

AD-A253 327



Technical Report

CMU/SEI-92-TR-9

ESD-TR-92-009

2



Carnegie-Mellon University

Software Engineering Institute

**Parallels in Computer-Aided Design
Framework and Software Development
Environment Efforts**

Susan A. Dart

May 1992

DTIC
ELECTE
JUL 30 1992
S B D

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

92-20293



92 7 27 209

The following statement of assurance is more than a statement required to comply with the federal law. This is a sincere statement by the university to assure that all people are included in the diversity which makes Carnegie Mellon an exciting place. Carnegie Mellon wishes to include people without regard to race, color, national origin, sex, handicap, religion, creed, ancestry, belief, age, veteran status or sexual orientation.

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admissions and employment on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders. In addition, Carnegie Mellon does not discriminate in admissions and employment on the basis of religion, creed, ancestry, belief, age, veteran status or sexual orientation in violation of any federal, state, or local laws or executive orders. Inquiries concerning application of this policy should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Technical Report

CMU/SEI-92-TR-9

ESD-TR-92-009

May 1992

**Parallels in Computer-Aided Design Framework
and Software Development Environment Efforts**



Susan A. Dart

Environments Project

Approved for public release.
Distribution unlimited.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

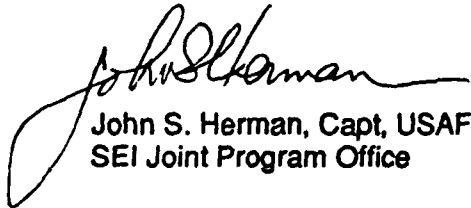
SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER



John S. Herman, Capt, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.

This report was funded by the U.S. Department of Defense.

Copyright © 1992 by Carnegie Mellon University.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Table of Contents

1	Introduction	1
2	State of Automated CM in the SDE Community	3
3	State of Environment Technology in the SDE Community7	
4	State of Automated CM in the CAD Community	9
5	State of Environment Technology in the CAD Community11	
6	Comparisons Between the Communities	13
7	Conclusion	17
8	Acknowledgments	19
	References	21

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

List of Figures

Figure 6-1	Sequence of Events in Understanding Concepts	13
Figure 6-2	Differences Between CAD and SDE Notions	14

Parallels in Computer-Aided Design Framework and Software Development Environment Efforts

Abstract: This paper is an attempt to raise awareness about the similarities between the efforts of the software development environment (SDE) community and the electronic computer-aided design (CAD) framework community. Apparently, SDE and CAD engineers are not aware of what is happening in each other's fields, yet cross-pollination of efforts would assist progress. Both communities are addressing the same problems of providing configuration management (CM), tool integration, and process management support in their environment. Each community can benefit from the other since both have similar needs and have found, and are finding, similar solutions. It is particularly useful to consider collaborative efforts as both communities are evolving towards standardization.

1 Introduction

The SDE community is focusing on the problems of CM, third-party tool integration, and process modelling, as is the CAD community. Commercially, this can be seen from the plethora of third-party tools for CM, computer-aided software engineering (CASE) tools, third-party design kits, and tool and design management frameworks. Also, books about software engineering environments [Long 91] and electronic design automation frameworks have been appearing [Rammig 91], as well as articles about standardization efforts in reference models for environments [Brown92], tool integration [Zarrella 90], and agreement upon a universal design automation framework, such as the CAD Framework Initiative (CFI) [Malasky 91]. Thus, from a software engineering point of view, it is clear that both communities are currently tackling similar solutions for supporting automated CM and framework technology that enables integration of third-party tools and support for process (task) management. In particular, CAD framework vendors have to address the needs of both communities, since they need a SDE in order to develop their CAD framework for their customers.

The following sections give an overview of the key concepts and challenges for CM support and environments in both communities. A discussion follows summarizing the similarities and differences between the communities, and the conclusion suggests how each community could benefit from the other.

