IPSE 2.5 - ONE IPSE THAT IS NECESSARY

B.C. Warboys

Rapporteurs: Mr. F.L. Hughes
Mrs. A. Petrie

Extended Abstract:

Introduction

The IPSE 2.5 Project is an Alvey sponsored collaboration between STC, Manchester University (Cliff Jones' group), SERC Rutherford Appleton Laboratories (Chris Wadswcrth's group), Dowty Electronics, Plessey and the British Gas Corporation. The project is of four year's duration and has been running for nearly a year. This first year has been spent in Requirements Analysis and Concept/Architecture formulation. Inevitably therefore this lecture will focus on the objectives and the general framework of the solution strategy rather than details of its implementation. The term IPSE is attributable to the Alvey programme and is an acronym for Integrated Project Support Environment.

Scenario

My last lecture (see previous section) outlined the view that the project has taken of the needs for Environment support in the next decade. The project aims to be an aid to increasing the effectiveness of Software development, from a personal point of view within the STC corporation, and more widely if others can be persuaded of the benefits. To summarise and remind listeners the main attributes of this future scenario are:

- A trend towards systems formed from "off the shelf" components.
- The disparity in lifetimes of many components.
- Trends towards integration/cost reduction for many components.
- Needs to improve product quality to reflect the increasingly endemic nature of IT components in broader ecosystems.
- Product variety gives rise to a multitude of design parameters and methods.
- Rapid changes in the level of interfacer between various suppliers in the value-added solution chain.

Why IPSE is called 2.5?

The three generations of IPSE technology were identified by the Alvey programme (see Rob Witty's lecture). IPSE 2.5 exceeds the second generation because of an emphasis on the support of formal methods and falls short of the aims of the third generation as it has a relative lack of reliance on IKBS techniques.

Motivations

The project takes the view that the essence of such a support environment is the active support of the design process. That the classical view of the development life cycle is too static a representation to act as the basis for a management tool for understanding and directing development projects. Further that current implementations have viewed the IPSE as adding value in specific areas to existing development systems rather than viewing the IPSE or a total ecosystem which needs to completely replace the developers image of his tools set.

It also takes the view that the Component Engineering requirement identifed in the previous lecture can only be satisfied by an increased reliance on Formalism allowing the development process to take greater advantage of evolving program transformation technology in particular allowing developers to practise product validation and maintenance at the earliest possible stages of the life-cycle rather than on some already highly optimised implementation.

Requirements of the project

To support to flexibly support Information Flows, both

- Product flows between process elements
- Process evolution flows between product elements

To support evolution of

- Tools and Methods
- Implementation Technology
- Representation of revised process models

To support diverse user communities

To support evolving industry standard components

To support Component Reuse through Formal Methods

By formal methods we mean

- a given concrete syntax for describing a logical system together with associated semantic/proof/evaluation rules.

- notions of objects and transformations to capture a decomposition of the development process

- procedures which explain the order in which certain objects should be constructed and certain transformations applied

Hence support implies consideration of

- design support - standard transformations and data types

- process support - methods of use described and supported

Project Response

Effective integration of

- Process Support
- Adaptability
- Formal methods

though the production of an advanced IPSE

- Fine grain database
- Active process structures
- Advanced MMI
- Multi-user working

and the demonstration of the IPSE's effectiveness.

The IPSE is characterised as an OPEN environment in three senses of the word

- Allowing other projects to add Tools/Methods/Processes
- Allowing users to use other facilities outside of those provided by the IPSE
- Being portable to other regimes

Technical Themes

- Support for the requirements of a software project
- Architectural themes
    - Convergence of fine grain data structures and large scale development structures
    - Provision of active object store elements
    - Balance of processing capabilities between servers and work stations.
- MMI themes
    - Individual at the workstation is viewed as being part of the IPSE
- Formalism support
    - Emphasis on effective support
    - Basis for component engineering

At the architectual level a distinction can be made between current IPSE's which tend to act as passive repositories of information which force the collaboration between tools to be explicitly defined and implemented in the tools themselves and the ultimate IPSE which describes the management of objects together which their allowed transformations and hence allows the tools to manipulate these objects in a prescribed way without having to implement the conventions for tool collaboration explicitly. IPSE 2.5 takes the view that currently only a relatively small set of logics and processes can be thus described and hence must allow an Aspect type Public Tools Interface to coexist for certain toolsets. However in the case of both tool and process support for formalism the ultimate IPSE is already capable of realisation.

