System/Software Engineering-Enablement:
Technology-Challenge and Critical issues

Position Paper from the Steering Committee* of the Monterey Workshops

This position paper is to help identify the overall system/software engineering-enablement and technology-challenge critical issues for the foundations and the practice of a systematic cost-aware software and systems engineering capability. In our view these critical issues include

- The need for technology, based on theoretical/foundational advances (i.e., logic, formal language theory, automata, complexity theory, decision procedures etc.); however, the real value of any theoretical advances are expected to be realized only when they allow for engineering application in an integrated context; hence, research conducted devoid of context/practical-use, and possibly based on unrealistic assumptions, is unlikely to have as strong an impact on engineering capabilities as that which results in advances that allow for efficient integration of the "point-solution" capabilities.

- The need for technology (supporting an integrated process, methodology, modeling and description techniques, and tools/tool-suites) for the rapid elicitation of requirements, and

- their systematic refinement and validation;

- along with the requirements we need to consider the production of precise and "useful" requirements documentation; "useful" means that the documented requirements are directly usable in a semi-automatic fashion for the subsequent steps involved in software production, test, and evolution. As to documentation, more generally, the "overall system documentation" must adequately not only capture the system requirements but they must accurately represent system design; hence there is the need to consider multiple-points of view/representation.

Another of the technology needs therefore is the

- principled translation to specifications that allow for the coupled development of the system, by "domain experts" in the language of their choice close to the culture of their application domain.

Subsequent to the development of the system by domain experts, represented in a development language (or legacy code), we need

- support for the semi-automated translation to the language(s) of a given modeling language with a proper scientific foundation

- a design/analysis suite of tools (engineering platform) that supports that language.

Coupled to the design/analysis platform we need

- to create the design documentation as an artifact of the design process.

The above somewhat represents our current view of what needs to be covered in a "vision"; this view has to be made complete and comprehensive. It represents the totality of a holistic view of the system/software engineering process.

We believe that "embedded system" applications offer a near-term opportunity to eventually affect the generic engineering task. We further believe that we would be making a fundamental strategic mistake, by continuing to attempt to add perceived-embellishments to the "programming paradigm", without concern for the need for an integrated process that provides the obviously needed technology support for various aspects of the system/software engineering challenge.

We think recent history has shown that clever "point-solutions" (CASE tools, devoid of consideration of the need for integration etc.) are unlikely to be used to improve cost, implementation-schedule predictability, and system robustness to any significant degree. Also, the critical issues of design documentation and a platform that allows for principled evolution have hardly been addressed.