2 State of Automated CM in the SDE Community

In the software engineering community, the standard definition of CM involves four requirements: identification, control, status accounting, and audit and review. The paper by Dart extracts 15 CM concepts from existing SDEs and tools [Dart 91]. These concepts indicate that what is implemented in CM systems really goes beyond that standard definition into support for manufacture, process modeling and teamwork. These 15 concepts are:

1. **Repository:** captures CM information and stores versions of files as immutable objects, as in RCS [Tichy 82].
2. **Distributed component:** allows distributed users to have access to a centralized repository; to them, the CM facilities seem to span the network of heterogeneous workstations, as in Sherpa DMS [Deitz 91].
3. **Context management:** captures in a domain-specific manner the working context for the user, thereby eliminating the need for users to remember how they got to a particular working status and what all the data items, relationships, and tools are in that context, as in Powerframe [Johnson 89].
4. **Contract:** represents a formal plan for, and a record of, a unit of work on a configuration item, as in ISTAR [Graham 88].
5. **Change request:** assists in driving the process of change to configurations and keeping an audit trail of the changes, as in LIFESPAN [Whitgift 91].
6. **Life-cycle model:** represents the process of developing and maintaining configurations based on a predefined life cycle, as in CCC [Softtool 87].
7. **Change set:** represents a logical change to a product and a means of creating any version of a configuration that is not necessarily dependent on the latest version of that configuration, as in ADC [SMDS 89].
8. **System modelling:** abstracts the notion of a configuration from an instance of it, and, by fully describing the configuration, enables tools to maintain the integrity of the configuration, as in Jasmine [Marzullo 86].
9. **Subsystem:** provides a means to limit the effect of changes and recompilation, and a means for the environment to check the validity of combining parts of a system, as in Rational [Feiler 88].
10. **Object pool:** optimizes the need for regenerating objects and maximizes the sharing of derived objects, as in DSEE [Leblang 85].
11. **Attribution:** permits the description of a system at a higher level of abstraction via its characteristics rather than in terms of a lengthy composition list of files, as in Adele [Estublier 85].
12. **Consistency maintenance:** enables the environment to identify any inconsistencies and to preserve consistencies in creating and reusing configurations, as in CMA [Ploedereder 89].

13. **Workspace:** provides isolation of work between engineers and distinguishes between a global, long-term repository for immutable objects and a private, shorter-term repository for mutable objects, as in shape [Mahler 88].
14. **Transparent view:** gives a viewing mechanism for a configuration from the public repository with protection against unauthorized access, as in SMS [Cohen 88].
15. **Transaction:** synchronizes and coordinates teams of engineers changing the same or different parts of a system, as in NSE [Feiler 90].

(Note that although the examples for the concepts distributed component and context management were taken from CAD frameworks, variations on these concepts exist in some SDEs.)

No single CM system provides all the above concepts, although a survey of CM systems indicates that CM systems are approaching a common set of concepts in the SDE arena. Yet there is a lack of common terminology; no well-understood vocabulary exists for software engineers to discuss CM. Different terms are meant to be the same concept while similar terms mean different concepts. Implementations of the same concept can have different semantics. For instance, in various implementations of version graphs, varying semantics for merging branches and different purposes for branches can be found [Feiler 91]. The lack of a common terminology inhibits progress as there is no standard specification or model for CM.

At the Software Engineering Institute (SEI), we are attempting to address the need for a standard CM model by unifying the above concepts into a CM services model. This model will be a conceptual framework for a set of well-defined services. "Service" is meant as a particular CM functionality. "Well-defined" means that a service is defined in such a way that its semantics, interface, and other properties are understood enough to be included in framework reference models and to be implemented, albeit with possibly different mechanisms.

The services in the model take into account the software engineering marketplace's need to apportion and distribute functionality. That is, CASE tools and environments provide parts of CM solutions and customers buy these pieces as building blocks for their solution (since no single CM system is a panacea). The CM solution is generally spread across tools. The services model, in essence, is intended to provide plug in/plug out, "black box" capabilities so that over time, services could be replaced with new ones. Examples of services are repository, system modelling, version control, derived object management, change management, version differentiation, and so on. The model addresses upward compatibility, since no single environment will emerge as the most popular given that customers have their existing solution and no off-the-shelf solution exactly matches the homegrown solution.

The future needs for CM can be characterized by five aspects: technology, process orientation, management, standardization, and politics. New technology is needed, such as support for distributed CM capabilities, interoperability between CM systems, customizable CM systems, and a global perspective on CM repositories. The problems of tool integration with CM capabilities must be addressed. Better CM process definition and implementation of

processes are required. More support for management in decisions about CM systems, such as the buy versus build decision, and in evaluating CM systems is crucial. Reference models for environments are being standardized, as are the CM services involved with them. Governmental contracts will eventually require contractors to have a certain level of CM support in their environment, so the ramifications of this requirement need to be addressed.

