Professor Alfred V. Aho

Team 19		
Project Manager	Chantal Galvez	cg2486@columbia.edu
System integrator	Kunal Mishra	ksm2135@columbia.edu
System Architect	Lixing Pan	lp2441@columbia.edu
Language Guru	Jing Zhang	jz2300@columbia.edu

Introduction

Organizational creation in a programming language requires knowledge of data structures in order to be able to manage the hierarchical and relational aspects of the organization. Obtaining data from the structure also requires the user to know how to structure queries.

NSBL is a programming language that seeks to help group and organizational creation by offering a simple language to create the backend data structures to describe the organization or group topology and to perform complex queries and operations. The language is meant to be simple for the user that is to make the creation of the structure simpler for non advanced users or fast prototyping for more experienced ones.

Buzzwords

Graph-based

The primary characteristic of the **NSBL** language is the graph-based programming style, which is desired for delivering a simple as well as efficient language with regard to processing the general purpose graph-related problems.
The graph-based programming style in the context of the **NSBL** language consists of two levels of meaning. First, the target data and operations are described by the graph data structure. Every vertex in a graph is considered as a container, which stores a finite set of data/properties that are belong to this vertex and are aimed to describe the inherent characteristics, as well as a finite set of operations that can be applied to the stored data. The relationship/interaction between those vertices are described by the directed edges (the topology of the graph), which are associated with the description of the relation of the connected vertices.
Secondly, the code flow of a **NSBL** program is represented by the graph traversal in which a sequence of operations are performed on each pass-by vertex. This general design on the graph-based programming feature distinguishes the **NSBL** language with other available graph-based programming language/API, which is focused only on either graph-based data description or graph-oriented code construction, such as Gremlin, JBOSS, etc.

Highly Abstract

A programmer using **NSBL** does not have to bother regarding the underlying implementation of graph based objects. All that remains to be done is for the user to declare the required object, for example, *Graph g*, and not worry about whether the Graph is going to implemented using an adjacency list or an adjacency matrix. **NSBL** takes care of the implementation. Such high levels of abstraction let the user focus on other design issues of the graph based problem. This also translates into productivity and efficacy.
Through such abstraction, **NSBL** makes it simpler for the user to learn graph data structures easily and enjoy the experience provided by **NSBL**.

Easy Syntax

The purpose of **NSBL** is to make graph programming simple, the syntax also complies with this philosophy. The syntax of **NSBL** is supposed to be easy and intuitive. We do not want a complex syntax be an obstacle for programmers to grasp a new language. We wish any one come to this language can write useful programs as soon as possible. The easy syntax can also make the code succinct and the development efficient. And the style of of graph-oriented programming make program logics very clear and easy to read.

Flexible

The **NSBL** language provides the programmer the flexibility to arbitrarily manipulate the data and operations within the boundary of a vertex. The programmer is also granted full controllability on building up user-defined graph structures that are suitable to specific interested problems.

Portable

Today's computing demands guaranteed performance across multi platforms. Portability is no more a feature, it is a necessity. **NSBL** complies with this necessity. **NSBL** is designed keeping portability in mind. **NSBL**'s performance is independent of the underlying hardware or software architecture.
The reason why **NSBL** leverages portability is because **NSBL** is built on Java. Java itself is portable and only requires JVM to execute its code. And so does **NSBL**.

Interpreted and Interactive

NSBL programs are translated and executed on the fly by the **NSBL** Interpreter. This makes human interactive operations over the underlying data structure possible. Users do not need to think through every states before creating or modifying a graph data structure. They can simply do that step by step and check the topology of the graph on any step. This may reduce errors made by programmers and make things easier. In addition, programmers can benefits from fast development cycles.

Summary

NSBL is a portable, graph-based language, meant to provide an easy syntax to code a highly abstract implementation to problems that require creation of groups or organizations, without the trouble of creating the back-end data structure.
NSBL language is meant to be natural and straightforward, even for the beginner, to be used to simulate the social networks, to implement and to query a graph-based database, or to constructed graph-oriented program.