**NIRAV BHUT**

GIPSY (General Intenstional Programming System) is an ongoing research project developed to investigate on a general solution for the evaulation purpose of programs, using a distributed demand driven evaluation model. It provides framework, which includes more than one tiers.

Organization and management of GIPSY networks were very complex and time consuming, which used to require manual handling using command line console. So new interactive graph based assistant component has been developed that allow users to picture, represent and easilly create, configure and manage those networks virtually as a graph. It also allow users to control the all related network parameters and the interconnection among nodes at run time.

New implementation is more focused on automation such as upon starting, a node reads its preconfigured file and establishes the connection according to the information provided in the file.

The graph based tool assists with the automation of the all system operations such as Configuration, Bootstrapping, GIPSY node registration, Tier Allocation and Tier Deallocation.

The main aim or objective of developing s graph-based graphical user interface is to increase the usability of the running system and allowing the user to have a control over the network with very less manual intervention. This tool provides a drag and drop mechanism to change the conectivity between tiers at run time. It also allows user to seamlessly inspect the status and properties of GIPSY nodes and tiers at run time.

**TIRTH PATEL**

A Type System for Hybrid

Intensional-Imperative

Programming Support in GIPSY

*Tirth Patel (#6893783)*

*Master of Software Engineering (Student)*

*Concordia University*

*Montreal, Canada*

*tirth\_it@yahoo.com*

*SOEN-6611 Software Measurement*

1. BACKGROUND

Lucid is a multidimensional intensional programming languagewhich has implicit Data Types so there will be no type declaration in lucid programming at syntactic level where in language like Java,C++ and other imperative language are explicit so programmer declares data types. Reason behind types system support in GIPSY is because of difficulty in communication between lucid language and imperative language. They want lucid program to can call Java methods and Java method can produce result without any difficulty. The uniqueness of type system presented here is that it bridges gap between intensional and imperative programming language. The introduction of JLucid, Objective Lucid, and the GeneralImperative Compiler Framework (GICF) promptedthe development of the GIPSY Type System as implicitly understoodby the Lucid language and its incarnation within the GIPSYto handle types in a more general manner as a glue between the imperativeand intensional languages within the system. There are so many types in the GIPSY type system and some of them are used by hybrid programming and other is used by compiler (GIPC) andrun-time environment (GEE) to check types. GIPC checks static type and GEE checks dynamic types at run time. Overall this type system provides static and dynamic type check intensional and hybrid programs.

**CHILAT SHAH**

OBJECT-ORIENTED INTENSIONAL PROGRAMMING: INTENSIONAL CLASSES USING JAVA AND LUCID

*Presentation of Object-Oriented Intensional Programming(OO-IP) a new hybrid language.*

*Chilat Shah (6780024)*

*Master of Software Engineering (Student)*

*Concordia University*

*Montreal, Canada*

*Chilatshah1@gmail.com*

*SOEN-6611 Software Measurement*

1. BACKGROUND

Object-Oriented Intensional programming (OO-IP) is a new hybrid language between object oriented and intensianal programming languages., which is new evolutions of Lucid. OO-IP is combination of characteristics of Lucid and java. OO-IP introduce notion of object streams which enable each element in lucid stream to be an object with embedded intensional properties. The difference between Intensional programming and object oriented programming is Intensional programming is an off main stream which is not widely accepted language, whose concrete applicability still need to be proven in order to be widely accepted. While object oriented languages are widely used language and almost known by all computer scientists and software developers.

The GLU – Granular Lucid system which provided with hybrid intensional- procedural by allowing Indexical Lucid programs to use C and functions in order to increase the granularity of data and operators manipulated by the Lucid part. The General Intensional Programming System (GIPSY) provides more flexibility and adaptability then GLU, but it designed using same model as GLU. Due to some flexibility with GIPSY it uses to develop compilers for different variants of Lucid and also allow lucid programs to use procedures defined in virtually any procedural language.

**HIREN**

**Case study: #32**

Design and Implementation of Context Calculus in the GIPSY Environment

**Presented by:**Serguei A. Mokhov,JoeyPaquet and Xin Tong

Concordia University

**Summarized by:**

Hirenkumar R. Tarsadiya

Email Id: hrtarsadiya647@gmail.com

Student Id: 6742904

This study talks about the integration of Luck’s context calculus in the GIPSY where GIPSY gives us the framework to compile and execute programs done using intentional programming language. The main goal of this study is to integrate context calculus theory so to integrate it into already implemented code of GIPSY, Wan has done it through expression of context calculus operators and by simulating context but this approach was not efficient so that is done through conservative extension of GIPL. Here context calculus indicates the relation between dimensions and tags. This paper describes how context class gets fit into system. It has two standard intentional operators i.e. E @ C for evaluating an expression E in context C, and #d for determining the position in dimension d of the current context of evaluation in the context space. The context calculus operators have some semantic restrictions on what are the valid operands. The type checking can be done at compile time or run time depending on situation. The paper concludes by stating thatwith the help of contexts, set of context calculus operators have access of performing operation on the context objects to give us the benefit of constructing and manipulating contexts which can be used in diverse application domains in GIPSY.