As CM is examined more closely in relation to software engineering in toto, several questions emerge, with the most interesting technical one asking "What part of the software engineering problem is CM?" It is not clear whether general software engineering issues, such as team support, process support, and tool integration are areas that are part of CM, yet CM vendors generally need to address these issues when giving the customer an off-the-shelf solution. This question also arises under the guise of "Where do we put CM?" when committees are trying to decide upon the architecture for their environment framework reference models.

At the moment, a long-term CM solution that supports long-lived, changing software for large organizations with diverse software applications will not be found in a single CM tool, nor in a single environment. Other aspects of software engineering need to be addressed in unison, such as process modelling, software architectures, team support, and tool integration. CM has an impact on these aspects and vice versa. Thus, CM is a keystone to the software engineering problem and should not be viewed in isolation from other software engineering problems and solutions. Good CM support in an environment will greatly enhance its usability, whereas inadequate CM support makes an environment useless.

3 State of Environment Technology in the SDE Community

An SDE is an environment that assists software engineers in developing and maintaining software. There has been, and continues to be, a lot of work on environments in the SDE community. The book on SDEs edited by Long [Long 91] and the symposium proceedings of ACM SIGSOFT/SIGPLAN [Henderson 88] provide examples of work being done. Two major areas of concern are tool integration for third-party CASE tools, and software process modelling for describing and automating the steps involved in developing software. International conferences and workshops are devoted to these topics [Dowson 91], [Fuggetta 91].

One way to begin examining environment technology is by categorization of environments, as done by Dart, Ellison, Feiler and Habermann, which suggests that SDEs, from a user's perspective, fall into four categories: language-centered, structure-oriented, toolkit, and method-based environments [Dart 87]. Other categorizations have been published presenting varying perspectives [Perry 91], [Penedo 88]. The various categorizations show that different criteria can be chosen for categorizing and that environments generally don't fit exactly into one category.

A major concern these days is that of tool integration: what it means and how the notion of "openness" for third-party tool integration can be supported. The paper on tool integration technologies by Brown, Feiler and Wallnau [Brown 91] divides the tool integration solution into **integrated project support environments (IPSEs)** such as those based on the international standard, the portable common tool environment (PCTE) [Boudier 88], and into **tool coalitions** that are based on vendor-supplied tool integration facilities at the source code level. PCTE is a set of standardized, public tool interfaces to basic environment mechanisms which support a common interface to a set of cooperating, data sharing tools. It is intended to be a solution to a variety of integration and portability problems encountered when running multiple tools from multiple vendors on multiple platforms. PCTE essentially provides a centralized database of schemas. The IPSEs are intended to provide an all encompassing infrastructure that enables tool integration and provides features for life-cycle activities (such as specification, design, and coding) and cross-life-cycle activities (such as configuration management, project management, and documentation production). The CASE coalitions provide a piecemeal, esoteric solution where vendors sign a pact and integrate their tools to form a tightly coupled set of tools to suit particular life-cycle activities, such as design and coding. Overall, the IPSE and the tool coalition approaches provide benefits but require tradeoffs.

Another approach to the tool integration problem is to try to avoid it by providing engineers with a **meta-tool**, a tool that enables them to rapidly develop their own, highly customized set of CASE tools. (Examples include the Virtual Software Factory [Bloor 90] and the Tool Builder's Kit [Alderson 91].) These serve as a repository for the logical model of the environment as well as for some elements of the physical implementation. The tools are developed using the pro-

vided formalism and run under the provided runtime environment and with the user-defined process model.

For process modelling, the design of formalisms for describing processes and the development of mechanisms for implementing them are active research topics. Environments built on top of PCTE, such as the EAST, Enterprise II and ALF environments [Oquendo 90] try to provide flexible process modelling capabilities, but must address difficulties such as the need for common services (for complex object passing and sharing) and active mechanisms (for notification and triggers).

At the SEI, we are bringing together terminology and concepts that assist in understanding the issues for tool integration and process modelling which can be incorporated into an environment framework [Brown 91].