Relationship between tools and the "database"

The project takes the view that

- tools are the means by which elements in the database are supported
- definition of projection functions to yield the MMI are held in the database
- projection functions provide access to overlapping areas of the database and hence allow for
  - design conferencing
  - information sharing between windows

This is a contrast to existing supporting environments which use the database as a passive repository of tools outputs and at best constrain the order of tools execution.

People are viewed as part of the IPSE

Support must be provided for models of the complete development process (which may go well beyond that required for just software production - concerns of total ecosystems must be addressed).

Development is viewed as the "execution of this model"; thus developers act as program counters traversing the "plan program". The process model and project plan are acting as the "programs".

Further the interface between the developer and the IPSE is viewed as that of a giant scale "structured editor" session.

Activity at some instant is defined (constrained) by

- the definition of the development process - methods
- the nature of the development task - products
- the operation of the supporting environment - tools
- the resource plan - resources
- the progress of other activities - dependencies

Each of the constraints can be thought of as sub-schemas. The intersection of these produces the context for activity. Where activity may be

- instantiation of a sub-schema
- refinement of their instantiation
- product development

Thus the IPSE 2.5 Machine consists of an abstract store full of development objects with embedded projection functions to give user views. These views are then the basis for user interactions via a structured editor which adds value through the interacting sub-schemas previously outlined.

Project Concepts

The project concepts have been studied under six sub-headings.

What concerns
- Management Support
- Design Support
- Formal Reasoning Support

How concerns
- User Interface
- Language
- Object Space

Management Support
- Process Description - Formal basis
- Reuse
- Process Scheduling
- User Access

Design Support
- Design - a process in which language terms (expressing designs and the qualities of designs) are constructed, verified and validated.
- Construction - a process in which language terms (expressing designs and the qualities of design) are constructed, verified and validated.
- Construction - a process in which language terms are
  - edited by input and output
  - re-used by storage and retrieval
  - transformed by functions (into other language terms)

Formal Reasoning Support
- the role of Formal Methods in IPSE 2.5 production
- requirements in terms of both education and tools
- IPSE 2.5 will facilitate "formal reasoning" but will not prescribe their usage
- Need for genericity
- Need for reasoning at different levels
  - deriving properties
  - verifying programs - using different paradigms
  - specification/design/verification
  - program transformation
  - constructive mathematics
- proofs of compilers etc.

Genericity implies parameterisation - hopeful candidates are
- the process model
- the formal method paradigm
- logics and theories

User Interface
- complete environment for project support
- aid users by providing them with roles defined by the process models(s)
- Support multiple contexts for interaction allowing concurrent activities across rules and within rules.

Language

- variety of roles within the project and hence the major unifying strategy
  - Specification Role for the project
  - Implementation Role (from generic deliverable)
  - Interaction Role
  - Planning language role
  - Application role to express designs and the qualities of designs

Research Goals

- Infrastructure
  - Data structures to support the manipulation of formal objects (e.g. Formulae)
  - "Active" Database requirements and implications
  - Object language for database manipulation
- Engineers tools
  - Making Formal Methods Usable
    - Handling Formal Texts
    - Usable interfaces to "proof editors"
- Demonstrators of improvement
  - VDM
  - Algebraic presentations
  - Trace assertions (CSP like)

Conclusion

By allowing the IPSE to be both an active object store and a means of defining intersecting and refined process models it is hoped that a flexible environment can be provided. This flexibility making possible both good project control and the use of different tools/methods in a single development cycle whilst preserving the creative freedom of individual designers. Further the integration of formalisms should enable the reuse of specification, designs and proofs as well as process models. This reuse leading not only to more efficient development cycles but also safer and more relevant products.

