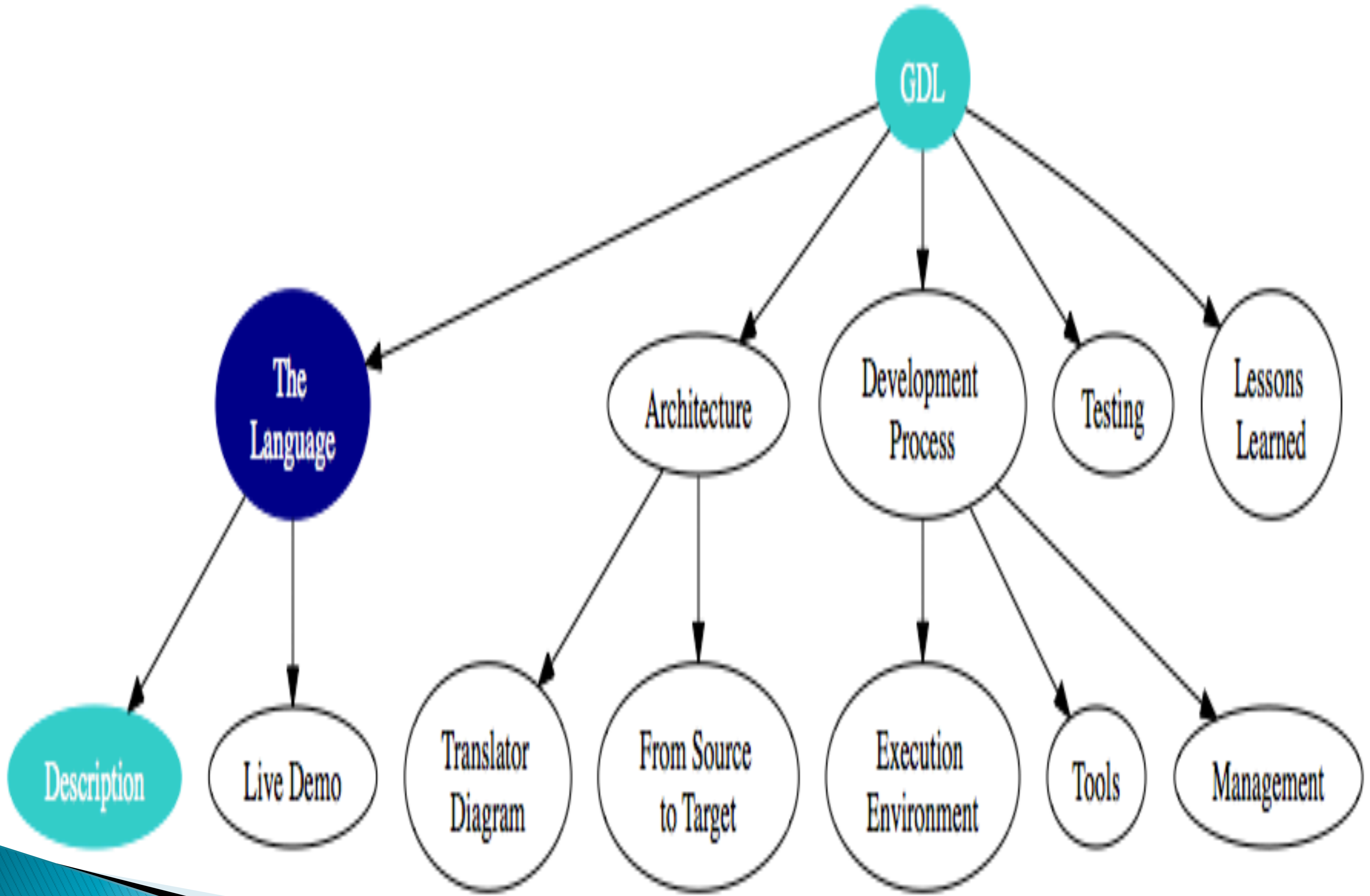
GDL

(pronounced gōōdəl)

Goodle Manager – Lindsey Heller
Goodle Guru – Joseph Corbisiero
Goodle Architect – Ilan Elkobi
Goodle Integrator – Henrique Maia
Goodle Tester – Elayna Tuck


GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

Why GDL…

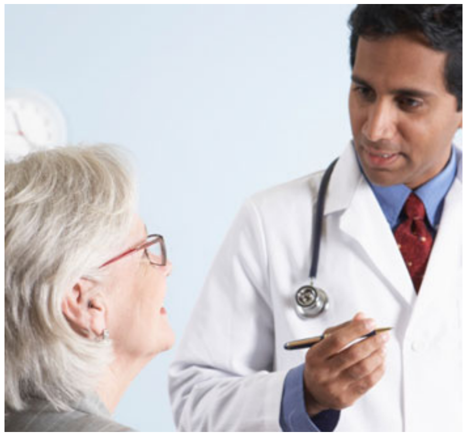
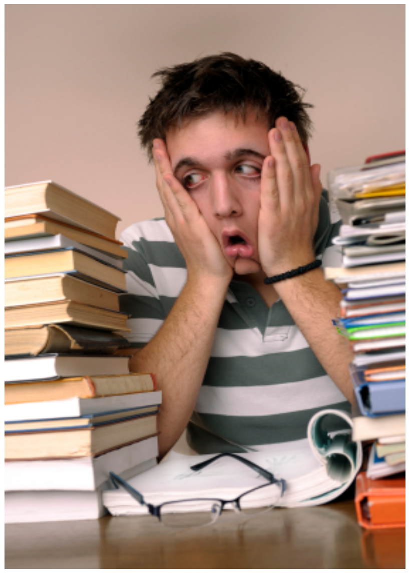
makes programming decision trees easy!

Buzz Words

- Flexible
- Familiar
- User-Friendly / Easy to Use
- Useful

Target

- Professionals
- Students
- Publications

Syntax: Keywords

Similar to Java & C	GDL Specific
`for`	`begin`
`while`	`graph`
`if`	`state`
`else`	`start`
`return`	`accept`
`true/false`	`func`
	`goto`

Syntax : Primitive Types

`string`	equivalent to a Java `String`
`number`	equivalent to the Java primitive `double`
`bool`	equivalent to the Java primtive `boolean`

Syntax : Conditionals

- `while`
- `for`
- `if/else`

- `goto`

```
goto, {list_states}, condition;
```

Syntax: Graphs

- begin

```
begin( )
{
        //states and functions
}
```

- graph

```
graph <name>( )
{
        //states and functions
}
```

Syntax : States

- start = the start state of a graph

```
start <name>( )
{
        //actions
}
```