4 State of Automated CM in the CAD Community

Katz gives an overview of the basic versioning capabilities for CAD frameworks in which he distinguishes between seven classes of mechanisms [Katz 90]. These include:

1. **Organization of space of versions:** represents a repository with a version tree of objects.
2. **Dynamic configurations and dereferencing:** refer to static and dynamic binding of versions to configurations and consistency checking.
3. **Hierarchical compositions across versions:** determine the structure and composition of configurations.
4. **Alternative groupings of versions:** refer to partitioning of a system and regrouping of objects based on certain properties.
5. **Distinctions of instances versus definitions:** make a clear distinction between the instance and definition of an object and that relationship.
6. **Change notification and propagation:** relate to time-based notification and acknowledgment of events, and propagation of changes.
7. **Workspace organization:** provides a means for multiple designers to share objects and ways of providing multiple repositories (private, group/project, and public/archive).

Katz concludes that the versioning technology for the CAD community has variations on a theme; many of the mechanisms are similar. Thus, there is a need for a unified model of mechanisms and a unified terminology. But the real challenge of the future is to create a single, tailorable framework for CAD encompassing all of the versioning capabilities above. Perhaps the work of the CFI will address this challenge.

From a software engineering perspective, it appears that, in general, commercial CAD frameworks generally offer a version control facility along with a notification facility. Most of the other classes of mechanisms tend to be in research CAD frameworks and fall into the database research arena. The CM problems evolve around the need to deal with the existence of multiple versions of chips that are to be used in different chip configurations as well as "competitive tools", i.e., allowing for the coexistence of multiple versions of tools where any version can be used at any time.

5 State of Environment Technology in the CAD Community

A CAD framework is an environment that assists hardware engineers in designing, drawing and simulating circuits. It is intended to provide a broad spectrum of services relating to data, task, process and methodology management [Harrison 90]. There is a significant amount of work on environments in the CAD community. The book on electronic design frameworks by Rammig and Waxman provides examples of tool integration strategies and frameworks [Rammig 91], and the proceedings of the Design Automation Conference provides a wide range of issues in the CAD community [ACM 91]. When examining CAD frameworks, it appears that CAD framework vendors sell two kinds of products. One is a tool framework that encompasses tightly integrated tool sets with inter-tool communication, a common representational database, and version control; these optimize the performance of combinations of vendor's products via source code integration. The second is a design-management framework that assists in the process of designing and testing circuits by providing data management capabilities such as CM, data flow, and event triggers; tool run management for sequencing and executing tools; and tool encapsulation for enabling an "open" architecture for integrating third-party tools.

The goal of the CFI efforts is to provide a standardized framework for CAD tools that serves as a general, common software infrastructure for efficiently building, maintaining and configuring open, integrated CAD environments. This involves developing guidelines for enabling cooperation of CAD tools. Such work is progressing within the CFI.

6 Comparisons Between the Communities

There are many similarities in the technological solutions that each community uses, and some differences that make for varying levels of complexity in the solutions.

Similarities in Concepts and Technology

It is clear that both communities would like a common terminology for CM concepts. CM mechanisms appear very similar, and many are variations on a theme. For tool integration, the need for a standard framework is common to both communities, as are the natures of the problems and solutions. The challenges of the future—tool integration, data management, process implementation, and managing multiple methodologies—appear the same for both communities. From a software engineering perspective, the CFI work attempts to define an infrastructure that is similar to many efforts in the SDE community for defining reference models for IPSEs.

