Evolving Software Reengineering Technology For the Emerging Innovative–Competitive Era

Phillip Q. Hwang
Defense Mapping Agency
8613 Lee Highway
Fairfax, VA 22031–2137

Evan Lock, Noah Prywes
Computer Command and Control Company
2300 Chestnut Street, Ste. 230
Philadelphia, PA 19103

ABSTRACT

This paper reports on a multi–tool commercial/military environment combining software Domain Analysis techniques with Reusable Software and Reengineering of Legacy Software. It is based on the development of a military version for the Department of Defense (DOD). The integrated tools in the military version are: Software Specification Assistant (SSA) and Software Reengineering Environment (SRE), developed by Computer Command and Control Company (CCCC) for Naval Surface Warfare Center (NSWC) and Joint Logistics Commanders (JLC), and the Advanced Research Project Agency (ARPA) STARS Software Engineering Environment (SEE) developed by Boeing for NAVAIR PMA 205. The paper describes transitioning these integrated tools to commercial use. There is a critical need for the transition for the following reasons: First, to date, 70% of programmers' time is applied to software maintenance. The work of these users has not been facilitated by existing tools. The addition of Software Reengineering will also facilitate software maintenance and upgrading. In fact, the integrated tools will support the entire software life cycle. Second, the integrated tools are essential to *Business Process Reengineering*, which seeks radical process innovations to achieve breakthrough results. Done well, process reengineering delivers extraordinary gains in process speed, productivity and profitability. Most important, it discovers new opportunities for products and services in collaboration with other organizations. Legacy computer software must be changed rapidly to support innovative business processes. The integrated tools will provide commercial organizations important competitive advantages. This, in turn, will increase employment by creating new business opportunities. Third, the integrated system will produce much higher quality software than use of the tools separately. The reason for this is that producing or upgrading software requires keen understanding of extremely complex applications, which is facilitated by the integrated tools. The radical savings in the time and cost associated with software, due to use of CASE tools that support combined Reuse of Software and Reengineering of Legacy Code, will add an important impetus to improving the automation of enterprises. This will be reflected in continuing operations, as well as in innovating new business processes. The proposed multi–tool software development is based on state of the art technology, which will be further advanced through the use of open systems for adding new tools and experience in their use.

1. Introduction And Summary

The paper describes a multi–tool Software Engineering Environment for commercial and military applications combining software Domain Analysis techniques with those of Reusable Software and Reengineering of Legacy Software.

The paper is based on the dual use of a version developed for the Department of Defense (DOD). The tools used in the military version must be modified for use in a commercial version. They are as follows: Software Specification Assistant (SSA) and Software Reengineering Environment (SRE) developed by Computer Command and Control Company (CCCC) under contracts with the Naval Surface Warfare Center (NSWC) and Joint Logistics Commanders (JLC), and the ARPA STARS Software Engineering Environment (SEE) developed by Boeing for NAVAIR PMA 205.

The *Software Reengineering Environment* (SRE) will facilitate:

a) Translation of legacy software from old languages (Fortran, Cobol, C) to modern languages (C++ and Ada) and a modern open–ended Operating System (OSF).
b) Software Understanding and Documentation through analysis of relations within the software and their graphical display.
c) Reorganization of software to obtain object oriented programs for concurrent network operations.

The *Software Engineering Environment* (SEE) will facilitate:

a) New software design.
b) Reuse of code associated with a Domain.
c) Code generation.

This system is much more important than just combining capabilities of tools. It is a change in Computer–Aided Software Engineering (CASE) technology that will radically improve overall business automation. It will drastically lower the cost and time to develop software, for the following reasons:

First, 70% of programmer time is applied to software maintenance [5]. The work of these users is not facilitated by existing CASE tools. The Software Reengineering tools proposed here will also facilitate the work of users engaged in software maintenance and upgrading. The integrated tools will have a much wider coverage of the software life cycle and a much wider audience.

