);
                  match(ROI);
                  match(SEMI);
                  match(LBRACE);
                  revenue_def();
                  astFactory.addASTChild(currentAST, returnAST);
                  itcosts_def();
                  astFactory.addASTChild(currentAST, returnAST);
                  actions_def();
                  astFactory.addASTChild(currentAST, returnAST);
                  match(RBRACE);
                  roi_def_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_0);
            }
            returnAST = roi_def_AST;
      }

      public final void revenue_def() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST revenue_def_AST = null;

            try {      // for error handling
                  AST tmp5_AST = null;
                  tmp5_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp5_AST);
                  match(REVENUE);
                  match(SEMI);
                  match(LBRACE);
                  {
                  _loop70:
                  do {
                        if ((LA(1)==LOB)) {
                              lob_def();
                              astFactory.addASTChild(currentAST,
returnAST);
                        }
                        else {
                              break _loop70;
```

```
                        }

                  } while (true);
                  }
                  match(RBRACE);
                  revenue_def_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_1);
            }
            returnAST = revenue_def_AST;
      }

      public final void itcosts_def() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST itcosts_def_AST = null;

            try {      // for error handling
                  AST tmp9_AST = null;
                  tmp9_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp9_AST);
                  match(ITCOSTS);
                  match(SEMI);
                  match(LBRACE);
                  router_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  storage_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  servers_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  bandw_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  power_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  pers_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  match(RBRACE);
                  itcosts_def_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_2);
            }
            returnAST = itcosts_def_AST;
      }

      public final void actions_def() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
```

```
            AST actions_def_AST = null;

            try {      // for error handling
                  AST tmp13_AST = null;
                  tmp13_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp13_AST);
                  match(ACTIONS);
                  match(SEMI);
                  match(LBRACE);
                  forecast_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  optimize_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  provision_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  match(RBRACE);
                  actions_def_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_3);
            }
            returnAST = actions_def_AST;
      }

      public final void lob_def() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST lob_def_AST = null;

            try {      // for error handling
                  AST tmp17_AST = null;
                  tmp17_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp17_AST);
                  match(LOB);
                  AST tmp18_AST = null;
                  tmp18_AST = astFactory.create(LT(1));
                  astFactory.addASTChild(currentAST, tmp18_AST);
                  match(ID);
                  match(SEMI);
                  match(LBRACE);
                  mtr_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  response_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  profit_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  txcost_stmt();
                  astFactory.addASTChild(currentAST, returnAST);
                  {
                  switch ( LA(1)) {
                  case FSCOPE:
                  {
                        for_stmt();
```

```
                        astFactory.addASTChild(currentAST, returnAST);
                        break;
                  }
                  case PENALTY:
                  {
                        penalty_stmt();
                        astFactory.addASTChild(currentAST, returnAST);
                        break;
                  }
                  default:
                  {
                        throw new NoViableAltException(LT(1),
getFilename());
                  }
                  }
                  }
                  match(RBRACE);
                  lob_def_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_4);
            }
            returnAST = lob_def_AST;
      }

      public final void mtr_stmt() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST mtr_stmt_AST = null;

            try {      // for error handling
                  AST tmp22_AST = null;
                  tmp22_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp22_AST);
                  match(MTR);
                  match(ASGN);
                  AST tmp24_AST = null;
                  tmp24_AST = astFactory.create(LT(1));
                  astFactory.addASTChild(currentAST, tmp24_AST);
                  match(NUMBER);
                  time();
                  match(SEMI);
                  mtr_stmt_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_5);
            }
            returnAST = mtr_stmt_AST;
      }
```

```
	public final void response_stmt() throws RecognitionException,
TokenStreamException {


		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST response_stmt_AST = null;

		try {	// for error handling
			AST tmp26_AST = null;
			tmp26_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp26_AST);

			match(RT);
			match(ASGN);
			AST tmp28_AST = null;
			tmp28_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp28_AST);
			match(NUMBER);
			time();
			match(SEMI);
			response_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_6);
		}
		returnAST = response_stmt_AST;
	}

	public final void profit_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST profit_stmt_AST = null;

		try {	// for error handling
			AST tmp30_AST = null;
			tmp30_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp30_AST);
			match(PROFIT);
			match(ASGN);
			AST tmp32_AST = null;
			tmp32_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp32_AST);
			match(NUMBER);
			currency();
			match(SEMI);
			profit_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_7);
		}
```

```
            returnAST = profit_stmt_AST;
      }

      public final void txcost_stmt() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST txcost_stmt_AST = null;

            try {      // for error handling
                  AST tmp34_AST = null;
                  tmp34_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp34_AST);
                  match(TXCOST);
                  match(ASGN);
                  AST tmp36_AST = null;
                  tmp36_AST = astFactory.create(LT(1));
                  astFactory.addASTChild(currentAST, tmp36_AST);
                  match(NUMBER);
                  currency();
                  match(SEMI);
                  txcost_stmt_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_8);
            }
            returnAST = txcost_stmt_AST;
      }

      public final void for_stmt() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST for_stmt_AST = null;

            try {      // for error handling
                  AST tmp38_AST = null;
                  tmp38_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp38_AST);
                  match(FSCOPE);
                  match(ASGN);
                  AST tmp40_AST = null;
                  tmp40_AST = astFactory.create(LT(1));
                  astFactory.addASTChild(currentAST, tmp40_AST);
                  match(NUMBER);
                  match(LITERAL_to);
                  AST tmp42_AST = null;
                  tmp42_AST = astFactory.create(LT(1));
                  astFactory.addASTChild(currentAST, tmp42_AST);
                  match(NUMBER);
                  match(SEMI);
                  match(LPAREN);
                  penalty_stmt();
```