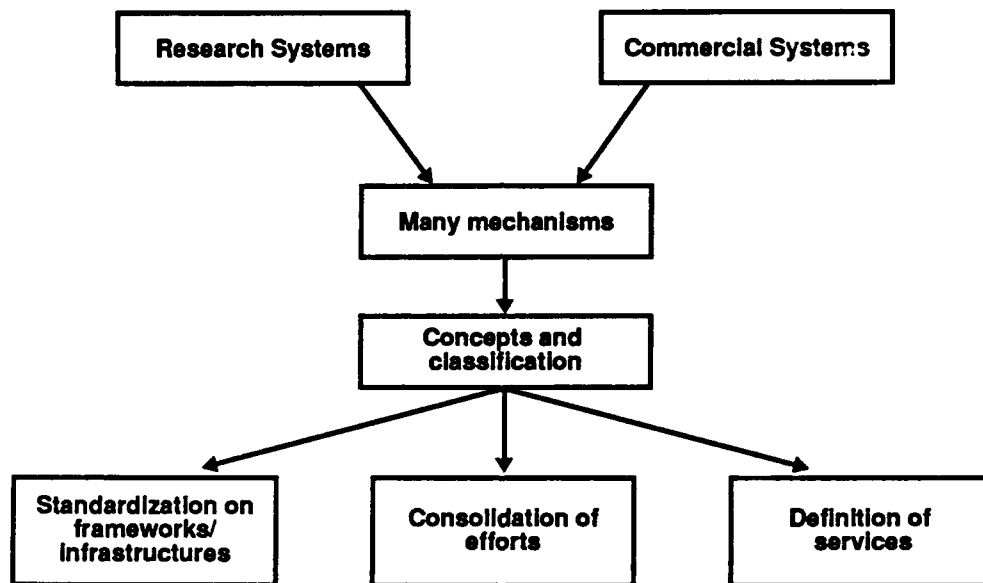


Figure 6-1 Sequence of Events In Understanding Concepts

Scanning the literature published in the SDE and CAD communities shows that the work on environments and CM is progressing in a similar fashion. Figure 6-1 shows a simplified and generalized view of the process of the evolution towards an understanding of the concepts and mechanisms. Researchers experiment with ideas and create prototype systems. Commercial vendors create production-quality tools, sometimes using research ideas but scaling them up to industrial strength, and sometimes coming up with innovative mechanisms. Surveying all the research and commercial systems gleans a plethora of mechanisms used. Technology analysts then start to extract concepts from the mechanisms and begin to classify them and develop a vocabulary enabling engineers to discuss the technology. From here begins the work of standardization of mechanisms, concepts and frameworks, along with the definition of classes of services that provide certain functionality. Also, different groups tend to realize that

there are similarities in efforts, and will combine forces to consolidate work efforts. This happens in cycles, for each new round of technology.

A good example is with CM in the SDE community. In the past, most CM systems were home-grown, that is, built in-house by the organization since it was impossible to buy a CM system. Researchers started experimenting with developing third-party tools, and start-up companies began to develop them from their experience in industry and from research. Today we have a flourishing CM tool industry with many tools and environments providing CM capabilities. As customers and industry analysts attempt to evaluate all these tools and understand the many mechanisms available, it is necessary to find a means for understanding such. Places such as the SEI examine technology and extract the concepts and classify the tools which helps the customers understand the state of the technology. Upon further examination of the plethora of tools many similarities are seen. As industry sees trends of variations on themes, it becomes clear that consolidation of efforts and standardization are possible. This then evolves into a standardization of common CM services, which is beginning today.

The SDE community is at the level of standardization, definition of services, and consolidation of efforts for tool integration (with PCTE at least) and CM. It appears that the CAD community is also at this level with its CFI efforts.

Differences Between the Communities

The communities appear to differ on the degree of domain specificity that they are dealing with. The differences relate to how well we understand objects, how objects are used, the way objects are created, the breadth of the problem domain, the complexity of transformations, and the understanding of the design process. These are highlighted in Figure 6-2:

CAD Community	SDE Community
Known objects (pin, unit, etc.)	Unknown objects (file, procedure, etc.)
Uses reusable objects	Builds new objects
Many views (layout, etc.)	Many instances (compilation, etc.)
Domain specific	Operating-system based
Many tool transformations	Compiler-based transformations
Known design process	Ad hoc design process

Figure 6-2 Differences Between CAD and SDE Notions

In the CAD community, engineers deal with known objects, objects that have some semantics associated with them such as pins or logic units. To create new objects, engineers can reuse existing parts of chip designs. For a description of an object, many different views (such as netlist and logic schemata) can be seen. CAD environments are specifically suited to the domain of hardware design and therefore the hardware designer. There are many tools in a CAD framework that transform a description into a different view. Thus, a CAD framework is multi-

tool oriented. The CAD community uses the standard language VHDL as a formalism for describing hardware designs. The design process of creating an integrated circuit is fairly well understood, and the cost and time for developing a particular version is known. There are many tools that aid the design process in ways such as analyzing and simulating the circuit.