Second, the integrated Software Engineering and Reengineering are essential to *business process reengineering*, which seeks radical process innovations to achieve breakthrough results [25], [42]. Done well, process reengineering delivers extraordinary gains in speed, productivity and profitability. More importantly, it discovers new opportunities for products and services offered in collaboration with other organizations. Rapid offering of the new products and services is essential, in order not to miss business opportunities. Innovation in legacy computer software has been recognized in studies of reengineering as the major obstacle in rapidly reengineering manufacturing processes. Software must change rapidly to support the core requirements of business process engineering. These requirements are [27], [24]:

a) Support of concurrent operations
b) Leveraging human resources
c) Sharing information
d) Collaborating with other organizations

The transitioned tools will offer commercial organizations the necessary means to attain strong, competitive advantages. This, in turn, will impact employment favorably by creating new business opportunities.

Third, the integrated tools offer a new, higher quality of software than the tools offer separately. The reason for this is that producing or upgrading software requires keen user understanding of extremely complex applications, which is facilitated by the Software Reengineering tool [35].

The transitioning from a military to commercial environment requires the following changes (only components of the military version that require changes for the commercial version are listed here):

i) Translation of commercial programming languages Fortran, Cobol, C, and C++ into C++. The present translations in the military version are from CMS–2 and Ada to Ada. C++ is selected as the modern programming language preferred by the commercial community.

ii) Integration of CCCC's SRE tools with those of Domain Engineering and Application (DEA) [40], developed by Boeing under ARPA/STARS sponsorship, and with that of the PTECH tool [38] for Object Oriented Software De-

sign. The integrated tools will be transitioned to operate under the Open Software Foundation (OSF) Operating System. This will make them portable to multiple vendors' platforms. The military version operates on Digital's Vaxstations under the VMS Operating System.

2. Overview Of The System

Figure 1 on the following page illustrates the technical approach. It shows the integrated operation of the Software Reengineering Environment (SRE) with Domain Engineering and Application (DEA) and PTECH, an object oriented CASE tool. The integrated operation covers the software life cycle for commercial use. Figure 1 shows the integration of three main tool groups, as follows:

Software Specification Assistant (SSA): It is shown at the top left of Figure 1. SSA facilitates the creation and updating of software requirements and specifications. In the military version it conforms with DOD–STD 2167A [19]. A similar set of standards will be selected for the commercial version. The inclusion of SSA reflects the importance of Software Specifications for an orderly software life cycle. SSA is an integrated set of information repositories and tools. SSA guides, instructs and informs staff in composing, updating and evaluating preliminary requirements and specifications. Typical users of SSA are a Development Manager, Software Support Activities, or Contractors. SSA allows a user to manage, query and update its extensive knowledgebase of information related to an application system. Staff may ask complex technical questions about the software and retrieve answers. Fragments of retrieved answers can be extracted for inclusion in updates to relevant new documents. SSA leads the inexperienced specifier in a "step–by–step" manner and provides traceability to the sources for updates in the documentation. This component is very flexible and open ended and does not require changes (beyond defining a commercial standard) to transition it to the commercial version.

SSA has been used in a demonstration project for the Tactical Air Mission Planning System (TAMPS) program at the Naval Air Warfare Center, Warminster, PA. It is planned to distribute the SSA tool to DOD agencies.

Software Engineering Environment (SEE) [23]: It is shown at the top right of Figure 1. SEE incorporates new software development technology for Reuse of software and for automatic program generation, following ARPA's Domain Specific Software Architecture (DSSA) Program [31]. The ARPA STARS SEE for NAVAIR PMA 205, used in the military version, includes two parts: i) Domain Engineering, to define the process of producing software for a class of related applications in a Domain, and ii) Application Engineering, for producing software for an application that belongs to the Domain [40]. It also contains PTECH [38] for object oriented design of software. The SEE can be transitioned to the commercial version without any changes. DEA is language independent and PTECH already generates object oriented programs in C++. The SEE incorporates meta tools for tool integration.

A specific Domain is comprised of software for a closely related family of applications. For example, the DSSA program has been experimenting with Domains in the areas of Guidance, Navigation and Control, Avionics, Command Control and Communications, etc. [8] Once a Domain Software Architecture [1] is developed, it can be used for generating applications. Business Domain architectures will be developed by respective users for their areas of interest, and will be populated by artifacts from legacy code. Domain Engineering contains a repository of Reuse Software artifacts and associated tools. The creator of a Domain Architecture is called the *Domain Engineer.* The DEA facilitates selection of Reuse Software and generation of software to create a specific application system. The user of the repository and of the tools is called the *Application Engineer* [40].

