GIPSY bridges gap between intensional and imperative programming languages. Compiler like JLUCID, Objective LUCID and GeneralImperative promoted the development of GIPSY type system. Overall GIPSY system provides static and dynamic type check for intensional and hybrid programs***[T].***

Object Oriented Intensional Programming(OO-IP) is a new hybrid language which is a combination of Object Oriented(JAVA) and Intensional Programming(Lucid) languages. The main need to develop hybrid system is combine the power and concrete applicability of intensional programming and the broad acceptability of Object Oriented language. GIPSY provides support for this type of hybrid languages with great flexibility and adapatibility which is lacking in the currently used GLU system***[C].***

This framework approach of GIPSY helps developing compiler components of intensional programming languages to execute on language independent run – time system. The data type in java is explicit while Lucid programming language uses implicit data types. So GIPSY provides the mechanism of automatic type casting from imperative to intensional and viceversa at runtime***[S].***

The main goal of GIPSY is to provide generality, adaptability and efficiency. It is divided into 3 subsystems, the General Intensional Programming Language Compiler(GIPC), the General Eduction Engine(GEE) and the Intensional Run-time Programming Environment(RIPE). Here GIPL translates the LUCID code into into Intensional Data dependency Structure (IPS) and Intensional Communication procedures (ICP) for data communication, GEE computes the remote procedure call and stores it in the warehouse and hence it helps in reducing the overhead generated by the procedure call, the RIPE module enables use to visualize the dataflow diagram which in turn enables us to dynamically inspect warehouse values and change i/o channels of the program ***[A].***

Earlier it was not possible to assign a context to a variable, pass as a parameter, return as a result of a function and that was necessary for integrating LUCID variants to GIPSY system. To resolve this issue Lucx parser has been developed and by that means we got efficiency of execution much higher than the classical translation methods proposed previously ***[H].***

GIPSY is a demand driven educational model and it creates chain of demands while translating any intentional programming language into GIPL. JINI and JMS are two JAVA distributed middleware used in the implementation of demand migration framework. By doing so we are refactoring their parent DMF***[K].***

To advance software engineering design and implementation of the multi-tier run-time system for GIPSY we decided to integrate four local and distributed computation prototypes which are multi-threaded, RMI, Jini and JMS together by applying the abstract factory, factory method and strategy design patterns***[M].***

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 easily 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. The main aim or objective of developing is 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***[N].***

References:

[T]

[A]

[M]

[N]

[S]

[C]

[K]

[H]