In the SDE community, the lowest granularity of a well understood object is that of a file or procedure, and there generally are no real semantics associated with these. To create new objects, programmers tend to create new code by editing and compiling rather than reusing existing code, thus creating instances of object code. SDEs are based on a specific operating system in most cases and are intended to be fairly generic and suited to any kind of software engineer in any domain. Most of the transformations done are carried out via the compiler since SDE's are very single tool oriented, and that tool is the compiler. No single, standard programming/design language exists for software engineers to describe their applications. The software design process is still an ad-hoc one and very few tools exist to aid the designer in going from one level of abstraction to the next.

While differences can be found between the ways the CAD and SDE communities do their work and the kinds of tools that are available to them, the biggest gap is really that neither community seems to be aware of what the other is doing. (Obviously this is a generalization since some people do span the gap.) The future looks promising though, since several hardware-software workshops are being held and various groups such as the architecture group of the CFI and a PCTE group are willing to look closely and seriously at the similarities and differences between the communities. For the SEI, we are interested in finding "the best practice" so that we can help improve the state of the practice. To find the best practice, it is necessary to be involved with as many communities as possible.

7 Conclusion

SDE and CAD CM technology have a large degree of overlap. In both communities, there is a need for a unified CM model to assist in defining a CM vocabulary as well as a CM functionality within the frameworks. There is a possibility that defining unified CM services across hardware and software frameworks would have the benefit of common terminology, at least in the CM arena, and would thus aid progress. It would be beneficial to avoid duplication of effort.

From a software engineering viewpoint, CAD and SDE environments are progressing in similar directions. Both are tackling the third-party tool integration problem either via the same mechanisms, or by consensus through common models and interfaces. Research is continuing on understanding process models and implementing them. The CAD frameworks are intended to suit the particular domain of developing integrated circuits. SDE frameworks are intended to suit all of the software engineering community. Domain specificity should enable faster progress.

What can be learned by the software engineering community from the CAD efforts? One answer concerns the user interface model and the design process. CAD frameworks present a more intuitively obvious user interface that takes into account the domain of the user through use of graphics, and by hiding as much of the underlying implementation of the tools as possible. Also, they provide automated support for the design process such as tools that help take a chip design from one level of abstraction to the next via analyzing, synthesizing and simulating.

What can be learned by the CAD community from SDE work? One answer is teamwork, multi-user support, and data modelling in the sense of consistency analysis. SDEs provide mechanisms to allow multiple users to work in a synchronized and coordinated manner on the same or parallel pieces of code, such as with transactions. Also, mechanisms are provided for describing the structure and characteristics of data and for identifying and maintaining consistency between the data, such as checking for version skews and compatibility between source code interfaces and bodies.

It would be beneficial for each community to be aware of each other's work. There are many-fold advantages to broadening the concepts and mechanisms of each community. Perhaps it is possible that a unified framework suiting SDE and CAD needs could become a reality one day. Such a framework would offer the best of both worlds. The communities can cooperate by reading each other's papers and attending each other's conferences and workshops.

There are many problems confronting SDEs. No doubt the CAD community is, or will have to confront these problems too. Thus, any joint efforts will be the first steps towards cross fertilization of ideas. The future problems confronting SDEs relate to technology (such as addressing new CM requirements), process (such as better process guidance and tailoring), standardization (such as for CM services), and management (such as commitment of resources and assistance in helping management make the "buy vs. build" decision). In particular, the

technological problems concern interoperability between third-party CM tools, better support for large software changes enabling a global perspective on CM repositories, tailoring CM tools, distributed CM support, and framework infrastructures such as PCTE and object management systems. At the SEI, we are developing a CM services model to suit the future requirements of SDEs, particularly frameworks based on the client/server model technology. Our aim is to ensure that the CM services model is equally applicable to the CAD community.

8 Acknowledgments

I would like to thank the following people for their useful advice and discussion: Jay Brockman, Walter Heimerdinger, Fred Long, Peter Feiler, Alan Brown, the attendees at the Third International Workshop on Electronic Design Automation Frameworks, and the SEI technical writers.