The Reuse Software is part of the DEA Repository (see Figure 1). It is organized as a hierarchy of software artifacts that follow the Domain Architecture. The DOD software also follows a standard structure of hierarchical software units (DOD–Standard 2167A). Each hierarchical software unit has a specification of its position in the architecture hierarchy, its capabilities, interfaces and dependencies on other software units. The graphs for documentation of the architecture are listed in Table 1. The capabilities of the hierarchical software units determine *commonality* and *variability* between hierarchical software units. It is possible to navigate through the Domain hierarchy tree by referring to capabilities and selecting hierarchical software units based on commonality and variability of their respective capabilities. Hierarchical software units may be parametrized and a code generation tool may be used to select parameters of generic software. Alternately, hierarchical units may be completely generated based on models of their functionality.

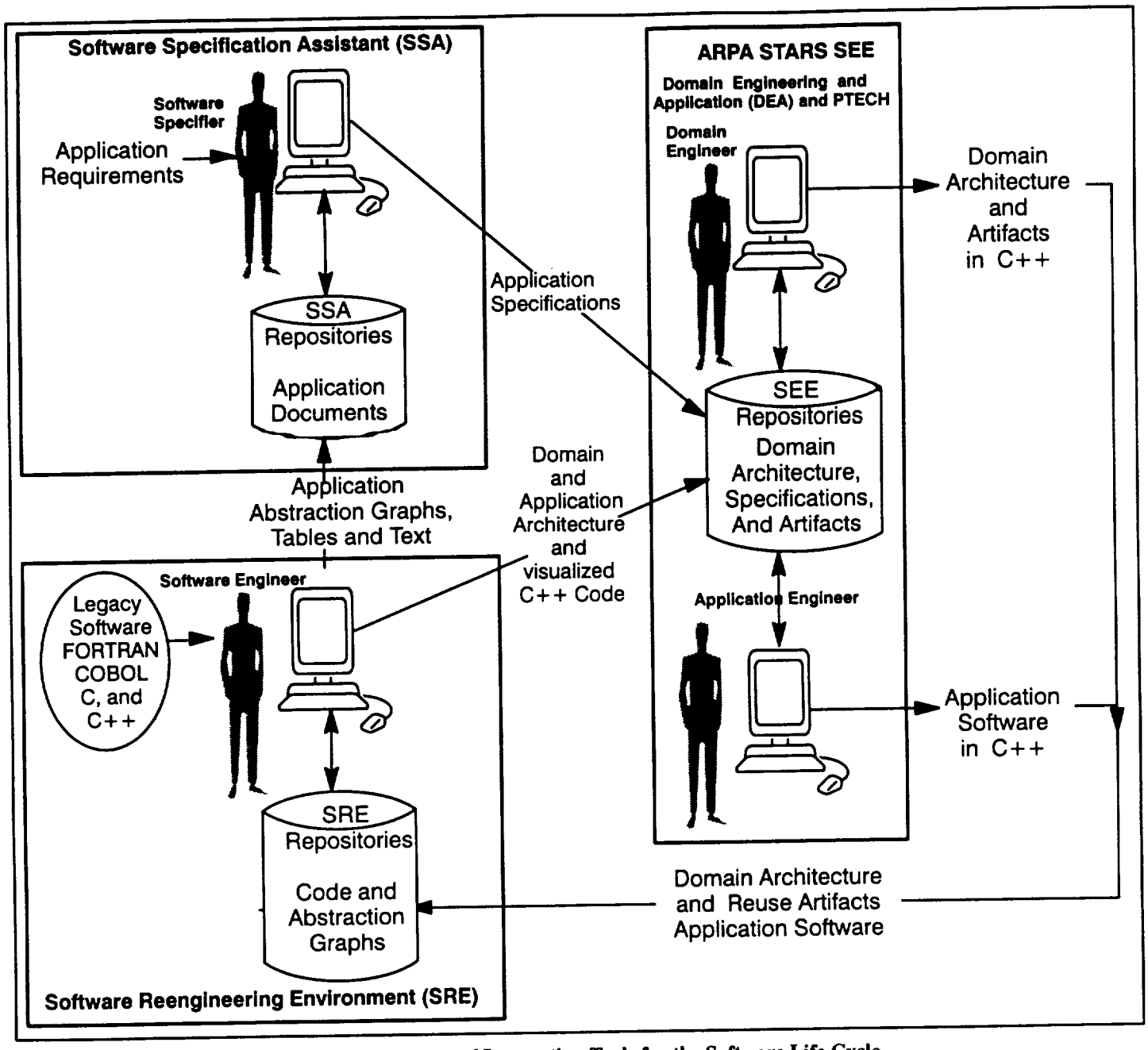

Figure 1: Overview of Integrating Tools for the Software Life Cycle.