```
                astFactory.addASTChild(currentAST, returnAST);
                match(RPAREN);
                for_stmt_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_3);
            }
            returnAST = for_stmt_AST;
        }

        public final void penalty_stmt() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST penalty_stmt_AST = null;

            try {      // for error handling
                AST tmp46_AST = null;
                tmp46_AST = astFactory.create(LT(1));
                astFactory.makeASTRoot(currentAST, tmp46_AST);
                match(PENALTY);
                match(ASGN);
                AST tmp48_AST = null;
                tmp48_AST = astFactory.create(LT(1));
                astFactory.addASTChild(currentAST, tmp48_AST);
                match(NUMBER);
                currency();
                match(SEMI);
                penalty_stmt_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                reportError(ex);
                consume();
                consumeUntil(_tokenSet_9);
            }
            returnAST = penalty_stmt_AST;
        }

        public final void time() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST time_AST = null;

            try {      // for error handling
                switch ( LA(1)) {
                case LITERAL_minutes:
                {
                    AST tmp50_AST = null;
                    tmp50_AST = astFactory.create(LT(1));
                    astFactory.addASTChild(currentAST, tmp50_AST);
                    match(LITERAL_minutes);
                    time_AST = (AST)currentAST.root;
```

```
                        break;
                  }
                  case LITERAL_minute:
                  {
                        AST tmp51_AST = null;
                        tmp51_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp51_AST);
                        match(LITERAL_minute);
                        time_AST = (AST)currentAST.root;
                        break;
                  }
                  case LITERAL_seconds:
                  {
                        AST tmp52_AST = null;
                        tmp52_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp52_AST);
                        match(LITERAL_seconds);
                        time_AST = (AST)currentAST.root;
                        break;
                  }
                  case LITERAL_second:
                  {
                        AST tmp53_AST = null;
                        tmp53_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp53_AST);
                        match(LITERAL_second);
                        time_AST = (AST)currentAST.root;
                        break;
                  }
                  default:
                  {
                        throw new NoViableAltException(LT(1),
getFilename());
                  }
                  }
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_10);
            }
            returnAST = time_AST;
      }

      public final void currency() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST currency_AST = null;

            try {      // for error handling
                  switch ( LA(1)) {
                  case LITERAL_dollar:
                  {
                        AST tmp54_AST = null;
                        tmp54_AST = astFactory.create(LT(1));
```

```
                        astFactory.addASTChild(currentAST, tmp54_AST);
                        match(LITERAL_dollar);
                        currency_AST = (AST)currentAST.root;
                        break;
                  }
                  case LITERAL_dollars:
                  {
                        AST tmp55_AST = null;
                        tmp55_AST = astFactory.create(LT(1));
                        astFactory.addASTChild(currentAST, tmp55_AST);
                        match(LITERAL_dollars);
                        currency_AST = (AST)currentAST.root;
                        break;
                  }
                  default:
                  {
                        throw new NoViableAltException(LT(1),
getFilename());
                  }
                  }
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_10);
            }
            returnAST = currency_AST;
      }

      public final void router_stmt() throws RecognitionException,
TokenStreamException {

            returnAST = null;
            ASTPair currentAST = new ASTPair();
            AST router_stmt_AST = null;

            try {      // for error handling
                  AST tmp56_AST = null;
                  tmp56_AST = astFactory.create(LT(1));
                  astFactory.makeASTRoot(currentAST, tmp56_AST);
                  match(ROUTERS);
                  match(ASGN);
                  AST tmp58_AST = null;
                  tmp58_AST = astFactory.create(LT(1));
                  astFactory.addASTChild(currentAST, tmp58_AST);
                  match(ID);
                  match(SEMI);
                  router_stmt_AST = (AST)currentAST.root;
            }
            catch (RecognitionException ex) {
                  reportError(ex);
                  consume();
                  consumeUntil(_tokenSet_11);
            }
            returnAST = router_stmt_AST;
      }
```

```
	public final void storage_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST storage_stmt_AST = null;


		try {	// for error handling
			AST tmp60_AST = null;
			tmp60_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp60_AST);
			match(STORAGE);
			match(ASGN);
			AST tmp62_AST = null;
			tmp62_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp62_AST);
			match(ID);
			match(SEMI);
			storage_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_12);
		}
		returnAST = storage_stmt_AST;
	}

	public final void servers_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST servers_stmt_AST = null;

		try {	// for error handling
			AST tmp64_AST = null;
			tmp64_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp64_AST);
			match(SERVERS);
			match(ASGN);
			AST tmp66_AST = null;
			tmp66_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp66_AST);
			match(ID);
			match(SEMI);
			servers_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_13);
		}
		returnAST = servers_stmt_AST;
	}
```