DISCUSSION

Professor Randell asked whether, when Professor Warboys said that the IPSE 2.5 solution would provide a complete infrastructure, did this mean complete in the sense of inextensible.

Professor Warboys replied that this was not the case and that IPSE 2.5 would be an open environment.

Professor Randell asked what was meant by saying that tools would co-exist in IPSE 2.5 and how was this reflected in how tools work.

Professor Warboys replied that in terms of how tools work it would be very much along the lines of the public tool interfaces that have been discussed in relation to ASPECT and ECLIPSE. Basically all one needs is some abstract data type sitting on the interface that is a representation of the conventions that the tool has established in the environment. However issues of standards and politics will probably influence the way things are done. It all depends on how the world moves on in the next year or so. If standards have not emerged by then, IPSE 2.5 will adopt whatever the consortium feels is most likely to succeed and this is quite likely to have a European flavour. (Of course by then Vic (Stenning) will be making millions out of his ISTAR and it will be obvious which interface to use.)

Professor Habermann referred to the diagram in which tools were first put in front of the database and later put underneath with associated projection functions. He suggested that the projection functions themselves represented tools and that, in fact, tools represent views of the database and that it is through a tool that you express your specific interest in how to look at the database at a particular time. Consequently tools don't belong where Professor Warboys put them but belong instead to the interface and give you the ability to operate on the database through the view you have of it.

Professor Warboys replied that, in the widest sense, Professor Habermann was right and that one runs into difficulties using words like tools, functions, abstract data types and so on. He had been using tools in the sense of a previous diagram. If one characterises tools as being compilers, editors and so on, then one of the main characteristics of current generation IPSES is that tools act as a way of filling up the database. The database remembers the constraints that are put upon it and provides the means of the next tool gaining access to it. The database remembers the constraints that are put upon it and provides the means of the next tool gaining access to it. The database can be regarded as a highly structured macro library. Professor Warboys referred to ICL's use of the CADES system for the last 15 or so years and said that, although it can handle enormous quantities of information and can handle multiple versions and other things, it presents a very static view of the development process. All you can see and manage are discrete events and you can't really use the database as part of process management.

The model needs to be inverted at least with respect to the tools so that the database becomes a more useful member of the project. True support of the process requires a much finer grain of integration. Paradoxically, the more thorough the description that is embedded in the database the more chance one has of producing a liberal environment for people.

Dr. Ritchie remarked that Professor Warboys had referred to including various things in IPSE 2.5 such as compilers, IPSE tools, databases, operating systems and proof systems and enquired whether IPSE 2.5 would begin anew or would existing products be used.

Professor Warboys said that IPSE 2.5 will need to use an existing database as, even though it is a large project, there isn't the money to write one. It should be possible to provide an abstraction which means that one doesn't need to worry too much about the particular database. The big implementation issue is whether to

1) use a conventional database, pick up a high level language and interface the language to the database.

2) pick up a language with a type structure which is very near to the specification system that is used - ML say - and accept that ML doesn't deal with concurrent access or persistence and has performance problems.

3) pick up something fairly crude and hack the thing out in C.

The intention is not to write a database system or operating system but to provide a rich language sub-system which sits on top of this which is basically a generator. What you get when you buy the IPSE is this generator which you then instantiate. This produces the first instantiation after which there are further instantiations which are a consequence of the product you are developing and the type of formalisms you are using at a lower level.

Dr. Ritchie asked whether there would be re-use of software.

Professor Warboys said that re-use was the true objective and that formalism is a way towards achieving this. Somebody needs to do the work and Alvey and the Corporation have been kind enough to give IPSE 2.5 a blank sheet of paper. Hopefully their trust will be rewarded.