A series of tools are available in the DEA for application modelling, unit testing and conversion to concurrent operation:

Software Reengineering Environment (SRE) [9]: SRE is shown at the bottom left of Figure 1. The SRE commercial version will consist of three main capabilities:

(i) *Software Understanding*: It consists of query and retrieval of graphic diagrams that illustrate the software from various perspectives. These diagrams are used to visualize specific aspects of the software. The diagrams are first divided into *in–the–large* and *in–the–small* diagrams. In–the–large diagrams visualize declarations of objects. In–the–small diagrams visualize execution statements within individual program units. The C++ program diagrams will be stored in a graphic form in the repository of a customized CASE system. A graphic query language is provided for ad–hoc browsing in the graphic repository, consisting of the documents in Table 1. These graphs show relations between high or low level hierarchical units. This facilitates the understanding of the software's architec-

ture as well as its code. Changes to the program can be made via the graphics used for visualization. This part of the military version can be used directly with no changes for the commercial version. The changes to C++ are all in the translation component (i).

(ii) *Software Abstraction and Documentation*: SRE partitions the software into multi–level hierarchical software units, in conformity with the standards for describing the software architecture. Software documents are then generated that describe the architecture in terms of the hierarchical software units, and describe the software in each unit from different perspectives shown in Table 1 below. This part of the military version can be used directly in the commercial version.

(iii) *Software Capture and its Transformation to C++ and OSF*: Fortran, Cobol, C or C++ source programs will be translated into C++ Entity–Relation–Attribute (ERA) diagrams. This representation is the main vehicle for graphic program analysis and visualization. Multiple passes are made over the source code to achieve 100% translation to object–oriented C++. Visualization is used for query, retrieval, understanding and generating documentation of programs. In a first pass, the SRE will translate the code, statement by statement, into pseudo–C++. Next, the generated pseudo–C++ programs will be transformed into the C++ programming paradigm in a series of passes. Each pass translates different aspects of the programming paradigm of the source language into the C++ programming paradigm (e.g., object declarations and classes). The Unix commands in legacy code will be replaced by OSF code having the same functionality. Other commercial subsystems (user interface and database) will be translated in this way as well. The military version uses Ada as the target language. It will have to add C++ as the target language.

(i)	Hierarchical Decomposition Diagrams	Showing decomposition of the overall software into hierarchical units.
(ii)	Flow Diagrams	Showing flow of data and control within and between hierarchical units.
(iii)	Interface Tables	Showing the structure of inputs and outputs of each hierarchical unit.
(iv)	Object/Use Diagrams	Showing (for hierarchical units) where types or generics are defined and where they are used.
(v)	Context Diagrams	Showing the library units and where they are used.
(vi)	Comments Text	Showing the comments in each hierarchical unit. They are assumed to contain information on the hierarchical unit's capabilities.

Table 1: Software Abstraction Documents Produced by SRE for Different Perspectives of the Software.

The integration of the SSA, SRE, DEA and PTECH involves primarily two interfaces, also shown in Figure 1. They are as follows:

Interface Between SRE and SSA: This interface is shown at the middle left of Figure 1 [37]. This interface provides a *reverse* process to produce information for the software requirements/specifications and other documentation from program code. SSA receives the documentation from SRE.

Interface Between SRE and SEE: The SRE provides DEA and PTECH with software documentation in the form of high level graphic–views of the architecture as well as detailed graphic views of algorithms. The SRE can process Legacy code as well as Reuse code from the DEA repository. It generates key parts of the specifications of each hierarchical software unit. The capabilities of each hierarchical software unit in the specifications are used for establishing commonality and variability among the domain architecture hierarchical software units.