```
	public final void bandw_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST bandw_stmt_AST = null;

		try {	// for error handling
			AST tmp68_AST = null;
			tmp68_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp68_AST);
			match(BANDW);
			match(ASGN);
			AST tmp70_AST = null;
			tmp70_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp70_AST);
			match(ID);
			match(SEMI);
			bandw_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_14);
		}
		returnAST = bandw_stmt_AST;
	}

	public final void power_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST power_stmt_AST = null;

		try {	// for error handling
			AST tmp72_AST = null;
			tmp72_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp72_AST);
			match(POWER);
			match(ASGN);
			AST tmp74_AST = null;
			tmp74_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp74_AST);
			match(ID);
			match(SEMI);
			power_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_15);
		}
		returnAST = power_stmt_AST;
	}
```

```
	public final void pers_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST pers_stmt_AST = null;

		try {	// for error handling
			AST tmp76_AST = null;
			tmp76_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp76_AST);
			match(PERS);
			match(ASGN);
			AST tmp78_AST = null;
			tmp78_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp78_AST);
			match(NUMBER);
			match(SEMI);
			pers_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_3);
		}
		returnAST = pers_stmt_AST;
	}

	public final void forecast_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST forecast_stmt_AST = null;

		try {	// for error handling
			AST tmp80_AST = null;
			tmp80_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp80_AST);
			match(FCAST);
			match(ASGN);
			AST tmp82_AST = null;

			tmp82_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp82_AST);
			match(ID);
			match(SEMI);
			forecast_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_16);
		}
		returnAST = forecast_stmt_AST;
	}
```

```
	public final void optimize_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST optimize_stmt_AST = null;


		try {	// for error handling
			AST tmp84_AST = null;
			tmp84_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp84_AST);
			match(OPTIMIZE);
			match(ASGN);
			AST tmp86_AST = null;
			tmp86_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp86_AST);
			match(ID);
			match(SEMI);
			optimize_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_17);
		}
		returnAST = optimize_stmt_AST;
	}

	public final void provision_stmt() throws RecognitionException,
TokenStreamException {

		returnAST = null;
		ASTPair currentAST = new ASTPair();
		AST provision_stmt_AST = null;

		try {	// for error handling
			AST tmp88_AST = null;
			tmp88_AST = astFactory.create(LT(1));
			astFactory.makeASTRoot(currentAST, tmp88_AST);
			match(PROVISION);
			match(LPAREN);
			AST tmp90_AST = null;
			tmp90_AST = astFactory.create(LT(1));
			astFactory.addASTChild(currentAST, tmp90_AST);
			match(PROVACT);
			match(RPAREN);
			match(SEMI);
			provision_stmt_AST = (AST)currentAST.root;
		}
		catch (RecognitionException ex) {
			reportError(ex);
			consume();
			consumeUntil(_tokenSet_3);
		}
		returnAST = provision_stmt_AST;
	}
```

```
public static final String[] _tokenNames = {
      "<0>",
      "EOF",
      "<2>",
      "NULL_TREE_LOOKAHEAD",
      "ALPHA",
      "DIGIT",
      "WS",
      "NL",
      "COMMENT",
      "ROI",
      "PROFIT",
      "LOB",
      "MTR",
      "RT",
      "REVENUE",
      "TXCOST",
      "PENALTY",
      "FSCOPE",
      "ITCOSTS",
      "ROUTERS",
      "STORAGE",
      "SERVERS",
      "BANDW",
      "POWER",
      "PERS",
      "ACTIONS",
      "FCAST",
      "OPTIMIZE",
      "PROVISION",
      "LPAREN",
      "RPAREN",
      "SEMI",
      "LBRACE",
      "RBRACE",
      "ASGN",
      "ID",
      "PROVACT",
      "NUMBER",
      "STRING",
      "\"to\"",
      "\"minutes\"",
      "\"minute\"",
      "\"seconds\"",
      "\"second\"",
      "\"dollar\"",
      "\"dollars\""
};

protected void buildTokenTypeASTClassMap() {
      tokenTypeToASTClassMap=null;
};

private static final long[] mk_tokenSet_0() {
      long[] data = { 2L, 0L};
```

```
            return data;
      }
      public static final BitSet _tokenSet_0 = new
BitSet(mk_tokenSet_0());
      private static final long[] mk_tokenSet_1() {
            long[] data = { 262144L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_1 = new
BitSet(mk_tokenSet_1());
      private static final long[] mk_tokenSet_2() {
            long[] data = { 33554432L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_2 = new
BitSet(mk_tokenSet_2());
      private static final long[] mk_tokenSet_3() {
            long[] data = { 8589934592L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_3 = new
BitSet(mk_tokenSet_3());
      private static final long[] mk_tokenSet_4() {
            long[] data = { 8589936640L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_4 = new
BitSet(mk_tokenSet_4());
      private static final long[] mk_tokenSet_5() {
            long[] data = { 8192L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_5 = new
BitSet(mk_tokenSet_5());
      private static final long[] mk_tokenSet_6() {
            long[] data = { 1024L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_6 = new
BitSet(mk_tokenSet_6());
      private static final long[] mk_tokenSet_7() {
            long[] data = { 32768L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_7 = new
BitSet(mk_tokenSet_7());
      private static final long[] mk_tokenSet_8() {
            long[] data = { 196608L, 0L};
            return data;
      }
      public static final BitSet _tokenSet_8 = new
BitSet(mk_tokenSet_8());
      private static final long[] mk_tokenSet_9() {
            long[] data = { 9663676416L, 0L};
            return data;
      }
```