- accept = accepting state of a graph

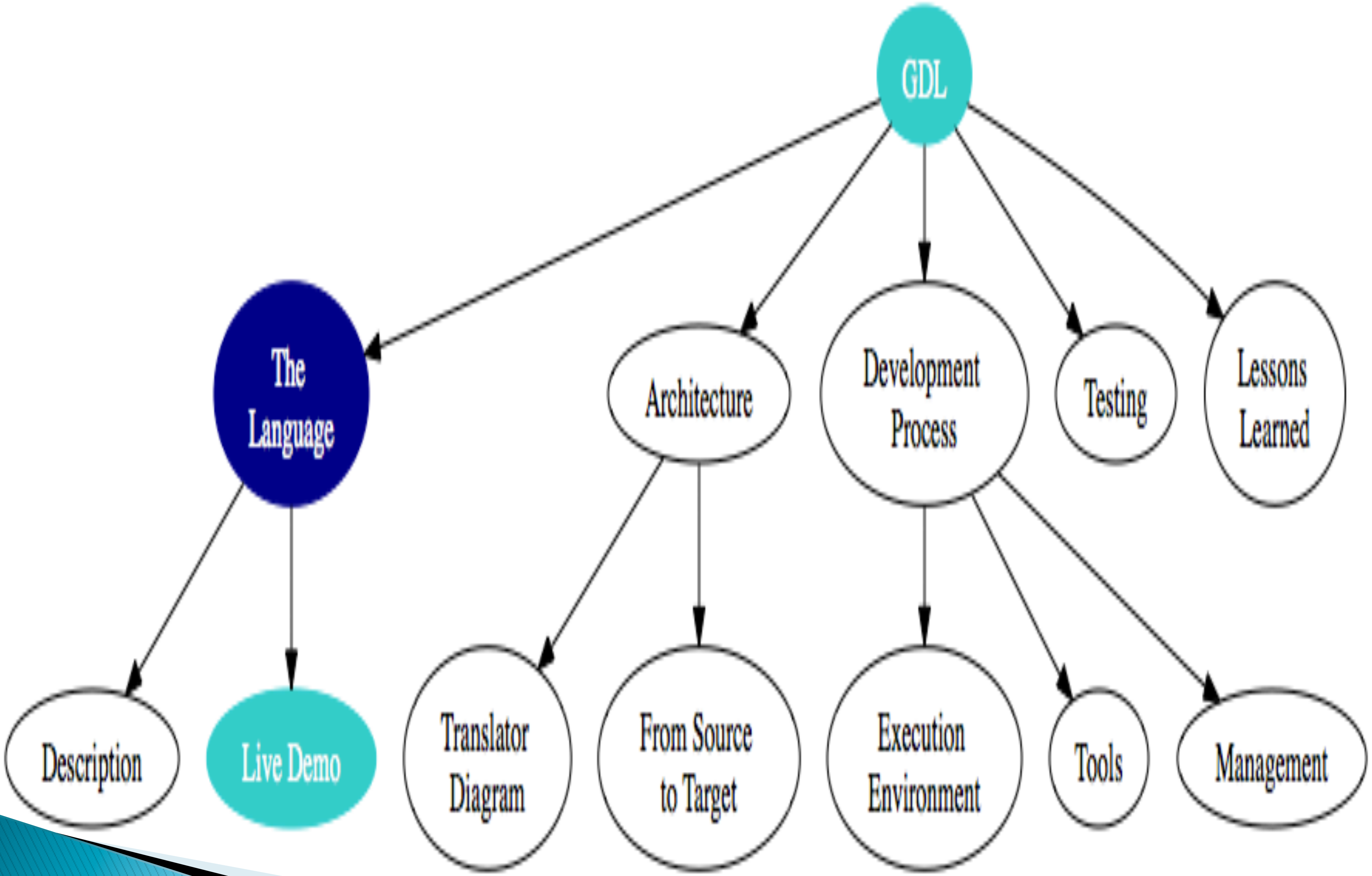
```
accept <name>( )
{
        //actions
}
```

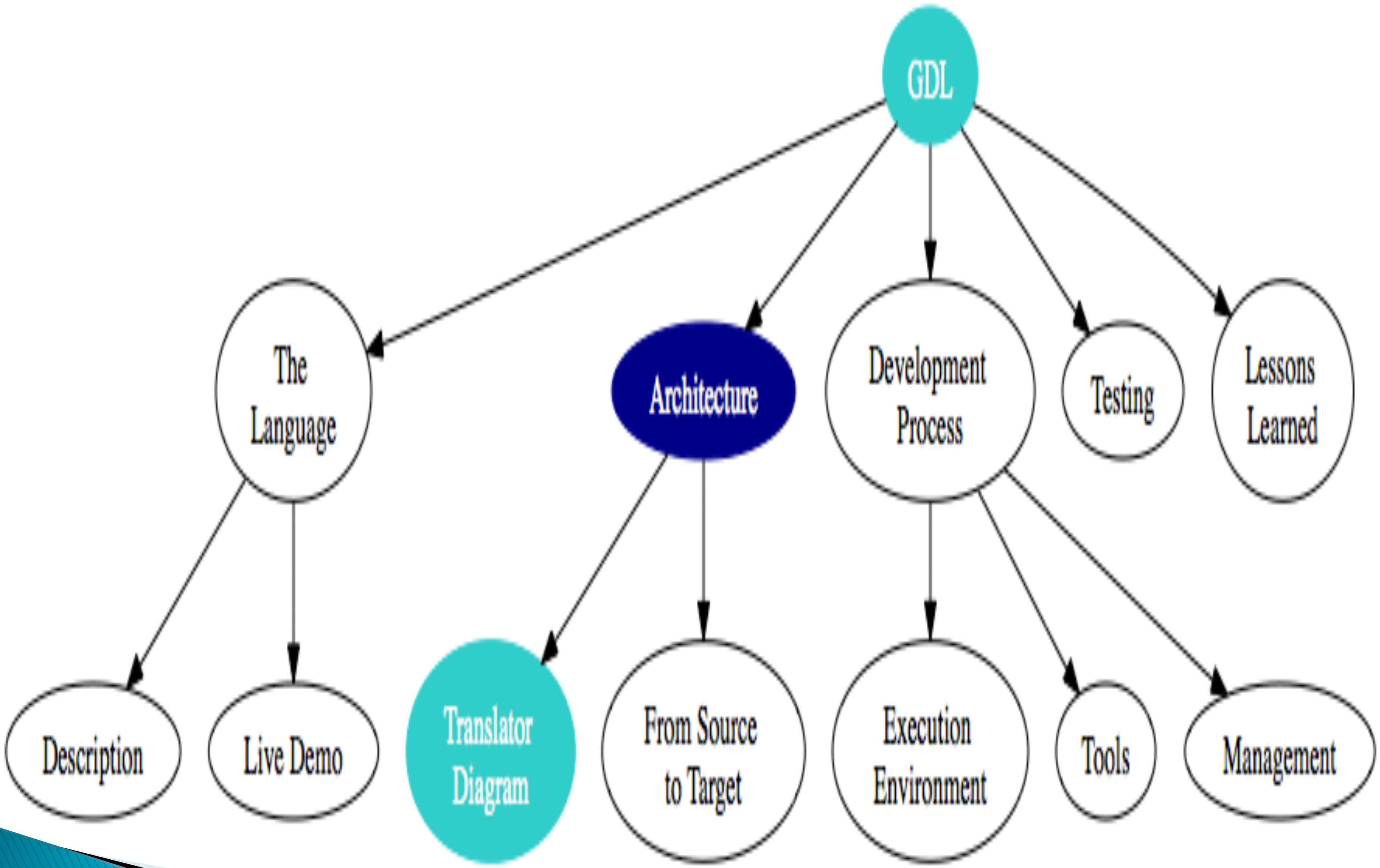
- standard = any state that is neither the start nor accept stat of a graph