Discussion of the Software Life Cycle Process using the integrated tools: The visualization provided by SRE facilitates human understanding. This is essential for successful employment of the entire Domain Specific Software Architecture (DSSA) concept. These SRE capabilities are needed for:

i) Analyzing and understanding the programs in the Reuse software library of Domain Engineering and in the target software produced by Domain Application.

ii) Analyzing, transforming and understanding existing software to expand the domain architecture with new artifacts. This capability includes the transformation of existing programs in other source languages to C++ .

DEA will use the SRE visualization graphs to compare the domain architecture's unit and assess commonality and variability of hierarchical software units. The SSA will be used to specify the capabilities of the architecture's hierarchical software units.

Typically the tools will be used iteratively until a desired new or upgraded application system is obtained. As an example, consider the following scenario. Assume for simplicity that totally new application software is desired. The preliminary requirements have been composed by the application's Program Manager. The platform to be used and its dynamics may be derived through modelling and simulation. The Specifier, with the aid of SSA, uses the preliminary requirements to compose the hierarchically structured specifications. The capabilities in the specifications are then communicated to the Application Engineer to select architecture units and generate the application software. If unable to do so, the Domain Engineer may be called to expand the scope of the domain. In either case, the SRE tool will be used to document and display the new domain and/or application software. Software Abstractions will be generated from the code. The Software Abstractions are next used by the Specifier, who employs SSA to update the specifications. The Domain Engineer will use the Software Abstractions to update the domain. The Application Engineer will use them to document the application software. This cycle may be repeated a number of times until satisfactory application software is realized.

3. Discussion Of The Transition

A military version of the system is partly operating (SRE) and partly in development (DEA). The transition into a commercial version will use the following technologies (The components of the military version which do not require alteration for the commercial version are not included):

i) Translation from older commercial languages (Fortran, Cobol, C) and from C++ to C++ and from Unix to OSF (instead of Ada in the military version).

ii) Use of meta–languages for assembling multiple tools. They must be transitioned from the Digital VAX workstations and VMS Operating System used in the military version, to the OSF Operating System, which is portable among different vendors' workstations.

3.1 Translation Technology

The methodology used in the present military version of SRE for translation of military source languages to Ada will be transitioned to translate Fortran, Cobol, C, and C++ into C++ and Unix to OSF. The graphic representation of programs in C++ is similar to the graphic representation of Ada. DEA is language independent of the target language and PTECH produces C++ code as well as Ada.

The processing in the SRE is shown in Figure 2 on the following page. SRE consists of four phases: *Parsing and Transformation* (P&T), *Analysis and Restructuring* (A&R), *Software Understanding* (SU) and *Software Abstraction* (SA). Only A&R will need to be revised.

The data is stored in three repositories: Software Reengineering (SR), Software Understanding (SU) and Software Abstraction (SA).

The input to P&T is a source language program. The output of P&T will be in Elementary Statement Language for C++ (ESL–C++). This is the graphic notation for visualizing C++ programs. An approach very similar to the one used in Ada is envisaged. Namely, the relations represented by edges in Ada can be retained in C++. These are: i) Edges between caller of a procedure and the procedure. ii) Edges between memory updating or referencing a variable and the variables' declaration. iii) Edges between message source and its destination. iv) Edges between a class and its instances. It is stored as a tree in the SR Repository in ASCII form, and may be modified by the users updating the programs.

A&R modifies and enhances the ESL–C++ tree in the SR Repository. It adds *tuples* that represent relations between statements. The results are restored in the SR repository. Visualization views are generated by A&R and stored in the SU Repository that is used for display by a CASE system.

The DECdesign CASE system is being used for the SU tool. The Graphic User Interface (GUI) used in SU is customized for software reengineering and understanding. SU supports graphic retrieval and generation of C++ code. Once the programs are generated in C++, they can be added to a DEA Reuse Repository.