References

- [ACM 91] ACM, IEEE Computer Society DATC. *Proceedings of the 28th ACM/IEEE Design Automation Conference June 17-21 San Francisco California*. ACM, IEEE, 1991.
- [Alderson 91] Alderson, A. "Meta-CASE Tool Technology", pp. 81-90. *Software Development Environments and CASE Technology*. Vol. 509, *Lecture Notes in Computer Science*. New York, New York: Springer-Verlag, June, 1991.
- [Bloor 90] Bloor, R. "The Software Tools' Software Tool", *DECUSER* (April 1990): pp. 53(2).
- [Boudier 88] Boudier, G., Gallo, T., Minot, R., and Thomas, I. "An Overview of PCTE and PCTE+", pp. 248-257. *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Engineering Environments*. Boston, Massachusetts: ACM, December 1988.
- [Brown 92] Brown, A. W. and Feiler, P. H. *The Conceptual Basis for a Project Support Environment Reference Model* (CMU/SEI-92-TR-2). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, January 1992.
- [Brown 91] Brown, A. W., Feiler, P. H., and Wallnau, K. C. *Understanding Integration in a Software Development Environment* (CMU/SEI-91-TR-31, ADA248119). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, December 1991.
- [Cohen 88] Cohen, E., Soni, D., Gleucker, R., Hasling, W., Schwanke, R., and Wagner, M. "Version Management in Gypsy", pp. 210-215. *Proceedings of the ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, New York, New York: ACM, November 1988.
- [Dart 91] Dart, S. A. "Concepts in Configuration Management Systems", pp. 1-18. *Proceedings of the 3rd International Workshop on Software Configuration Management*, New York, New York: ACM, June 1991.
- [Dart 87] Dart, S. A., Ellison, R., Feiler, P., and Habermann, N. "Software Development Environments", *IEEE Computer* 20, 11 (November 1987): pp. 18-28.
- [Deitz 91] Deitz, D. "Pulling the Data Together". *Mechanical Engineering*. 112, 2 (February 1990): pp. 56-57.
- [Dowson 91] Dowson, M., ed. *Proceedings of the First International Conference on the Software Process, Redondo Beach CA. Oct. 21-22, 1991*. Los Alamitos, California: IEEE Computer Society Press, 1991.

- [Estublier 85] Estublier, J. "A Configuration Manager: The Adele Data Base of Programs", pp. 140-147. *Workshop on Software Engineering Environments for Programming-in-the-Large*. Waltham, Massachusetts: GTE Laboratories, June 1985.
- [Feiler 91] Feiler, P. H. *Configuration Management Models in Commercial Environments* (CMU/SEI-91-TR-7, ADA235782). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, March 1991.
- [Feiler 88] Feiler, P. H., Dart, S., and Downey, G. *Evaluation of the Rational Environment* (CMU/SEI-88-TR-15, ADA198934). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, July 1988.
- [Feiler 90] Feiler, P. H. and Downey, G. *Transaction-Oriented Configuration Management* (CMU/SEI-90-TR-23, ADA235510). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, November 1990.
- [Fuggetta 91] Fuggetta, A., Conradi, R., and Ambriola, V., eds. *Proceedings of the First European Workshop on Software Process Modeling. Milan, Italy. 30-31 May 1991*. Milan, Italy: AICA, 1991.
- [Graham 88] Graham, M. and Miller, D. *ISTAR Evaluation* (CMU/SEI-88-TR-3, ADA201345). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, July 1988.
- [Harrison 90] Harrison, D., Newton, A., Spickelmier, R., and Barnes, T. "Electronic CAD Frameworks". *Proceedings of the IEEE* 78, 2 (February 1990): pp. 393-417.
- [Henderson 88] Henderson, P., ed. *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*. Boston, Massachusetts: ACM Press, 1988.
- [Johnson 89] Johnson, W. "Bringing Design Management to the Open Environment". *High Performance Systems* 10, 6 (June 1989): pp. 66-70.
- [Katz 90] Katz, R. "Toward a Unified Framework for Version Modeling in Engineering Databases", *ACM Computing Surveys* 22, 4 (December 1990): pp. 375-408.
- [Leblang 85] Leblang, D. and McLean, G. "Configuration Management for Large-Scale Software Development Efforts", pp. 122-127. *GTE Workshop on Software Engineering Environments for Programming in the Large*. Waltham, Massachusetts: GTE Laboratories, June 1985.
- [Long 91] Long, F., ed. *Proceedings of the Software Engineering Environments. University College of Wales, UK. 25-27 March 1991. Volume 3*. London, England: Ellis Horwood, 1991.