```
	public static final BitSet _tokenSet_9 = new
BitSet(mk_tokenSet_9());
	private static final long[] mk_tokenSet_10() {
		long[] data = { 2147483648L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_10 = new
BitSet(mk_tokenSet_10());
	private static final long[] mk_tokenSet_11() {
		long[] data = { 1048576L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_11 = new
BitSet(mk_tokenSet_11());
	private static final long[] mk_tokenSet_12() {
		long[] data = { 2097152L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_12 = new
BitSet(mk_tokenSet_12());
	private static final long[] mk_tokenSet_13() {
		long[] data = { 4194304L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_13 = new
BitSet(mk_tokenSet_13());
	private static final long[] mk_tokenSet_14() {
		long[] data = { 8388608L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_14 = new
BitSet(mk_tokenSet_14());
	private static final long[] mk_tokenSet_15() {
		long[] data = { 16777216L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_15 = new
BitSet(mk_tokenSet_15());
	private static final long[] mk_tokenSet_16() {
		long[] data = { 134217728L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_16 = new
BitSet(mk_tokenSet_16());
	private static final long[] mk_tokenSet_17() {
		long[] data = { 268435456L, 0L};
		return data;
	}
	public static final BitSet _tokenSet_17 = new
BitSet(mk_tokenSet_17());

	}
// $ANTLR 2.7.2: "sal.g" -> "SALAntlrLexer.java"$

public interface SALAntlrTokenTypes {
	int EOF = 1;
	int NULL_TREE_LOOKAHEAD = 3;
```

```
        int ALPHA = 4;
        int DIGIT = 5;
        int WS = 6;
        int NL = 7;
        int COMMENT = 8;
        int ROI = 9;
        int PROFIT = 10;
        int LOB = 11;
        int MTR = 12;
        int RT = 13;
        int REVENUE = 14;
        int TXCOST = 15;
        int PENALTY = 16;
        int FSCOPE = 17;
        int ITCOSTS = 18;
        int ROUTERS = 19;
        int STORAGE = 20;
        int SERVERS = 21;
        int BANDW = 22;
        int POWER = 23;
        int PERS = 24;
        int ACTIONS = 25;
        int FCAST = 26;
        int OPTIMIZE = 27;
        int PROVISION = 28;
        int LPAREN = 29;
        int RPAREN = 30;
        int SEMI = 31;
        int LBRACE = 32;
        int RBRACE = 33;
        int ASGN = 34;
        int ID = 35;
        int PROVACT = 36;
        int NUMBER = 37;
        int STRING = 38;
        int LITERAL_to = 39;
        int LITERAL_minutes = 40;
        int LITERAL_minute = 41;
        int LITERAL_seconds = 42;
        int LITERAL_second = 43;
        int LITERAL_dollar = 44;
        int LITERAL_dollars = 45;
}
// $ANTLR 2.7.2: "sal.g" -> "SALAntlrWalker.java"$

import antlr.TreeParser;
import antlr.Token;
import antlr.collections.AST;
import antlr.RecognitionException;
import antlr.ANTLRException;
import antlr.NoViableAltException;
import antlr.MismatchedTokenException;
import antlr.SemanticException;
import antlr.collections.impl.BitSet;
import antlr.ASTPair;
import antlr.collections.impl.ASTArray;
```

```
/*************************************************************************
 * walker.g : the AST walker.
 *
 * @author Donna Eng Dillenberger
 */
public class SALAntlrWalker extends antlr.TreeParser       implements
SALAntlrWalkerTokenTypes
 {

   public static LineOfBusiness [] SpecificLOB;    //Save parsed values
in Line of Business data structures
   public static int LOBInstance;
   public static ITCost IT;                         //Save parsed values
in IT components data structures
   public static int ITInstance;
   public static String [] ActionsToCall;          //Save parsed values
in Actions data structures
   public static int ActInstance;
   public void initialize()                         //Initialize above
data structures to prepare for
   {                                                //parsing
      LOBInstance = -1;
      SpecificLOB = new LineOfBusiness[3];
      for (int i=0; i<3; i++)
      {  SpecificLOB[i] = new LineOfBusiness();  }

      IT = new ITCost();
      ITInstance = 0;

      ActionsToCall = new String[3];
      ActInstance = 0;



   } // end initialize
public SALAntlrWalker() {
      tokenNames = _tokenNames;
}

      public final String  roi_def(AST _t) throws RecognitionException
{
            String s;

            AST roi_def_AST_in = (AST)_t;
            AST ro = null;
            AST re = null;
            AST lo = null;
            AST l = null;
            AST m = null;
            AST mn = null;
            AST r = null;
            AST rn = null;
            AST pr = null;
            AST pn = null;
            AST tx = null;
            AST tn = null;
```