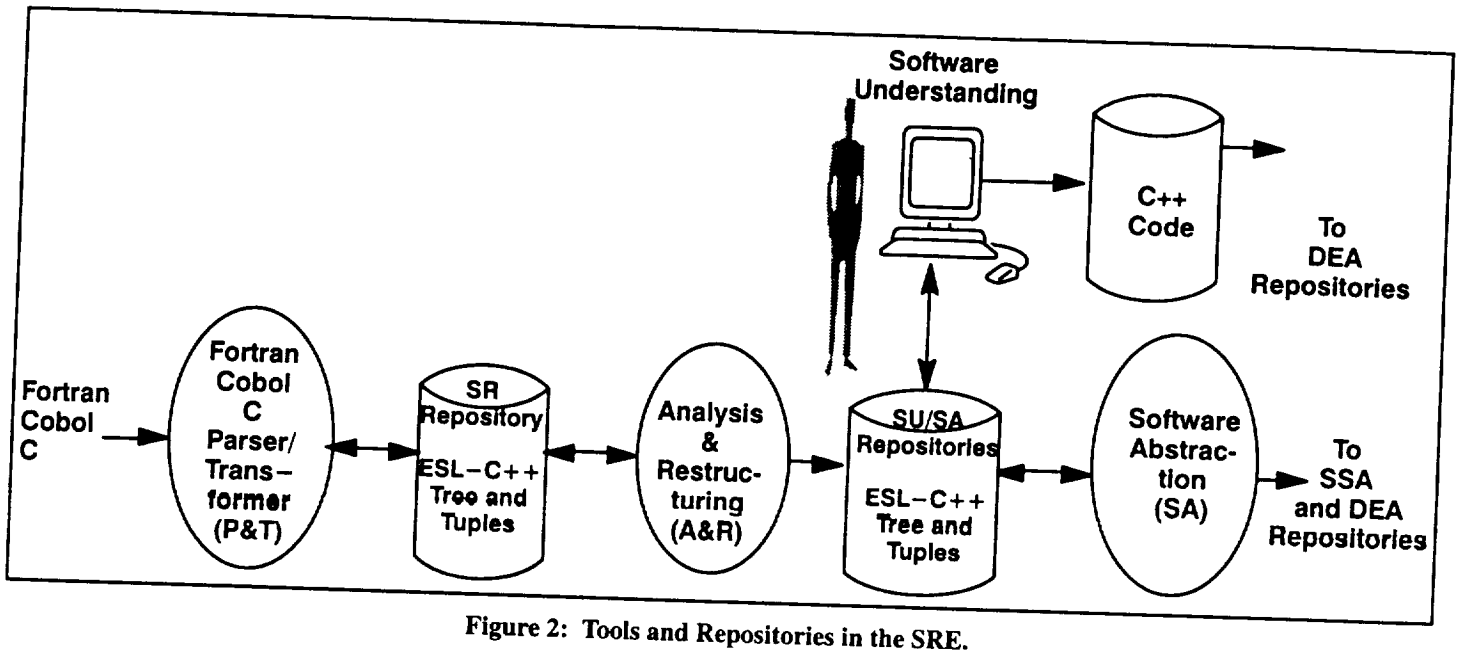

Figure 2: Tools and Repositories in the SRE.

SA produces diagrams, tables and text for the different perspectives of the software. These can be inserted in specifications of respective hierarchical software units. The SA Repository views are generated from the program visualization stored in the SU Repository. The SA Repository also uses formats acceptable by the CASE system used in SU.

3.2 Integrating Tools and Sharing Repositories

This section discusses systems used for Integration of Tools and Repositories. Three technologies are used in the ARPA STARS SEE. These systems are currently operational on Digital's Vaxstations under VMS. They need to be ported to OSF. They are as follows:

i) Process driven technology to combine operations of tools. Honeywell's AAA system is used [29].
ii) Tool and Repository integration technology. Digital's A Tool Integration Service (ATIS) [14] and HP–Softbench are used.
iii) Repository data dictionary technology for sharing data in a repository. Presently Digital's Common Data Dictionary/Repository (CDD/R) is used under VMS. It will be changed to use a vendor independent repository under OSF (Digital's Objectivity is a primary choice).

To incorporate new tools and repositories into the SEE, the schemas in ATIS or HP–Softbench and the Data Dictionary must be extended to include the following:

i) Declarations of new element types of the repositories that are managed by the tools.
ii) Declarations of the tools.
iii) Declarations of messages and methods that interface between user programs, the tools and the repository objects.

Finally, the processes that combine the use of multiple tools and a common user graphic interface must be redefined.