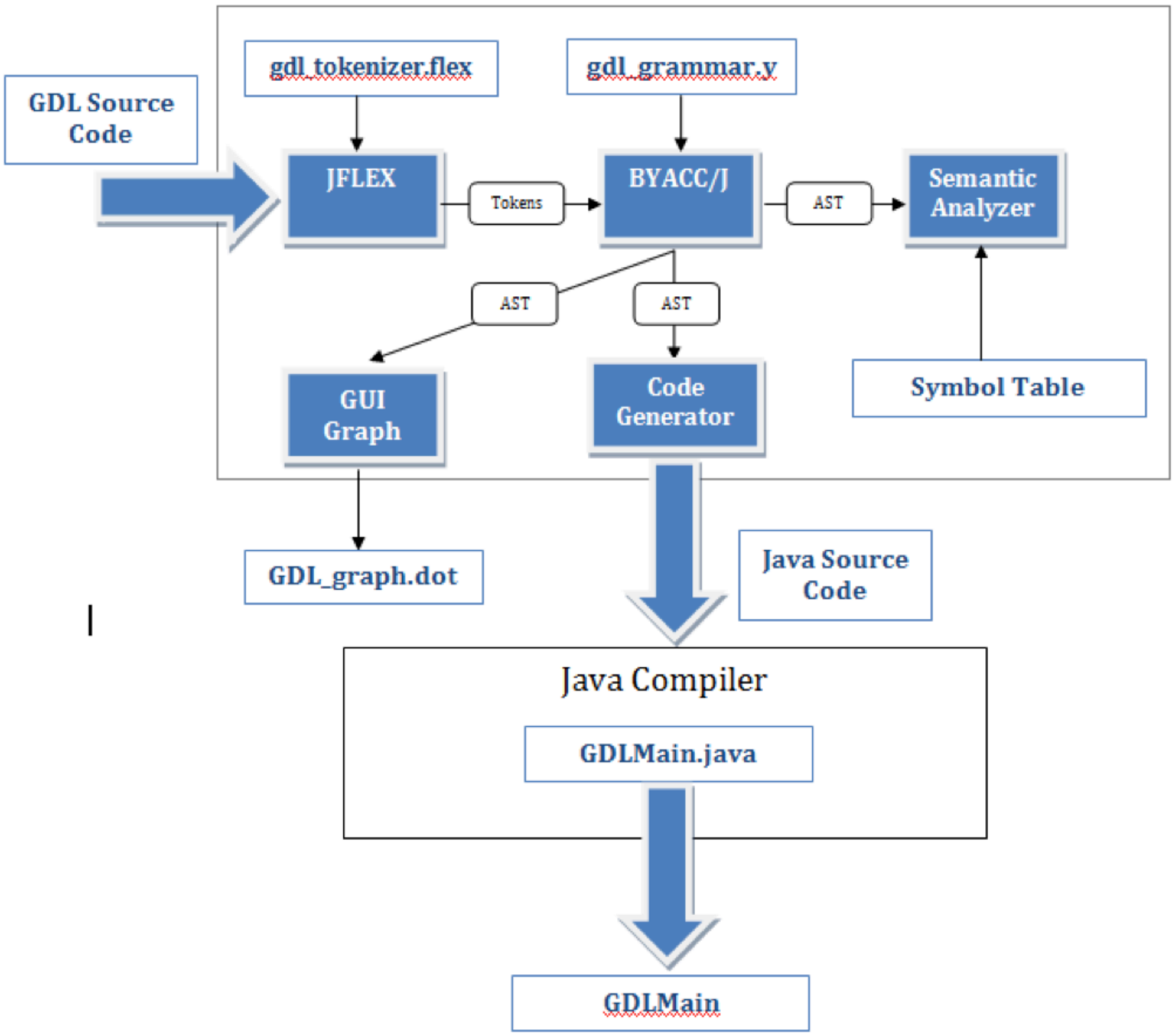
```
state <name>( )
{
        //actions
}
```

Syntax : Function Declaration

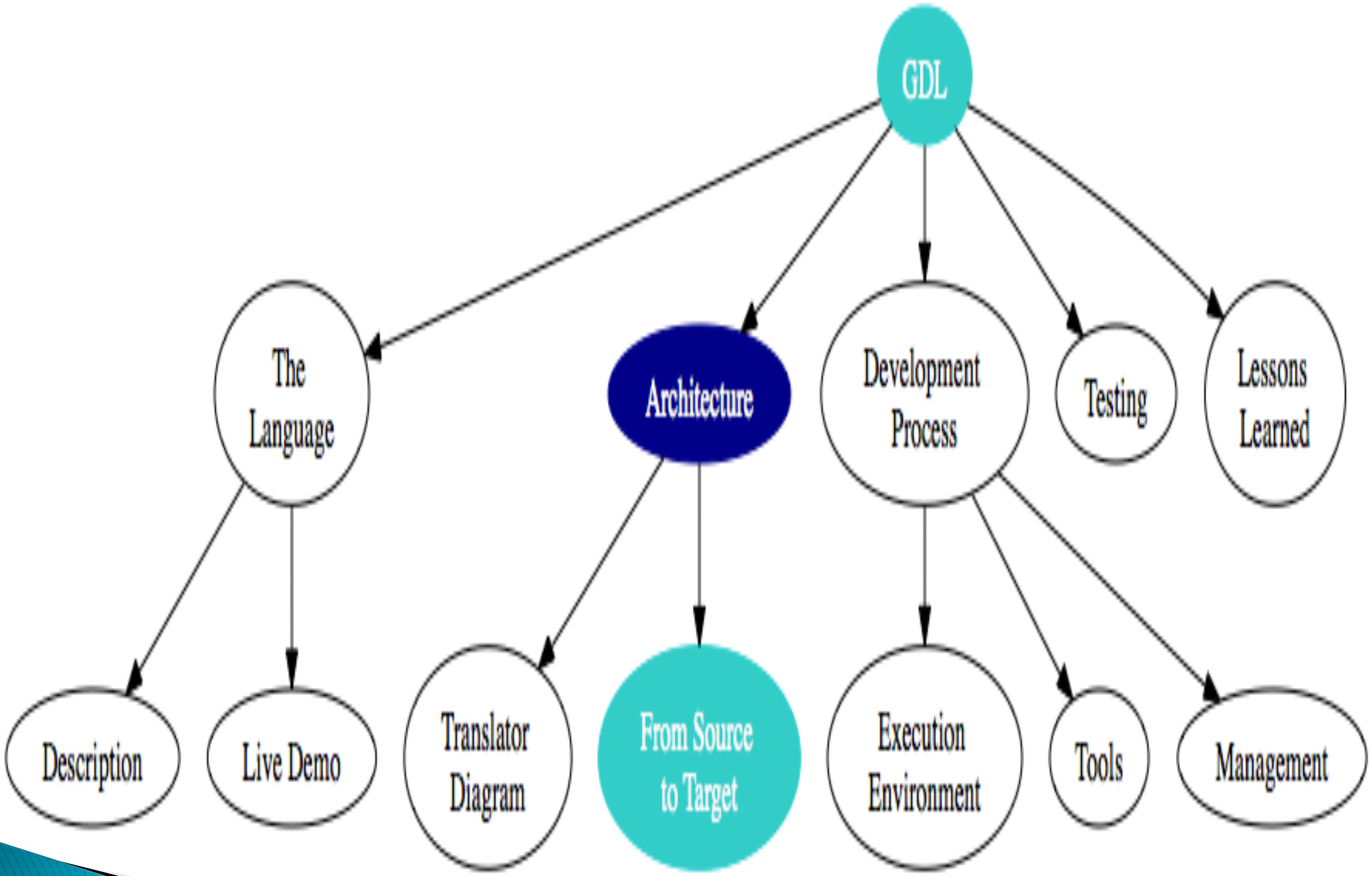
```
func return_type : <name> (parameter_list)
{
    //actions
}
```


GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

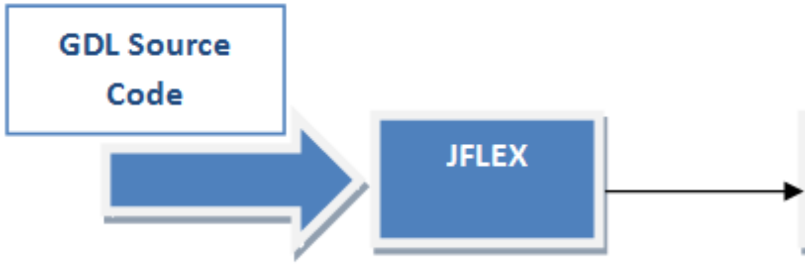

GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management



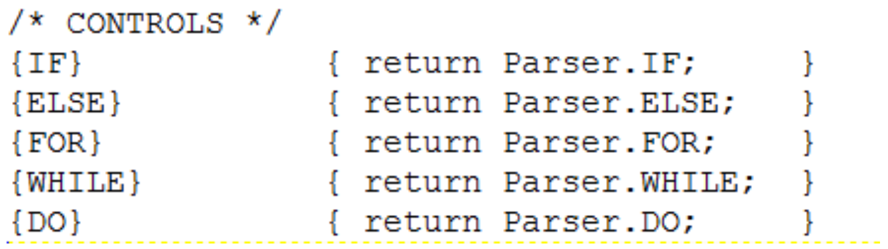
GDL Source Code
gdl tokenizer.flex
gdl grammar.y
JFLEX
Tokens
BYACC/J
AST
Semantic Analyzer
AST
AST
GUI Graph
Code Generator
Symbol Table
GDL_graph.dot
Java Source Code
Java Compiler
GDLMain.java
GDLMain


GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

Lexical Analyzer



- Returns Tokens
- Keywords of the language



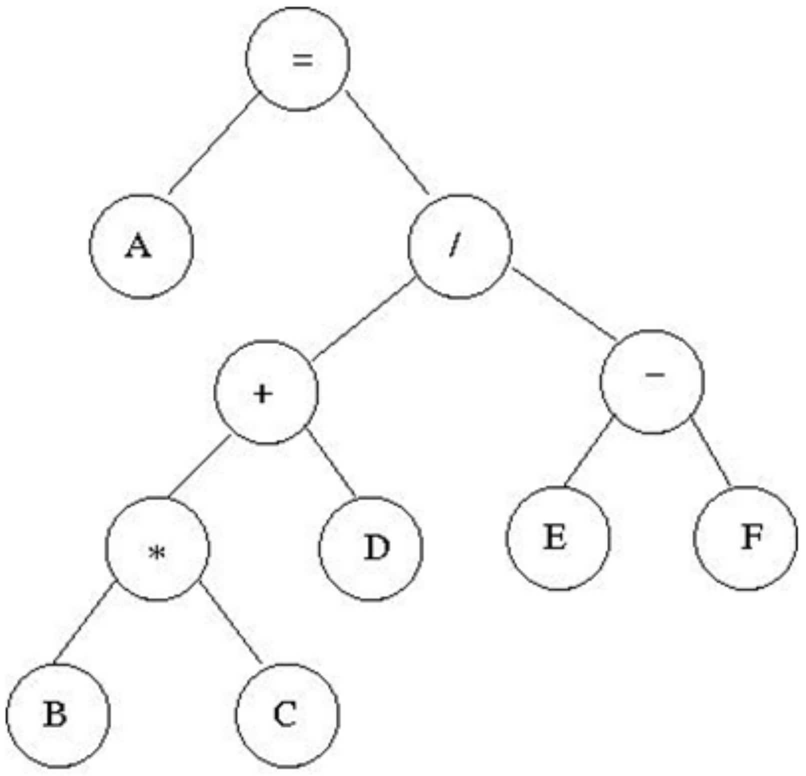
```
/* CONTROLS */
{IF}            { return Parser.IF;     }
{ELSE}          { return Parser.ELSE;   }
{FOR}           { return Parser.FOR;    }
{WHILE}         { return Parser.WHILE;  }
{DO}            { return Parser.DO;     }
```

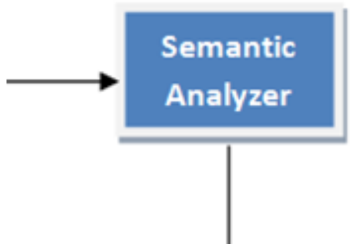
Syntax Analyzer

- Using BYAAC/J
- Creates an AST

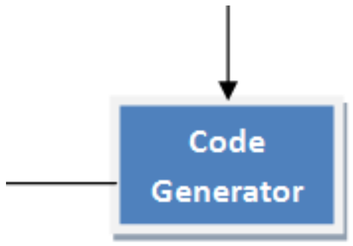
```
stmt          : decl SEMI                        { $$ = new Node( State.STMT, $1 );                  }

              | stmt_assign SEMI                 { $$ = new Node( State.STMT, $1 );                  }

              | begin                            { $$ = new Node( State.STMT, $1 );                  }

              | graph_closure                    { $$ = new Node( State.STMT, $1 );                  }

              | state_closure                    { $$ = new Node( State.STMT, $1 );                  }
              | start_closure                    { $$ = new Node( State.STMT, $1 );                  }
              | accept_closure                   { $$ = new Node( State.STMT, $1 );                  }

              | goto_stmt SEMI                   { $$ = new Node( State.STMT, $1 );                  }
              | if_stmt                          { $$ = new Node( State.STMT, $1 );                  }
              | while_loop                       { $$ = new Node( State.STMT, $1 );                  }
              | for_loop                         { $$ = new Node( State.STMT, $1 );                  }

              | func                             { $$ = new Node( State.STMT, $1 );                  }
              | func_call SEMI                   { $$ = new Node( State.STMT, $1 );                  }
              | return_stmt                      { $$ = new Node( State.STMT, $1 );                  }

              | NL                               { /* Nothing to do */                               }
              | SEMI                             { /* Nothing to do */                               }
              ;
```

