Project Summary

This collaborative research proposal seeks support for a 5 year study of the use of enriched declarative programming languages for system specification and development. The proposed research is aimed at exploiting the potential of recently designed extensions to logic programming languages for writing executable and testable specifications of software, and subsequently prototyping, testing and refining them into a target language.

This potential is well beyond the expressive capabilities of traditional logic programming languages. Furthermore, as *executable* formalisms, declarative languages offer a clear advantage over the leading formal systems for software specification since they can be run at any time in the software development cycle. There is no need to build and rebuild a simulation engine as the software architecture is being developed and refined.

Declarative programming languages have been especially successful in rapid prototyping of software, rule-based systems and metaprogramming, the design of front ends for heterogeneous systems, and in applications requiring the integration of database and programming components. They have been extensively used in US and EU government and industrial applications, as discussed below. However, there are serious limits to the expressive power and efficiency of the original core declarative languages, which have impeded the development of widespread, standardized applications. These limits, which have lead to misconceptions about the potential uses of declarative programming, can be successfully overcome today.

Extensions to the declarative paradigm in the past fifteen years (many developed by the members of this collaborative research initiative, some with US or EU government support) have greatly expanded the scope and potential of declarative programming and have afforded rich new specification languages for the *principled development* and *incremental testing* of correct large scale software systems. This potential remains largely untapped due to the lack of clear principles for the integrated use of such extensions in applications, of tools for constructing and analyzing programs that utilize these extensions and of methods for efficient implementation and compilation. This initiative will begin with a study of principles for specification design and testing, and efficient use of the new declarative tools and techniques for implementing large scale systems and the integration of these tools into actual programming practice.

The proposed work will result in a clear formal definition of the kinds of modules and extensions available, the principles underlying their structure, and models to reason about system behavior. The research will identify principles for embedded and integrated system development, and for writing specifications. This means

- THEORETICAL WORK to characterize language extensions and expressive power, and provide formal frameworks for practical specification disciplines, and models for understanding them. This will include assertion languages for static and dynamic testing of a specification as it is refined.
- TECHNICAL REPORTS on specification development guidelines and program development guidelines, with examples,
- a PROTOTYPE program development environment based on the results of our research program.

  One component of this prototype will be a language for expressing specifications, and for making assertions about program structure and behavior to be tested at compile time and run time. This environment will be able to check whether programs meet certain subsets of the specifications, as well as to statically and dynamically detect and flag incorrectness symptoms in such programs, and infer program properties.

The research will *not* restrict attention to one formal system or ontology (e.g. a given type theory or specification formalism like Z). Rather, it will study the viability of certain logical tools (declarative fragments of higher-order and linear logic, constraints) for software development. These tools, *precisely because they have been subjected to the test of functionality as programming languages*, have the potential to make up one of the most practical and viable frameworks for writing and refining specifications, and the only one which is executable. The principled use of such a framework can significantly contribute to programmers' ability to reliably develop correct, real-world software applications.