4. Conclusion

The key requirement in the emerging era will be to perform major software reengineering over a few days, at most. This will require extending the man–machine interaction of the human participants in the process. The approach described in Figure 1 can be expanded by adding support for multiple source and target programming languages. The proposed requirements are shown in Table 2. They consist of adding Cobol, Fortran, C and C++ as source program languages for the commercial world, and C++ as a target program language.

SRE: Software Reengineering	Accept Multiple Source Languages:	Military (CMS–2, Jovial, Ada) Commercial (Cobol, Fortran, C, C++)
	Generate Multiple Target Languages:	Ada, C++
	Reorganization:	For Object Orientation
	Understanding:	Graphical Query and Retrieval
	Abstraction:	Follows User Partitioning of Software Multiple Diagrams of Software Architecture: Decomposition, Flow, Objects and their instantiations
SSA: Software Specification	Very Large Repository of Documents (Text and Figures),	
	Very Rapid Search, Edit and Compose Facilities	
	Document Loading	
	Process Management	
	Portability Option	
SEE: Software Reuse and Generation	Domain Engineering Driven From Specifications	
	Domain Application Driven From Commonality/Variability	
	Structured Repository of Code and Specification	
	Code Generation	
	Concurrency Analysis	
	Simulation	

Table 2: Specification for Software Reengineering/Development For Emerging Era.

5. References

1. Agrawala, A., et al, "Domain–Specific Software Architectures for Intelligent Guidance, Navigation & Control," Proceedings of the DARPA Software Technology Conference 1992, Los Angeles, CA, April 1992.
2. Biggerstaff, T. J. "Design Recovery for Maintenance and Reuse," *IEEE Computer*, July 1989, pp. 36–49.
3. Biggerstaff, T. J., ed. *Software Reuse*, Addison Wesley 1987.
4. Black, E., "White Paper: ATIS, PCTE, CIS and Software Back Plane," Atherton Technology, Sunnyvale, CA 94089, 1991.
5. Bloom, P., "Case Market Analysis," Volpe, Welty and Co., San Francisco, CA 1990.
6. Chang, S., *Visual Languages and Visual Programming*, Plenum Press, 1990.
7. Chen, P., "The Entity–Relationship Model: Toward A Unified View of Data," ACM Trans. on Database Systems, May 1976.
8. Coglianese, L., et al, "An Avionics Domain–Specific Software Architecture Program," Proceedings of the DARPA Software Technology Conference 1992, Los Angeles, CA, April 1992.
9. Computer Command and Control Company, Final Report for Contract No. N60921–90–C–0298, "Software Intensive Systems Reverse Engineering," April, 1992.
10. Computer Command and Control Company, "Software Specification Assistant" Guides: Status Manager and Step–by–Step Guide, Document Manager Guide, Evaluation Guide and Installation Guide, Contract N00014–91–C–0160, December 1992.
11. Cvetanovic, Z., "The Effects of Problem Partitioning, Allocation and Granularity on the Performance of Multiple–Processor Systems," *IEEE Transactions on Computers*, Vol. C–36, No. 4, April 1987.
12. DeMarco, T. "Structured Analysis and System Specification," Prentice–Hall, Inc., Englewood Cliffs, NJ, 1987.
13. Digital Equipment Corporation, "Guide To DECdesign," Maynard, MA, May 1990.
14. Digital Equipment Corporation, "A Tool Integration Standard," ANSI X3H 4, Information Resource Dictionary System, ATIS, DEC Maynard, MA, February 1990.
15. Digital Equipment Corporation, "DECdesign: User's Guide," AA–PABRB–TE, Maynard, MA, May 1991.
16. Digital Equipment Corporation, "Digital: CDD/Repository: Using CDD/Repository on VMS Systems," DEC, Maynard, MA, Part No. AA–P51KA–TE, Oct. 1991.
17. Digital Equipment Corporation, "Digital: CDD/Repository: Architecture Manual," DEC, Maynard, MA, Part No. AA–PJ1JA–TE, Oct. 1991.
18. Digital Equipment Corporation, "Digital: CDD/Repository: Callable Interface Manual," DEC, Maynard, MA, Part No. AA–PJ1LA–TE, Oct. 1991.
19. DOD, DOD–STD–2167A: "Defense System Software Development," September 1988.
20. DOD, "Military Standard Software Development and Documentation (Draft)," DOD Harmonization Working Group, December 1992.
21. Donnelly, C. and R. Stallman, "BISON, The YACC Compatible Parser Generator" Free Software Foundation, Cambridge, MA 02134, 1990.
22. Evans, A. and Butler, K. J., "Descriptive Intermediate Attributed Notation for Ada Reference Manual," TL–83–4, Tartan Labs, Pittsburgh, PA 1983.
23. Foreman, J. "STARS: State of the Program," STARS '92 Conference, 1992, pp. 20–41.
24. Goldman, S.L. and Roger N. Nagel, "Management, Technology, and Agility: the Emergence of a New Era in Manufacturing," International Journal Technology Management, Vol. 8, No. 1/2, pp. 18–38, 1993.
25. Hammer, M. & J. Champy, "The Reengineering Corporation, A Manifesto For Business Revolution," HarperCollins, 1993.
26. Hayes–Roth, F., et. al, "Domain–Specific Software Architectures: Distributed Intelligent Control and Management (DICAM) Applications and Development Support Environment," Proceedings of the DARPA Software Technology Conference 1992, Los Angeles, CA April 1992.
27. Iacocca Institute, "Benchmark Agility," Lehigh University, Bethlehem, PA, 1992.
28. Kimball, J. and A. Srivastava, "AAA 1.0: An Experimental Notation For Engineering Processes," Honeywell Systems and Research Centers, Technical Report CS–R92–013, July 1992.