**KISHAN SHAH**

Towards Refactoring the DMF to Support JiniandJMS DMS in GIPSY

*Analysis of report on re-engineering effort to refactor and unify*

*two Java distributed middleware technologies- Jimi and JMS*

*Kishan Shah (#6819230)*

*Master of Software Engineering (Student)*

*Concordia University*

*Montreal, Canada*

[*shah.kishan0007@yahoo.co.uk*](mailto:shah.kishan0007@yahoo.co.uk)

*SOEN-6611 Software Measurement*

1. BACKGROUND

The General IntentionalPrograming System (GIPSY) is a unity of Java components arranged primarily into three packages GIPC, runtime programming environment (RIPE) and the general education engine (GEE).It provides the platform of investigation to the possibilities of the intestinal programming. The GIPSY compiler translates any functional programming language paradigm into source language independent GIPL program. GIPSY translate is capable to translate any intentional programming into GIPL program. In the demand driven educational model new demands are generated to request the values of identifiers, and same as these demands eventually generates new demands and so far it makes a chain of demands. In this way identifiers whose value depend on certain demands can be evaluated in turn. The Demand Migration Framework (DMF) is for the distributed execution and DemandMigratingSystem (DMS). Jini and JMS were provided to give a generic framework defining interface to migrate demands generators and workers.

**SOHAN ARGULWAR**

**Topic 24**

**A type system for Higher-Order Intensional Logic Support for variable bindings in Hybrid Intensional – Imperative programs in GIPSY**

**Original Paper by**

Serguei Mokhow

Concordia University, Montreal

**Summary by**

SOHAN ARGULWAR

Student ID: 6757871

sohanargulwar@gmail.com

This research paper is about building a system - General Intensional Programming System (GIPSY) to support intensional programming languages based on intensional logic and their execution model. This framework approach of GIPSY helps developing compiler components of intensional programming languages to execute on language independent run – time system. In, Lucid programming languages data types are implicit i.e. we have to identify the data type by using its result of the expression. GIPSY tries to allow this automatic type conversion when needed to evaluate hybrid expression. This paper present the connection between Java and Lucid.Paper provides the solution for data type casting between Java and other Lucid programming languages. There are some examples stated in paper which describes the situation we GIPSY automatically type cast the data type accordingly when required. One of the example is of void return type in Java which is matched to Boolean true in Lucid. Generally, intensional languages has dynamic data types thus their value is not known at compile time and hybrid languages are generally compile time. The paper then describes the data types like numeric, logical, composite, existential, intersection, union, linear, etc. The paper concludes by stating the implementation of these data types in intensional and hybrid intensional – imperative languages.

**References:**

http://arxiv.org/abs/0906.3919

**MOHIT PUJARA**

**Advances in the Design and Implementation of a Multi-Tier Architecture in the GIPSY Environment**

This paper states that how to increase efficiency in the software engineering design and implementation of the multi-tier run-time system for the General Intensional Programming System (GIPSY) by future implementing the Demand Migration Framework (DMF) in order to streamline distributed execution of hybrid intensional-imperative programs using Java.

Inventors of Lucid’s prescriptive semantics had mentioned the inherent parallelism of lucid programs: “…the whole programs can be understood as producer-consumer networks computing in parallel. Furthermore, this operational interpretation can be used as the basis of a distributed implementation…” However, a multi-threaded and distributed architecture using Java RMI was not fully integrated and many of the detailed working flow needed to be clarified. Consequently, Jini and JMS are not interoperable and their top interfaces are not exactly the same complicating the integration and unification effort thereby delaying the true parallel in the GIPSY’s implementation of the run-time system – the General Eduction Engine (GEE).

To overcome from this problem, we apply most of the high-level design work produced by Joey Paquet by constructing wrapper classes for each tier type introduced in there, specifically DGT (Demand Generator Tier), DST (Demand Store Tier), DWT (Demand Worker Tier) and GMT (General Manager Tier). Every single GIPSY node in the said design translates to a single physical computer who has been registered within the current GIPSY network of nodes participating in computation can host arbitrary number of instances of each tier. Now, we decided to integrate four local and distributed computation prototypes which are multi-threaded and RMI, Jini and JMS together by applying the abstract factory, factory method and strategy design patterns aiming at constructing a framework with high extensibility and maintainability for the further iterations.

In this paper, architecture proposed in these works was itself developed following the generator-worker architecture adopted successfully by GLU but the run-time system of GLU was not as scalable and flexible as the solution that we are designing for implementing in this paper. Therefore, our solution proposes a much more flexible approach by integrating demand migration and storage by using the Demand Migration Framework (DMF), which can be achieved by using various middleware technologies, such as Jini and JMS.

In conclusion, we defined a new package for the multitier implementation and described the details and relationship of the core classes. The multi-tier infrastructure, once fully implemented and tested, will offer high scalability and flexibility. To remain extensible and flexible we added some extra layers of abstraction in terms of the interfaces and APIs.

**AVI MODI**

# The GIPSY Architecture

The current available system for intensional programming tools are getting obsolete as a result of which it suffers from lack of flexibility and adaptability (e.g. GLU). Thus this paper describes the design and implementation goals of GIPSYthat reflects its goals such as generality, adaptability and efficiency. Intensional programming involves programming in a multi-dimensional context space which is included in GIPSY in the form of LUCID. It enables more natural understanding of problems (e.g. tensor programming, distributed OS) of intensional nature. GIPSY consists of three modular subsystems i.e. The General Intensional Programming Language Compiler(GIPL), the General Eduction Engine(GEE) and the Intensional Run-time Programming Environment(RIPE). Compilation of a GIPSY program is done in two stages first the intentional part of a GIPSY program is translated into C and then the resulting C program is compiled in a standard way.The task of GIPL is to translate the LUCID code into Intensional Data dependency Structure (IPS) and Intensional Communication procedures (IDC) for data communication. Since GIPSY is a demand-driven model of computation the task of GEE is to useeduction in conjunction with a value cache called warehouse which reduces the overhead induced by the procedure calls needed for computation of demands. The RIPE subsystem module us to dynamically inspect warehouse calues, change the i/o channels of the program, recompile sequential threads and change the communication protocols and most important of all it enables use to visualize the dataflow diagram of a GIPSY program.