them to occur again. In this predicament, structured modular top-down programming techniques appear likely to play an important role since they offer an opportunity to bring significant software reliability improvements.

There are certain aspects of a DCS that impose special requirements on the software methods and tools that can be used. The inherent loose coupling of processes that may operate asynchronously is one. The fact that ultra-large software systems can be assembled by connecting large ensembles of minicomputers is another. The currently evolving techniques of SP may have to be extended and adapted to cover such special requirements.

In the future, I expect to see increased use of DCS-like systems, such as systems of mini- and microprocessors, and nets of minis forming large and ultra-large (possibly too large) application systems.

In addition, there will be the need for advanced tools in producing software for DCS, including facilities to handle loose coupling and asynchronous operation.

Finally, I expect to see the use of process modeling structures and run time monitoring of the resultant system to check for system recoverability, system security, proper total operation, life time debugging aids, and performance data gathering.

The distributed programming environment offers both an area in which to demonstrate structured techniques and an area which demands the disciplines and modularity of structured approaches.

---

### The Need for Language Advances

**Alfred S. Liddle**

TRW Systems Group, Sunnyvale

We have heard a lot said in the past few days about some very advanced techniques and ideas relating to better ways of developing software. While I would not take anything away from what has been said, I would like to point out that there are some very serious practical problems in getting the great majority of practitioners in the industry to use and benefit from SP. If the people that I deal with are representative of the industry in general, there are a lot of engineers and programmers who do not yet have an appreciation of SP. If we do not consolidate the gains that are possible in the use of advanced languages and language facilities, we may lose the potential benefits of SP.

Understanding and communication are of the greatest importance in all phases of the software process, from the definition of requirements, through the design, implementation, test, operation and maintenance. One of the keys to understanding lies in better methods of expression, including improved languages carefully designed so that they can be visually more effective.

Language and tool development is essential to the success of SP. The time is past for simply using rules for the formation of SP constructs using existing programming languages. As we have seen, the state of the art today involves the uses of translators, SP dialects of existing programming languages, and the use of program design languages with or without computer aids.

Reasonable goals for the near future should include the following:

- The use of SP languages with generally-available compilers.
- Hopefully, some standardization of SP languages.
- Improved programming design languages and other design aids.

Some day, integrated requirements definition, design, and programming languages with ample compiler and support tools will be commonplace.

An integrated design and programming language is feasible today. Such a language might be of the following composition:

- It will contain both design and programming elements.
- There will be a common syntax for control structured for both design and programming aspects.
- It will be possible for the design language elements to stand alone without the programming language elements.
- When both design and programming language elements are present, the design language elements stand as comments which clarify the meaning of the programming language text.

Such an integrated design and programming language could be used as follows: during the design phase, a description of the design will be written using the design language elements. After design buy-off, the programming language elements will be added without discarding the design language text.

The use of such a language could produce the following major benefits:

- Design and implementation will be correlated.
- Documentation of the design will be kept current at all times.
- Redesign will be recognized as such because the design language text will necessarily be changed when deviations occur.
- If the language is used faithfully, it will eliminate uncommented source text.

If we take advantage of the opportunities to advance the state of the art in language development and software development tools, SP will have a favorable impact fully as great as that of the advent of higher order languages some 15 years ago.