29. Kimball, J. and K. Thelen "AAA Example: Structured Technical Review," Honeywell Systems and Research Center, June 1993.

30. Lock, E., N. Prywes, and S. Andrews, "Case For Development And Re-engineering Of Real-time Distributed Applications," Fourth International Conference, Software Engineering and Its Applications, Toulouse, France, December 9-13, 1991.

31. Mettala, E. and M. Graham, "The Domain Specific Software Architecture Program," Proceedings of the DARPA Software Technology Conference 1992, Los Angeles, CA, April 1992.

32. Meyers, B., "The State of Art in Visual Programming and Program Visualization," Technical Report CMU-CS-88-114, Carnegie Mellon University, February 1988.

33. Naval Sea Systems Command, PMS 412, User Handbook for CMS-2 Compiler, NAVSEA 0967-LP-598-8020, 30 March 1990.

34. Naval Sea Systems Command, PMS 412, Program Performance Specification for CMS-2 Compiler, NAVSEA 0967-LP-598-9020 30, March 1990.

35. Nielsen, J., "Noncommand User Interfaces," *CACM* (36), No. 4, April 1993, pp. 83-99.

36. Prywes, N., E. Lock, and X. Ge, "Automatic Abstraction of Real-Time Software and Re-Implementation in Ada," Proc of Tri-Ada '91, October 21-25, 1991.

37. Prywes N., I. Lee "Integration of Software Specification, Reuse and Reengineering," Computer Command and Control Company, Philadelphia, PA, 19103, June 1993.

38. PTECH Design and PTECH Code, Release 3.5, Tool User's Guide," Associative Design Technology, Mar, 1992.

39. Selby, R., A.A. Porter, D.C. Schmidt, J. Barney, "Metric Driven Analysis and Feddback Systems for Enabling Empirically Guided Software Development," Proc. 13th International Conference on Software Engineering, 1991.

40. SPC "Domain Engineering Guidebook," Technical Report SPC-92019-CMC, Software Productivity Consortium, December 1992.

41. Srivasta, A. "Strategy For Process Programming Using Agents, Activities And Artifacts (AAA)," Proc. of STARS '92 Conference, December 1992.

42. Stewart, T., "Reengineering: The Hot New Managing Tool," Fortune, pp 41-48, August 23, 1993.

43. Sun-Joo Shin, "Valid Reasoning and Visual Representation." PhD thesis, Stanford University, August 1991.

44. Ward, P. and S. Mellor, Structured Design, Yourdon Press, Englewood Cliffs, NJ, 1972, 2nd ed.

45. Wileden, J. A.L. Wolf, W.R. Rosenblatt and P.L. Tarr. "Specification Level Interoperability," *CACM*, 34(5) May 1991.