Semantic Analyzer

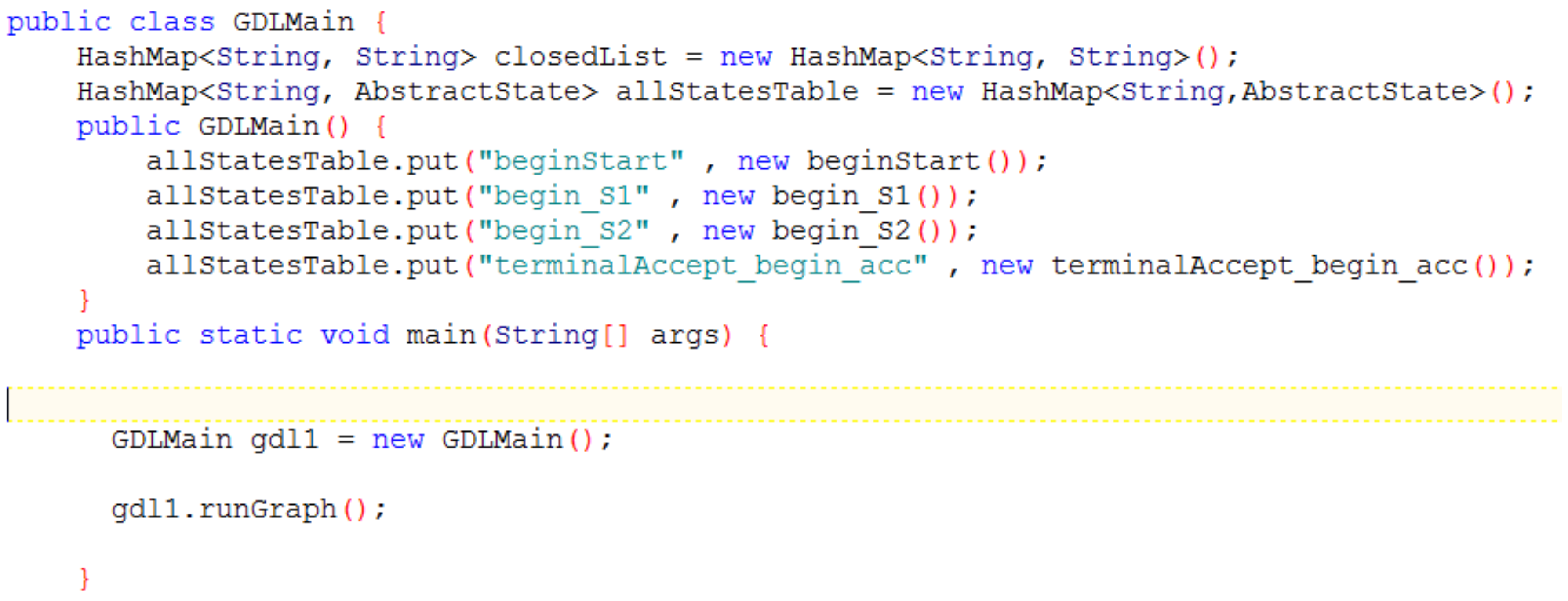




Code Generator



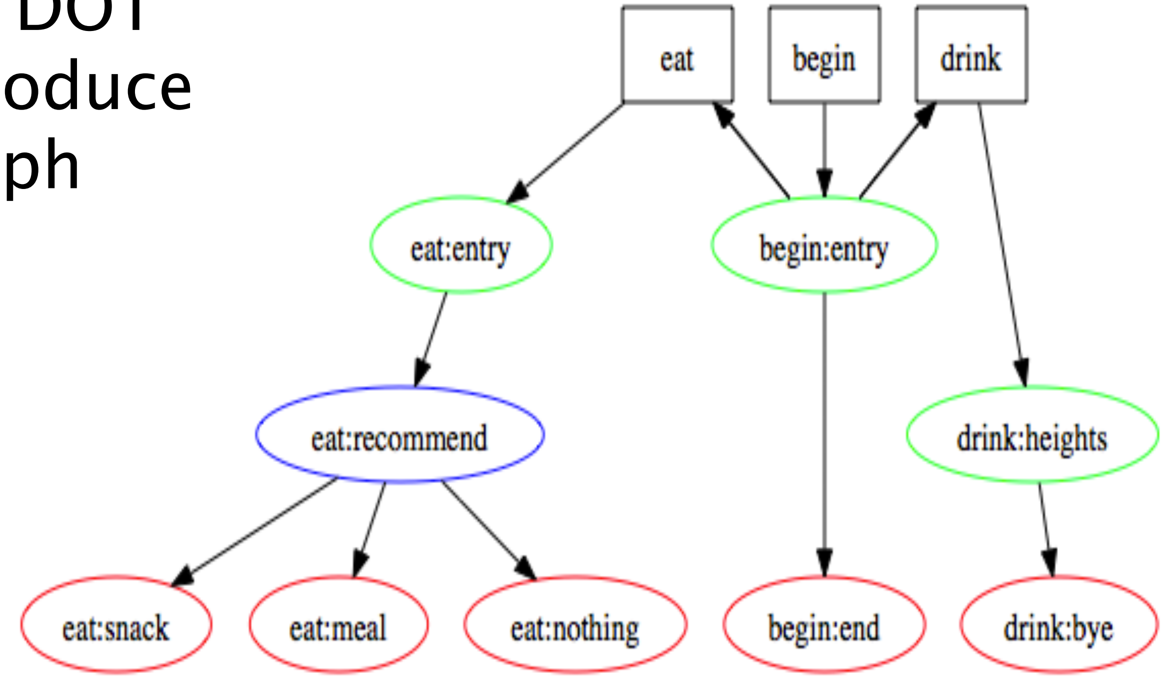
- Creates .java file, GDLMain.java

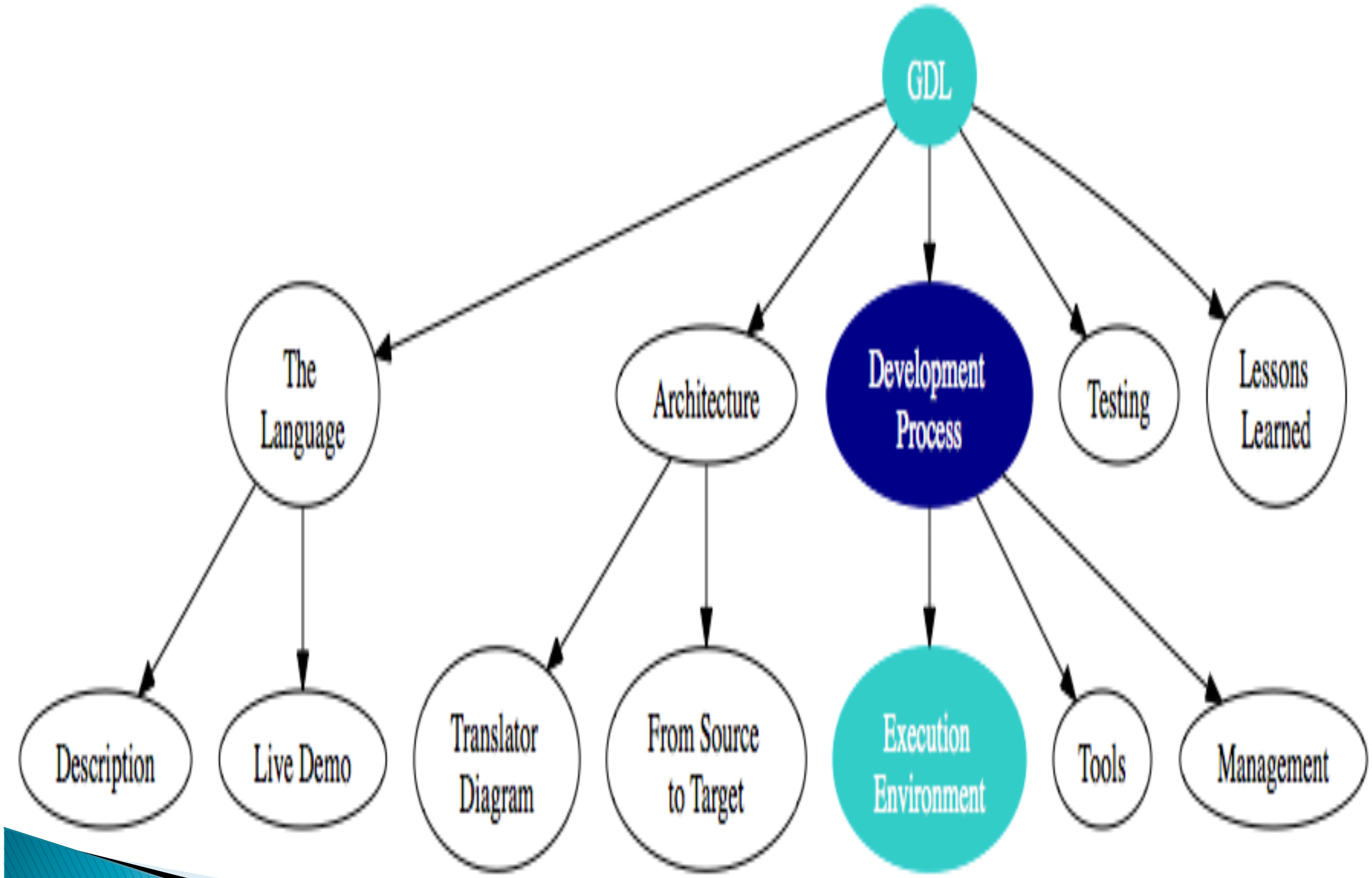


```
public class GDLMain {
    HashMap<String, String> closedList = new HashMap<String, String>();
    HashMap<String, AbstractState> allStatesTable = new HashMap<String,AbstractState>();
    public GDLMain() {
        allStatesTable.put("beginStart" , new beginStart());
        allStatesTable.put("begin_S1" , new begin_S1());
        allStatesTable.put("begin_S2" , new begin_S2());
        allStatesTable.put("terminalAccept_begin_acc" , new terminalAccept_begin_acc());
    }
    public static void main(String[] args) {


      GDLMain gdl1 = new GDLMain();

      gdl1.runGraph();

    }
```