```
            AST pe = null;
            AST pen = null;
            AST fo = null;
            AST f1n = null;
            AST f2n = null;
            AST it = null;
            AST rou = null;
            AST rid = null;
            AST sto = null;
            AST stid = null;
            AST ser = null;
            AST serid = null;
            AST ba = null;
            AST bid = null;
            AST pow = null;
            AST pid = null;
            AST per = null;
            AST pers = null;
            AST ac = null;
            AST fcastid = null;
            AST fid = null;
            AST opt = null;
            AST oid = null;
            AST prov = null;
            AST prid = null;

            s = null;
            String
mc,rc,pc,tc,l1,l2,l3,pec,si,sa,sc,stc,bc,poc,perc,sac,fc,oc = null;


            try {      // for error handling
                  if (_t==null) _t=ASTNULL;
                  switch ( _t.getType()) {
                  case ROI:
                  {
                        AST __t93 = _t;
                        ro = _t==ASTNULL ? null :(AST)_t;
                        match(_t,ROI);
                        _t = _t.getFirstChild();
                        s=roi_def(_t);
                        _t = _retTree;
                        si=roi_def(_t);
                        _t = _retTree;
                        sac=roi_def(_t);
                        _t = _retTree;
                        _t = __t93;
                        _t = _t.getNextSibling();
                        // Parsed values have all been collected, call
the Interpreter
                        // to run the user's program
                        SALInterpreter.Interpretermain (SpecificLOB,
IT, ActionsToCall);

                        break;
                  }
                  case REVENUE:
```

```
                        {
                              AST __t94 = _t;
                              re = _t==ASTNULL ? null :(AST)_t;
                              match(_t,REVENUE);
                              _t = _t.getFirstChild();
                              l1=roi_def(_t);
                              _t = _retTree;
                              l2=roi_def(_t);
                              _t = _retTree;
                              l3=roi_def(_t);
                              _t = _retTree;
                              _t = __t94;
                              _t = _t.getNextSibling();
                              break;
                        }
                        case LOB:
                        {
                              AST __t95 = _t;
                              lo = _t==ASTNULL ? null :(AST)_t;
                              match(_t,LOB);
                              _t = _t.getFirstChild();
                              l = (AST)_t;
                              match(_t,ID);
                              _t = _t.getNextSibling();
                              mc=roi_def(_t);
                              _t = _retTree;
                              rc=roi_def(_t);
                              _t = _retTree;
                              pc=roi_def(_t);
                              _t = _retTree;
                              tc=roi_def(_t);
                              _t = _retTree;
                              pec=roi_def(_t);

                              _t = _retTree;
                              _t = __t95;
                              _t = _t.getNextSibling();
                              SpecificLOB[LOBInstance].LOBName = l.getText();


                              break;
                        }
                        case MTR:
                        {
                              AST __t96 = _t;
                              m = _t==ASTNULL ? null :(AST)_t;
                              match(_t,MTR);
                              _t = _t.getFirstChild();
                              mn = (AST)_t;
                              match(_t,NUMBER);
                              _t = _t.getNextSibling();
                              _t = __t96;
                              _t = _t.getNextSibling();
                              // Save Mean Time To Recovery goals
                              LOBInstance = LOBInstance + 1;
                              SpecificLOB[LOBInstance].MeanTimeToRecoveryGoal
=
```

```
      (int)(Integer.parseInt(mn.getText()));

      break;
}
case RT:
{
      AST __t97 = _t;
      r = _t==ASTNULL ? null :(AST)_t;
      match(_t,RT);
      _t = _t.getFirstChild();
      rn = (AST)_t;
      match(_t,NUMBER);
      _t = _t.getNextSibling();
      _t = __t97;
      _t = _t.getNextSibling();
      SpecificLOB[LOBInstance].ResponseTimeGoal =
      (int)(Integer.parseInt(rn.getText()));


      break;
}
case PROFIT:
{
      AST __t98 = _t;
      pr = _t==ASTNULL ? null :(AST)_t;
      match(_t,PROFIT);
      _t = _t.getFirstChild();
      pn = (AST)_t;
      match(_t,NUMBER);
      _t = _t.getNextSibling();
      _t = __t98;
      _t = _t.getNextSibling();
      SpecificLOB[LOBInstance].Profit =
      (int)(Integer.parseInt(pn.getText()));


      break;
}
case TXCOST:
{
      AST __t99 = _t;
      tx = _t==ASTNULL ? null :(AST)_t;
      match(_t,TXCOST);
      _t = _t.getFirstChild();
      tn = (AST)_t;
      match(_t,NUMBER);
      _t = _t.getNextSibling();
      _t = __t99;
      _t = _t.getNextSibling();

      SpecificLOB[LOBInstance].TxCost =
      (int)(Integer.parseInt(tn.getText()));

      break;
}
case PENALTY:
{
```

```
        AST __t100 = _t;
        pe = _t==ASTNULL ? null :(AST)_t;
        match(_t,PENALTY);
        _t = _t.getFirstChild();
        pen = (AST)_t;
        match(_t,NUMBER);
        _t = _t.getNextSibling();
        _t = __t100;
        _t = _t.getNextSibling();