- [Mahler 88] Mahler, A. and Lampen, A. "shape — A Software Configuration Management Tool", pp. 228-243. *Proceedings of the International Workshop on Software Version and Configuration Control*. Stuttgart, Germany: B. G. Teubner, January 1988.
- [Malasky 91] Malasky, Bruce, et al. *Framework Architecture Reference*. Austin, Texas: CAD Framework, 1991.
- [Marzullo 86] Marzullo, K. and Wiebe, D. "Jasmine: A Software System Modelling Facility", pp. 121-130. *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*. New York: ACM, December 1986.
- [Oquendo 90] Oquendo, F. "Building Object and Process-Centered Software Environments on the PCTE Public Tool Interface". *Proceedings of the Sixth International Software Process Workshop, Haikido Japan*, Los Alamitos, California: IEEE Computer Society Press, October 1990.
- [Penedo 88] Penedo, M. H. and Riddle, W. E. "Guest Editors' Introduction: Software Engineering Environment Architectures". *IEEE Transactions on Software Engineering* 14, 6 (June 1988): pp. 689-696.
- [Perry 91] Perry, D. E. and Kaiser, G. E. "Models of Software Development Environments". *IEEE Transactions on Software Engineering* 17, 3 (March 1991): pp. 283-295.
- [Ploedereder 89] Ploedereder, E. and Fergany, A. "A Configuration Management Assistant", pp. 5-14. *Proceedings of the Second International Workshop on Software Configuration Management*. New York, New York: ACM, October 1989.
- [Rammig 91] Rammig, F. J. and Waxman, R. *Electronic Design Automation Frameworks*. IFIP, North Holland: Elsevier Science Publishers, 1991.
- [Softool 87] Softool. *CCC: Change and Configuration Control Environment. A Functional Overview*. Goleta, California: Softool Corporation, 1987.
- [SMDS 89] Software Maintenance & Development Systems, Inc. *Aide-De-Camp Software Management System, Product Overview*. Concord, Massachusetts: Software Maintenance & Development Systems, Inc., 1989.
- [Tichy 82] Tichy, W. "Design, Implementation and Evaluation of a Revision Control System", pp. 58-67. *6th International Conference on Software Engineering Tokyo*. Long Beach, California: IEEE Computer Society, September 1982.
- [Whitgift 91] Whitgift, D. *Methods and Tools For Software Configuration Management*. Chichester, New York: J. Wiley, July 1991.

[Zarrella 90] Zarrella, P. F. *CASE Tool Integration and Standardization* (CMU/SEI-90-TR-14, ADA235640). Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie Mellon University, December 1990.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS None		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-92-TR-9			5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-92-009		
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute		6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office		
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213			7b. ADDRESS (City, State and ZIP Code) ESD/AVS Hanscom Air Force Base, MA 01731		
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office		8b. OFFICE SYMBOL (if applicable) ESD/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003		
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213			10. SOURCE OF FUNDING NOS.		
			PROGRAM ELEMENT NO 63756E	PROJECT NO. N/A	TASK NO. N/A
11. TITLE (Include Security Classification) Parallels in Computer-Aided Design Framework and Software Development Environment Efforts					
12. PERSONAL AUTHOR(S) Susan A. Dart					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM TO		14. DATE OF REPORT (Yr, Mo., Day) May 1992	
15. PAGE COUNT 30 pp.					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) configuration management computer-aided design framework software development environment		
FIELD	GROUP	SUB. GR.			
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This paper is an attempt to raise awareness about the similarities between the efforts of the software development environment (SDE) community and the electronic computer-aided design (CAD) framework community. Apparently, SDE and CAD engineers are not aware of what is happening in each other's fields, yet cross-fertilization of efforts would assist progress. Both communities are addressing the same problems of providing configuration management (CM), tool integration, and process management support in their environment. Each community can benefit from the other since both have similar needs and have found, and are finding, similar solutions. It is particularly useful to consider collaborative efforts as both communities are evolving towards standardization. (please turn over)					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPTDTC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution		
22a. NAME OF RESPONSIBLE INDIVIDUAL John S. Herman, Capt, USAF			22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631		22c. OFFICE SYMBOL ESD/AVS (SEI)

ABSTRACT —continued from page one, block 19