GUI Graph

creates DOT
file to produce
a graph




GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

Execution Environment

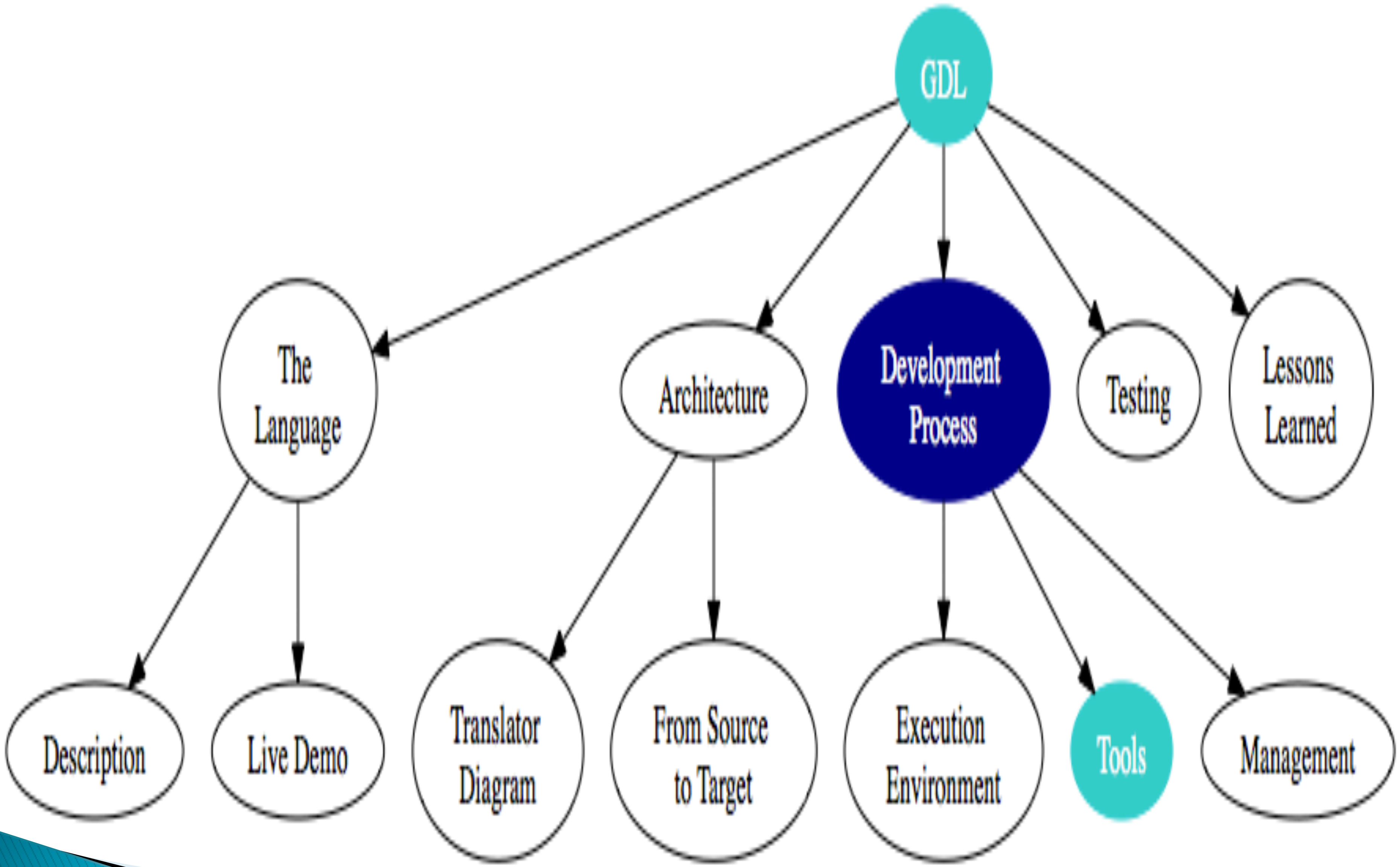
- Makefile calls:
  - Lex and Yacc
    - Creates Parser
      - Parser generates files in output folder
        - Helper classes are compiled and used
          - GDLMain is created with user program
            - Program executes on terminal
              - GDL_graph.dot is generated

Output

- User sees the results of accept states immediately after running the program
- A dot file is generated so the average user can better understand the results of the graph

It's Useful!

GDL has already been put to use for one of our AI projects this year and is currently being used by two team members to create FSM for Embedded Architectures!


GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

Development Tools

BYACC/J

graphviz

JFlex

git

Management Tools

groupme

github
SOCIAL CODING

Google
groups

Google Drive

Scrum

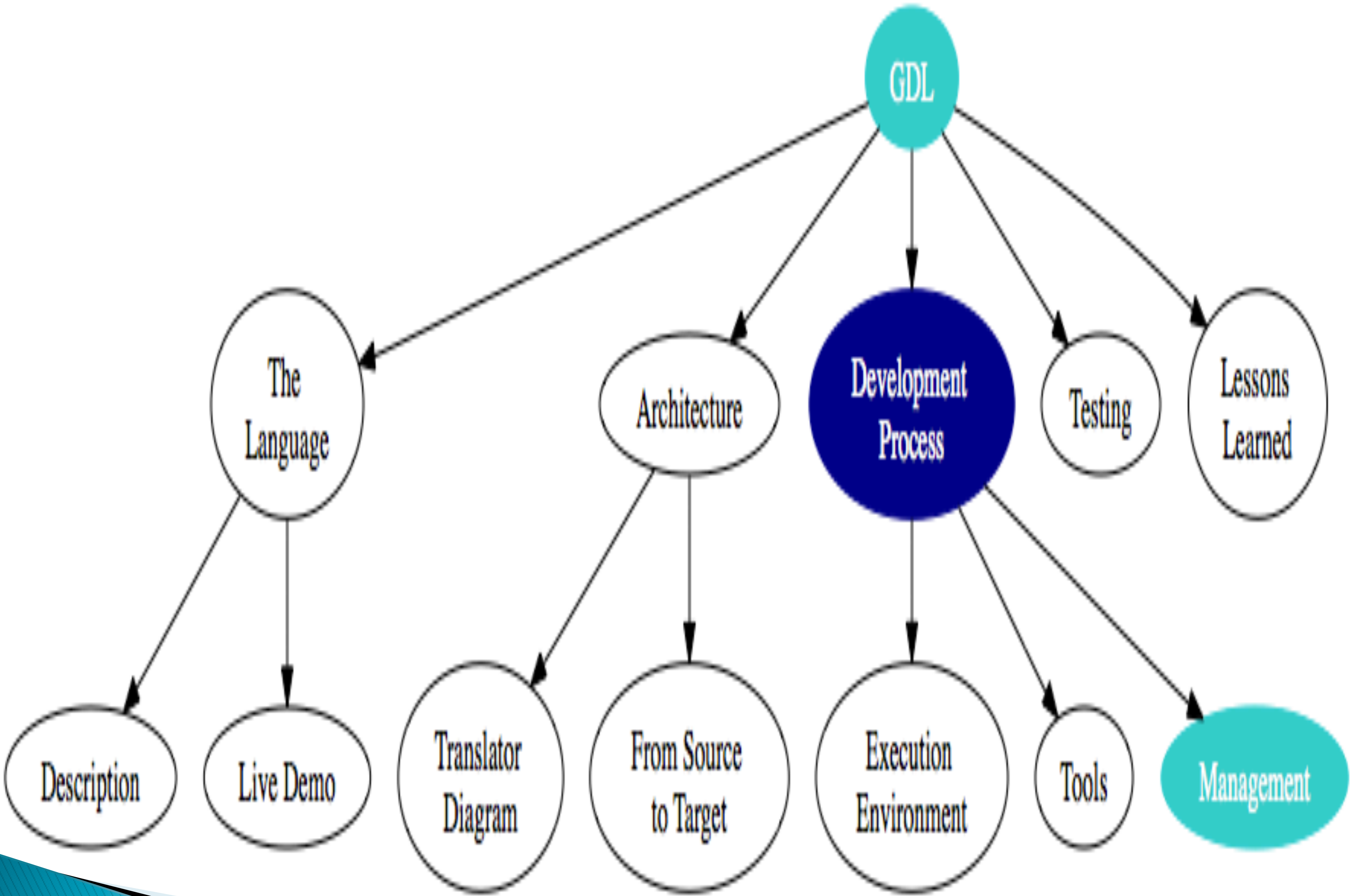
Efficiency Tools

ALWAYS FRESH
Tim Hortons®
COFFEE & BAKED GOODS

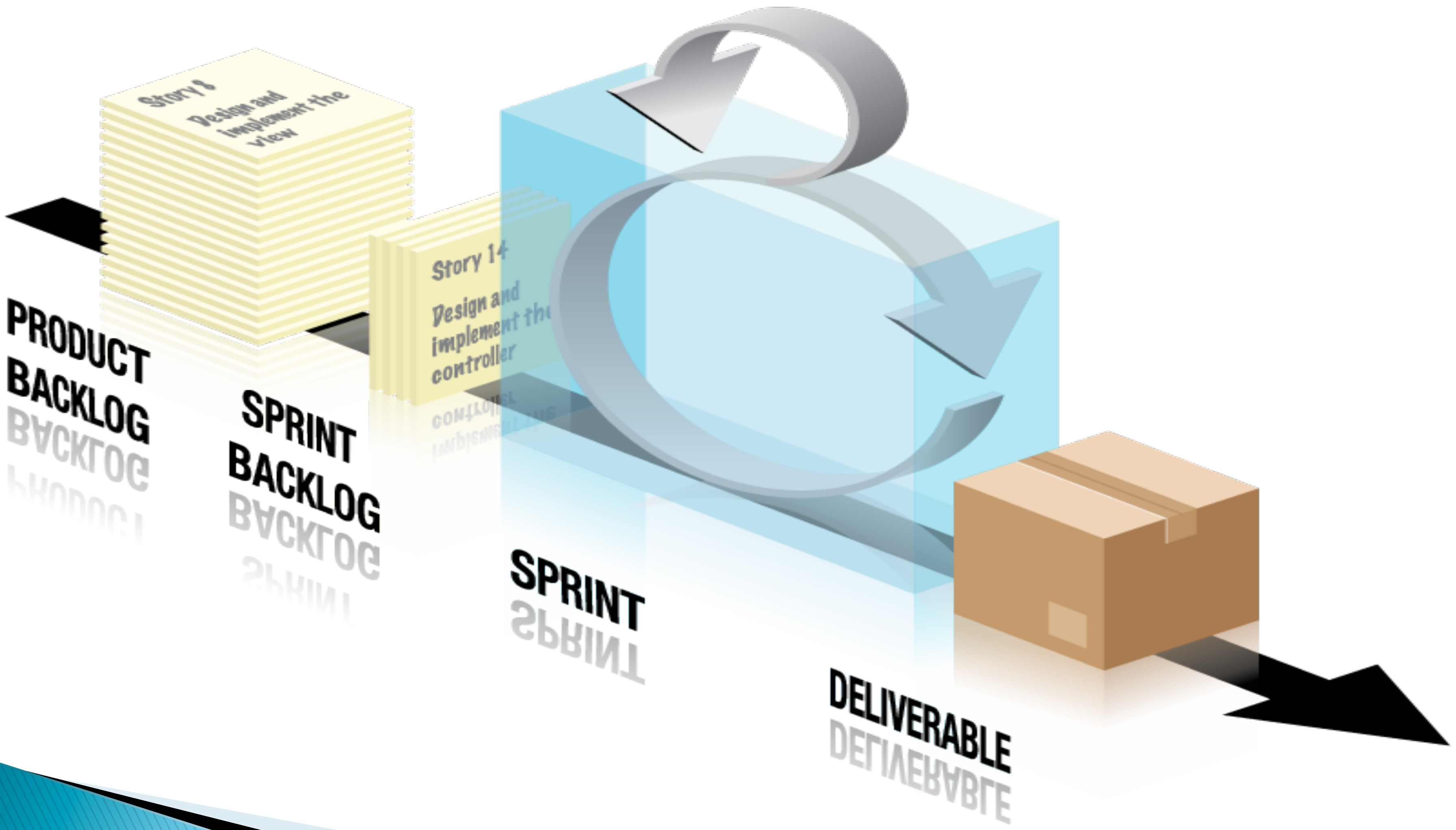
joe

STARBUCKS COFFEE®

DUNKIN' DONUTS®

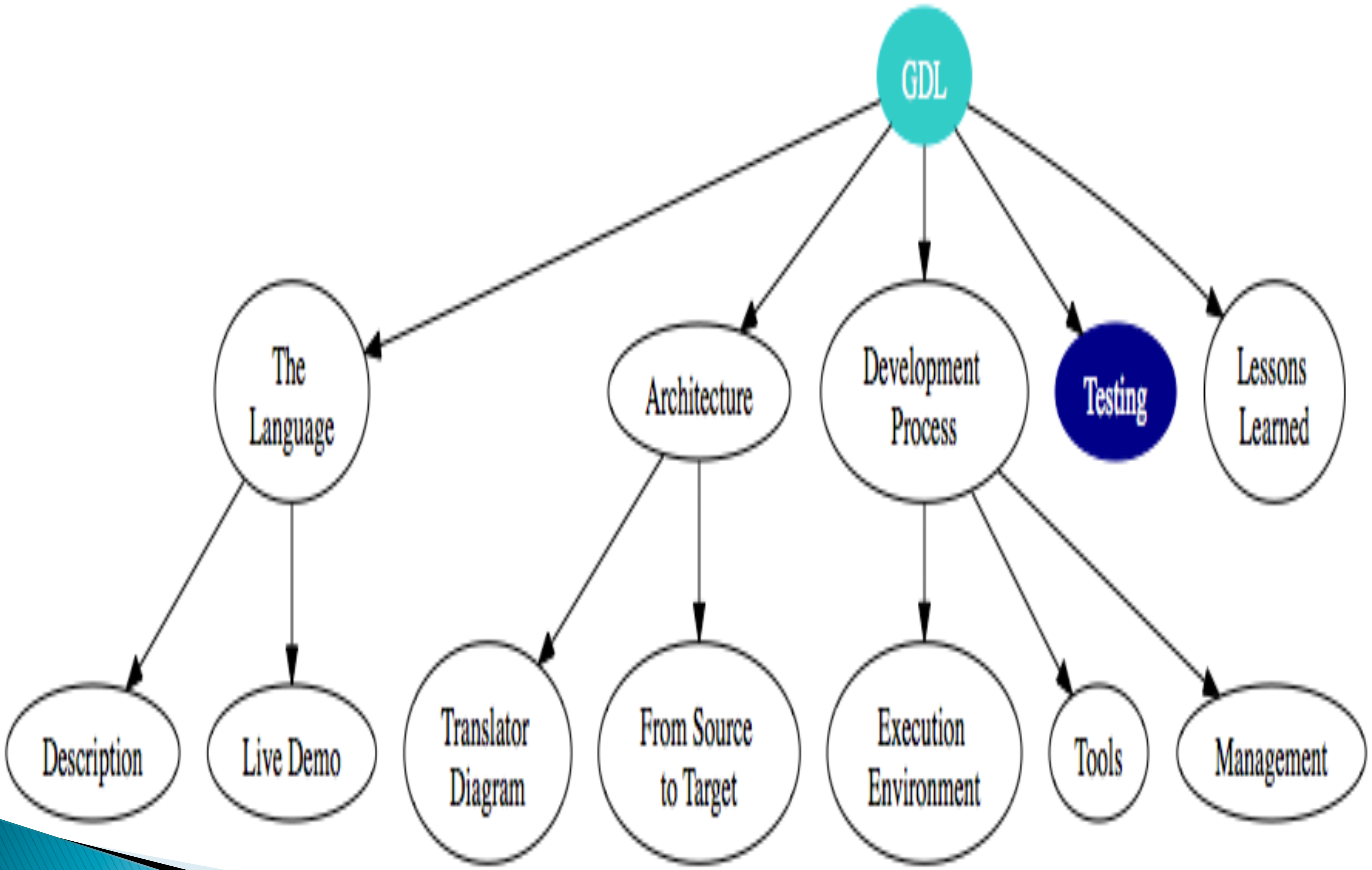

GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

Scrum


Story 8
Design and implement the view
Story 14
Design and implement the controller
PRODUCT BACKLOG
SPRINT BACKLOG
SPRINT
DELIVERABLE

Sprint Schedule

Sprint Dates	Sprint Number
March 6th – March 9th	Sprint 0
March 10th – March 16th	Sprint 1
March 17th – March 30th	Sprint 2
March 31st – April 14th	Sprint 3
April 15th – April 27th	Sprint 4
April 20th – May 11th	Sprint 5


GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

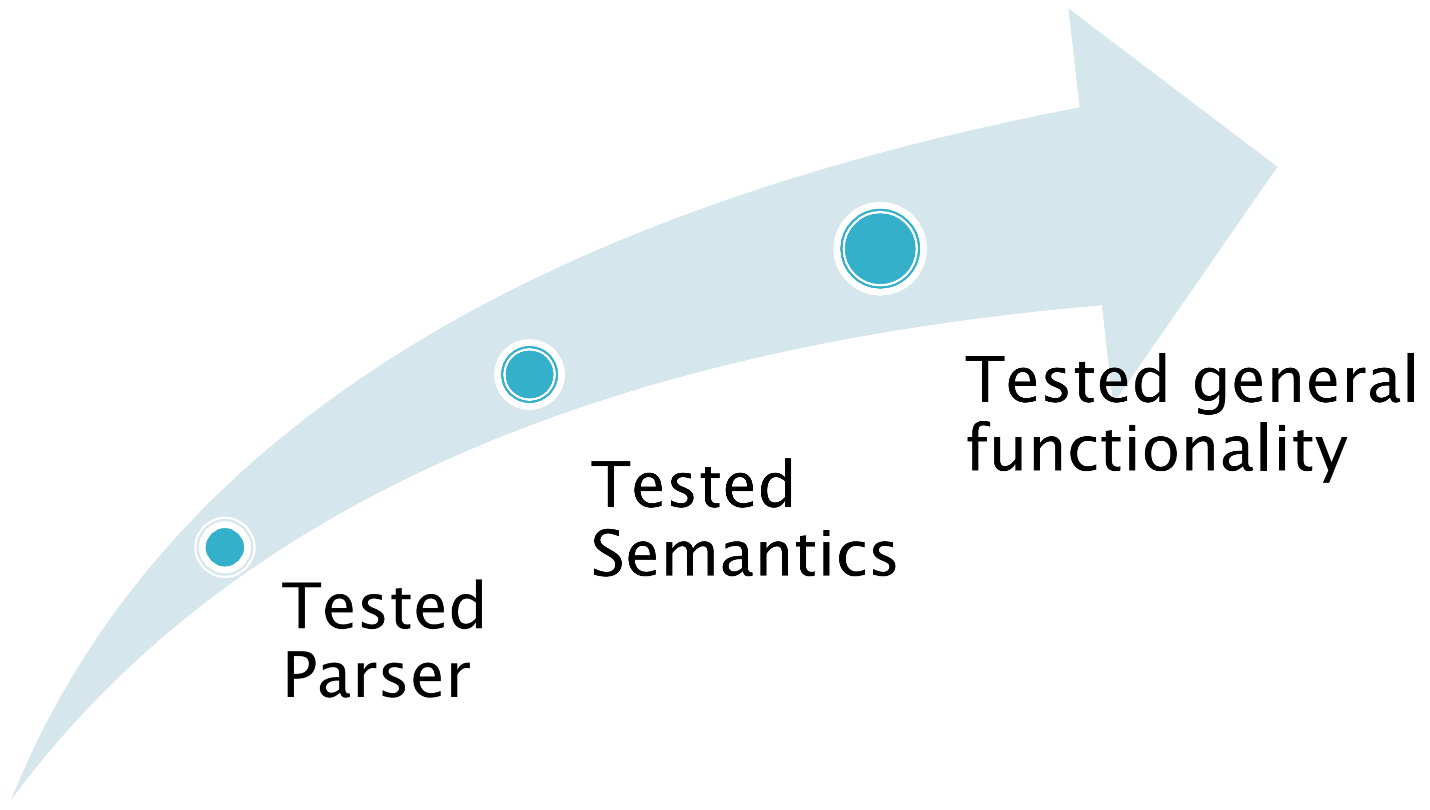
Testing

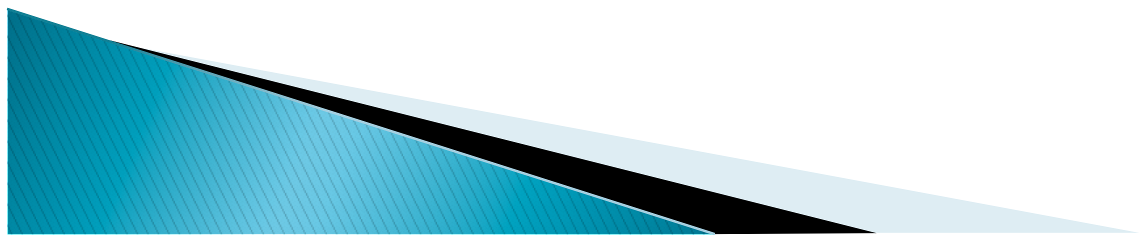
Unit Testing

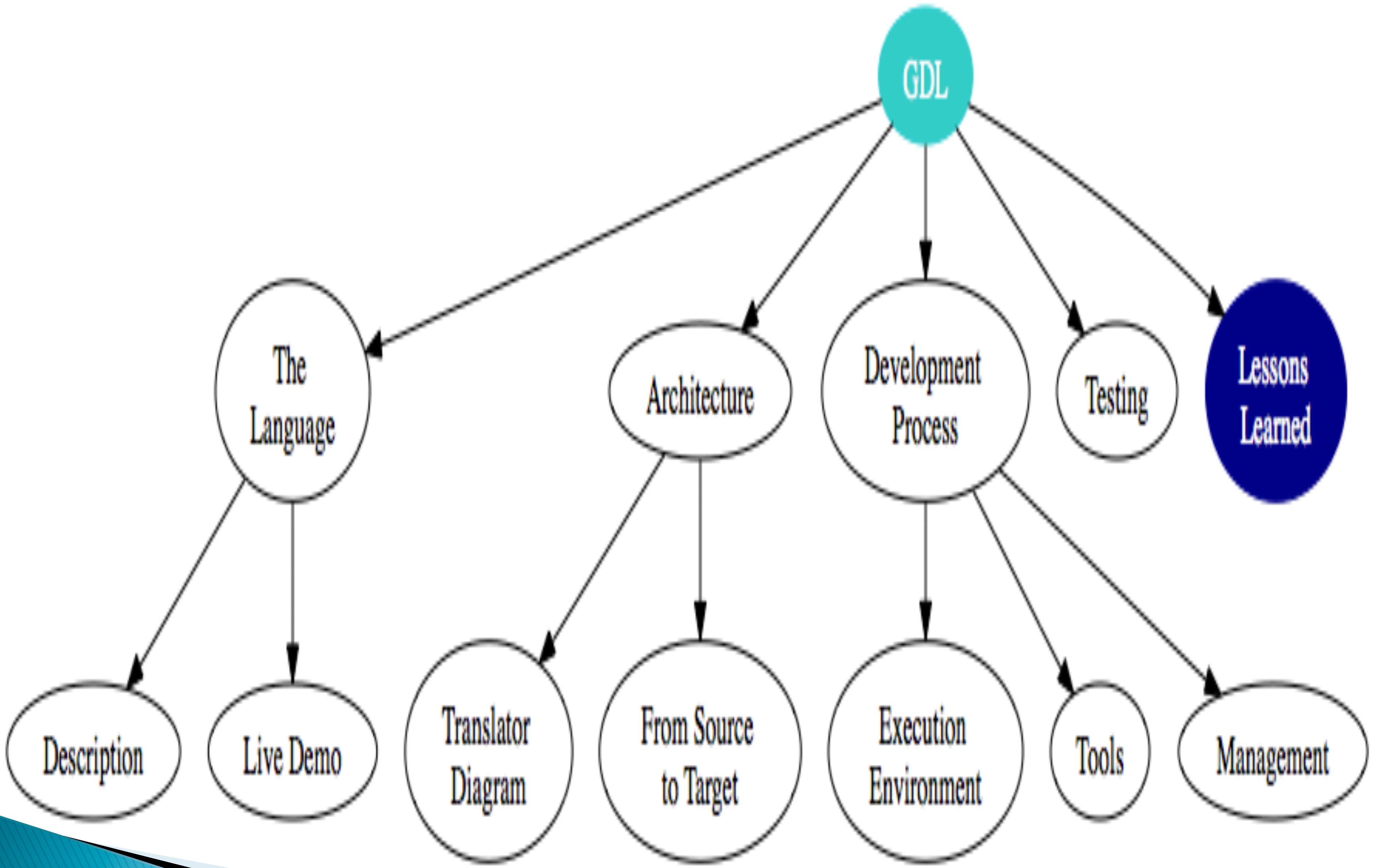
Black Box Testing

Regression Testing

Testing


Tested Parser
Tested Semantics
Tested general functionality




GDL
The Language
Architecture
Development Process
Testing
Lessons Learned
Description
Live Demo
Translator Diagram
From Source to Target
Execution Environment
Tools
Management

We learned our lesson...

- ensure all are using the same version
- always pull before you commit
- communication is key

- test
- test the test
- test the test that tested the test

Questions?