        SpecificLOB[LOBInstance].Penalty =
        (int)(Integer.parseInt(pen.getText()));

        break;
      }
      case FSCOPE:
      {
        AST __t101 = _t;
        fo = _t==ASTNULL ? null :(AST)_t;
        match(_t,FSCOPE);
        _t = _t.getFirstChild();
        f1n = (AST)_t;
        match(_t,NUMBER);
        _t = _t.getNextSibling();
        f2n = (AST)_t;
        match(_t,NUMBER);
        _t = _t.getNextSibling();
        s=roi_def(_t);
        _t = _retTree;
        _t = __t101;
        _t = _t.getNextSibling();
        // Save time scope for penalty
        SpecificLOB[LOBInstance].PenaltyBegin =
        (int)(Integer.parseInt(f1n.getText()));

        SpecificLOB[LOBInstance].PenaltyEnd =
        (int)(Integer.parseInt(f2n.getText()));

        break;
      }
      case ITCOSTS:
      {
        AST __t102 = _t;
        it = _t==ASTNULL ? null :(AST)_t;
        match(_t,ITCOSTS);
        _t = _t.getFirstChild();
        rc=roi_def(_t);
        _t = _retTree;
        sc=roi_def(_t);
        _t = _retTree;
        stc=roi_def(_t);
        _t = _retTree;
        bc=roi_def(_t);
        _t = _retTree;
        poc=roi_def(_t);
        _t = _retTree;
        perc=roi_def(_t);
```

```
        _t = _retTree;
        _t = __t102;
        _t = _t.getNextSibling();
        break;
}
case ROUTERS:
{
        AST __t103 = _t;
        rou = _t==ASTNULL ? null :(AST)_t;
        match(_t,ROUTERS);
        _t = _t.getFirstChild();
        rid = (AST)_t;
        match(_t,ID);
        _t = _t.getNextSibling();
        _t = __t103;
        _t = _t.getNextSibling();
        // Charge for Routers
        IT.Expense[ITInstance] = rid.getText();
        ITInstance = ITInstance +1;


        break;
}
case STORAGE:
{
        AST __t104 = _t;
        sto = _t==ASTNULL ? null :(AST)_t;
        match(_t,STORAGE);
        _t = _t.getFirstChild();
        stid = (AST)_t;
        match(_t,ID);
        _t = _t.getNextSibling();
        _t = __t104;
        _t = _t.getNextSibling();
        // Charge for Storage
        IT.Expense[ITInstance] = stid.getText();
        ITInstance = ITInstance +1;

        break;
}
case SERVERS:
{
        AST __t105 = _t;
        ser = _t==ASTNULL ? null :(AST)_t;
        match(_t,SERVERS);
        _t = _t.getFirstChild();
        serid = (AST)_t;
        match(_t,ID);
        _t = _t.getNextSibling();
        _t = __t105;
        _t = _t.getNextSibling();
        // Charge for servers
        IT.Expense[ITInstance] = serid.getText();
        ITInstance = ITInstance +1;

        break;
}
```

```
case BANDW:
{
      AST __t106 = _t;
      ba = _t==ASTNULL ? null :(AST)_t;
      match(_t,BANDW);
      _t = _t.getFirstChild();
      bid = (AST)_t;
      match(_t,ID);
      _t = _t.getNextSibling();
      _t = __t106;
      _t = _t.getNextSibling();
      // Charge for Bandwidth
      IT.Expense[ITInstance] = bid.getText();
      ITInstance = ITInstance +1;



      break;
}
case POWER:
{
      AST __t107 = _t;
      pow = _t==ASTNULL ? null :(AST)_t;
      match(_t,POWER);
      _t = _t.getFirstChild();
      pid = (AST)_t;
      match(_t,ID);
      _t = _t.getNextSibling();
      _t = __t107;
      _t = _t.getNextSibling();
      //Charge for Power
      IT.Expense[ITInstance] = pid.getText();
      ITInstance = ITInstance +1;


      break;
}
case PERS:
{
      AST __t108 = _t;
      per = _t==ASTNULL ? null :(AST)_t;
      match(_t,PERS);
      _t = _t.getFirstChild();
      pers = (AST)_t;
      match(_t,NUMBER);
      _t = _t.getNextSibling();
      _t = __t108;
      _t = _t.getNextSibling();
      //Charge for Personnel
      IT.Personnel =
      (int)(Integer.parseInt(pers.getText()));

      break;
}
case ACTIONS:
{
      AST __t109 = _t;
```

```
		ac = _t==ASTNULL ? null :(AST)_t;
		match(_t,ACTIONS);
		_t = _t.getFirstChild();
		fc=roi_def(_t);
		_t = _retTree;
		oc=roi_def(_t);
		_t = _retTree;
		pc=roi_def(_t);
		_t = _retTree;
		_t = __t109;
		_t = _t.getNextSibling();
		break;
	}
	case FCAST:
	{
		AST __t110 = _t;
		fcastid = _t==ASTNULL ? null :(AST)_t;
		match(_t,FCAST);
		_t = _t.getFirstChild();
		fid = (AST)_t;
		match(_t,ID);
		_t = _t.getNextSibling();
		_t = __t110;
		_t = _t.getNextSibling();
		//Save Forecasting option
		ActionsToCall[ActInstance] = fid.getText();
		ActInstance = ActInstance +1;


		break;
	}
	case OPTIMIZE:
	{
		AST __t111 = _t;
		opt = _t==ASTNULL ? null :(AST)_t;
		match(_t,OPTIMIZE);
		_t = _t.getFirstChild();
		oid = (AST)_t;
		match(_t,ID);
		_t = _t.getNextSibling();
		_t = __t111;
		_t = _t.getNextSibling();
		//Save Optimize option
		ActionsToCall[ActInstance] = oid.getText();
		ActInstance = ActInstance +1;

		break;
	}
	case PROVISION:
	{
		AST __t112 = _t;
		prov = _t==ASTNULL ? null :(AST)_t;
		match(_t,PROVISION);
		_t = _t.getFirstChild();
		prid = (AST)_t;
		match(_t,PROVACT);
		_t = _t.getNextSibling();
```

```
                    _t = __t112;
                    _t = _t.getNextSibling();
                    //Save Provisioning option
                    ActionsToCall[ActInstance] = prid.getText();


                    break;
              }
              default:
              {
                    throw new NoViableAltException(_t);
              }
              }
        }
        catch (RecognitionException ex) {
              reportError(ex);
              if (_t!=null) {_t = _t.getNextSibling();}
        }
        _retTree = _t;
        return s;
}


public static final String[] _tokenNames = {
        "<0>",
        "EOF",
        "<2>",
        "NULL_TREE_LOOKAHEAD",
        "ALPHA",
        "DIGIT",
        "WS",
        "NL",
        "COMMENT",
        "ROI",
        "PROFIT",
        "LOB",
        "MTR",
        "RT",
        "REVENUE",
        "TXCOST",
        "PENALTY",
        "FSCOPE",
        "ITCOSTS",
        "ROUTERS",
        "STORAGE",
        "SERVERS",
        "BANDW",
        "POWER",
        "PERS",
        "ACTIONS",
        "FCAST",
        "OPTIMIZE",
        "PROVISION",
        "LPAREN",
        "RPAREN",
        "SEMI",
        "LBRACE",
```

```
        "RBRACE",
        "ASGN",
        "ID",
        "PROVACT",
        "NUMBER",
        "STRING",
        "\"to\"",
        "\"minutes\"",
        "\"minute\"",
        "\"seconds\"",
        "\"second\"",
        "\"dollar\"",
        "\"dollars\""
};

}
```

 # Makefile for the SAL programming language

```
GRAMMAR = SALAntlrTokenTypes.java SALAntlrTokenTypes.txt \
          SALAntlrLexer.java SALAntlrParser.java

WALKER = SALAntlrWalker.java \
         SALAntlrWalkerTokenTypes.java SALAntlrWalkerTokenTypes.txt

SALJAVA = SALMain.java ITCost.java LineOfBusiness.java
SALInterpreter.java

GENJAVA = $(GRAMMAR) $(WALKER) $(SALJAVA)

CLASSES = SALMain.class ITCost.class LineOfBusiness.class
SALInterpreter.class \
          SALAntlrTokenTypes.class SALAntlrLexer.class
SALAntlrParser.class \
          SALAntlrWalkerTokenTypes.class SALAntlrWalker.class \


all : $(CLASSES)

$(CLASSES): %.class : %.java $(GENJAVA)
      javac  $<

$(GRAMMAR): sal.g Makefile
      rm -f $(GRAMMAR)
      java antlr.Tool $<

clean:
      rm -f *.class *~
```

#Regression Test Bucket Harness and Testcase Generator

```
for ($testcase=0; $testcase<50; $testcase++)
{
      #initialize result checks
      $LargestCurrentRevenue = -9999999;
      $LargestProjectedRevenue = -999999999;
      $LargestProjectedChange = -999999999;

      #run testcase types
      for ($testtype=0; $testtype<4; $testtype++)
      {
      print "\n";
      print "running testcase: $testcase\n";
      system ("java SALMain $testtype.sal>output.txt");

      #check results
      open(FILE, "output.txt");

      while ($line = <FILE>)
      {
         if ($line =~ m"Line Of Business: (\w+)")
         { $LOB = $1;
         }
         if ($line =~ m"Current Revenue (\d+)")
         { $CurrentRevenue = $1;

         }
         if ($line =~ m"Forecasted Revenue (\d+)")
         { $ProjectedRevenue = $1;

         }
         if ($line =~ m"Projected Revenue change *(\d+)%")
         { $ProjectedChange = $1;

         }

         #Get Correct Results
         if ($CurrentRevenue > $LargestCurrentRevenue)
         {  $LargestCurrentRevenue = $CurrentRevenue;
            $LargestCRLOB = $LOB;

         }
         if ($ProjectedRevenue > $LargestProjectedRevenue)
         {  $LargestProjectedRevenue = $ProjectedRevenue;
            $LargestProjectedRLOB = $LOB;
         }
         if ($ProjectedChange > $LargestProjectedChange)
         {  $LargestProjectedChange = $ProjectedChange;
            $LargestProjectedCLOB = $LOB;
         }

         #Compare Results with Output
         if ($line =~ m"Forecasted Largest Revenue: (\w+)")
         { $LOBFLR = $1;
           if ($LOBFLR eq $LargestProjectedRLOB)
           {   print "Forecasted Revenue successful\n"
```

```
            }
            else
            {   print "Forecasted Revenue unsuccessful. Results in
output.txt\n";

            }
          }
          if ($line =~ m"Forecasted Largest Growth: (\w+)")
          { $LOBFLG = $1;

            if ($LOBFLG eq $LargestProjectedCLOB)
            {   print "Forecasted Growth successful\n"
            }
            else
            {   print "Forecasted Growth unsuccessful. Results in
output.txt\n";

            }


          }
          if ($line =~ m"Current Largest Revenue: (\w+)")
          { $LOBCLR = $1;

            if ($LOBCLR eq $LargestCRLOB)
            {   print "Current Revenue successful\n"
            }
            else
            {   print "Current Revenue unsuccessful. Results in
output.txt\n";
                #exit(0);
            }
          }
       } #end read file
       close (FILE);
       $testcase++;

    }# end run all test types
     $testcase--;

} #end run all testcases
```

A Sampling of Test cases:

```
:ReturnOnInvestment;
{
  :Revenue;
  {
      :LineOfBusiness InternationalNews;
      {
        :MeanTimeToRecovery = 2 minutes;
        :ResponseTime = 5 seconds;
        :Profit = 10 dollars;
        :TxCost = 5 dollars;
        :Penalty = 2 dollars;
      } //End International News

      :LineOfBusiness NationalNews;
      {
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
       } //End National News

      :LineOfBusiness LocalNews;
      {
        :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 20 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700;
        (
            :Penalty = 100 dollars;

        )
       } //End Local News
   } //End Profit

   :ITCosts;
   {
      :Routers = no;
      :Storage = yes;
      :Servers = yes;
      :Bandwidth = no;
      :Power = yes;
      :Personnel = 200;
   }

   :Actions;
   {
      :Forecast = no;
      :Optimize = no;
      :Provision (:none);
   }
} //End ROI
```

```
:ReturnOnInvestment;
{
  :Revenue;
  {
      :LineOfBusiness OnlineBookSales;
      {
        :MeanTimeToRecovery = 3 minutes;
        :ResponseTime = 5 seconds;
        :Profit = 1 dollars;
        :TxCost = 5 dollars;
        :Penalty = 21 dollars;
      } //End International News

      :LineOfBusiness BookStoreSales;
      {
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
       } //End National News

      :LineOfBusiness UsedBookSales;
      {
        :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 20 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700;
        (
            :Penalty = 100 dollars;

        )
       } //End
   } //End

   :ITCosts;
   {
      :Routers = no;
      :Storage = yes;
      :Servers = yes;
      :Bandwidth = no;
      :Power = yes;
      :Personnel = 200;
   }

   :Actions;
   {
      :Forecast = yes;
      :Optimize = yes;
      :Provision (:CurrentRevenue);
   }
} //End ROI
```

```
:ReturnOnInvestment;
{
  :Revenue;
  {
      :LineOfBusiness TruckSales;
      {
        :MeanTimeToRecovery = 20 minutes;
        :ResponseTime = 3 seconds;
        :Profit = 10 dollars;
        :TxCost = 4 dollars;
        :Penalty = 2 dollars;
      } //End International News

      :LineOfBusiness CarSales;
      {
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
       } //End National News

      :LineOfBusiness ScooterSales;
      {
        :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 20 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700;
        (
            :Penalty = 100 dollars;

        )
       } //End
   } //End

   :ITCosts;
   {
      :Routers = no;
      :Storage = yes;
      :Servers = yes;
      :Bandwidth = no;
      :Power = yes;
      :Personnel = 200;
   }

   :Actions;
   {
      :Forecast = yes;
      :Optimize = yes;
      :Provision (:Growth);
   }
} //End ROI
```

```
:ReturnOnInvestment;
{
  :Revenue;
  {
      :LineOfBusiness CasualClothes;
      {
        :MeanTimeToRecovery = 2 minutes;
        :ResponseTime = 5 seconds;
        :Profit = 10 dollars;
        :TxCost = 5 dollars;
        :Penalty = 2 dollars;
      } //End International News

      :LineOfBusiness BusinessSuits;
      {
        :MeanTimeToRecovery = 5 minutes;
        :ResponseTime = 1 minute;
        :Profit = 6 dollars;
        :TxCost = 2 dollars;
        :Penalty =  0 dollars;
       } //End National News

      :LineOfBusiness VacationClothes;
      {
        :MeanTimeToRecovery = 1 minute;
        :ResponseTime = 1 second;
        :Profit = 20 dollars;
        :TxCost = 10 dollars;
        For Scope = 900 to 1700;
        (
            :Penalty = 100 dollars;

        )
       } //End
   } //End

   :ITCosts;
   {
      :Routers = no;
      :Storage = yes;
      :Servers = yes;
      :Bandwidth = no;
      :Power = yes;
      :Personnel = 200;
   }

   :Actions;
   {
      :Forecast = yes;
      :Optimize = yes;
      :Provision (:ProjectedRevenue);
   }
} //End ROI
```

Bibliography

[1] Conway, Christopher, Li Cheng-Hong, Pengelly Megan. Pencil: A Petri Net Specification Language for Java. Columbia University, COMS W4115, December 2002