Reference

NBS
PUBLICATIONS

NAT'L INST. OF STAND & TECH
A11106 034722

NBSIR 85-3113

Annotated Bibliography of Recent Papers on Software Engineering Environments

Raymond C. Houghton, Jr.

4105 Livingstone Place
Durham, NC 27707

Edited by

Dolores R. Wallace

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Programming Sciences and Technology
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

February 1985

Issued April 1985

U.S. DEPARTMENT OF COMMERCE
NAL BUREAU OF STANDARDS

QC
100
.U56
85-3113
1985

NATIONAL BUREAU
OF STANDARDS
LIBRARY

NBSIR 85-3113

ANNOTATED BIBLIOGRAPHY OF RECENT PAPERS ON SOFTWARE ENGINEERING ENVIRONMENTS

Raymond C. Houghton, Jr.

4105 Livingstone Place
Durham, NC 27707

Edited by

Dolores R. Wallace

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Programming Sciences and Technology
Institute for Computer Sciences and Technology
Gaithersburg, MD 20899

February 1985

Issued April 1985

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

TABLE OF CONTENTS

PREFACE

This document reports on the contents of fifty-five recent papers on software engineering environments. Several of these papers present an overview of software engineering environments. Other papers discuss issues to be considered in building software engineering environments. The remaining papers describe general software engineering environments, system development environments, and programming environments.

ACKNOWLEDGEMENT

This report was funded by the National Bureau of Standard's ICST under the U.S. Department of Commerce Purchase Order 415632.

ANNOTATED REFERENCES for SOFTWARE ENGINEERING ENVIRONMENTS

[Alfo81] M. W. Alford and C. G. Davis. Experience with the Software Development System, Software Engineering Environments, H. Hunke, Editor, North-Holland, 1981.

Paper presents a methodology and supporting environment for developing very large, complex, real-time systems. The environment emphasizes the discovery of errors early in the development process. The methodology consists of four major tasks: (1) Data Processing Systems Engineering (DPSE) - translate systems objectives into a consistent, complete set of subsystem functional and performance requirements (uses techniques based on verification graphs, petri nets, finite state machines, and graph models of decomposition for expressing requirements), (2) Software Requirements Engineering Methodology (SREM) - express functional and performance requirements as a graph model in Requirements Statement Language (RSL) and analyze with the Requirements Engineering Validation System (REVS), (3) process design engineering - translate requirements into a process design language, verify design, and evolve the design into code, (4) verification and validation - perform at all stages.

[Barn82] P. Barnard, N. Hammond, A. MacLean, and J. Morton. Learning and Remembering Interactive Commands, Proceedings of Human Factors in Computer Systems, Washington, D. C. Chapter, ACM, Gaithersburg, MD, 1982.

Paper presents an experiment to determine how the choice of command names influences interactive performance of users by measuring access to an on-line help system. The authors propose that help assistance is a relatively passive cognitive strategy for learning and that it leads to less efficient operation retention if commands have general names.

[Baye81] M. Bayer, et. al. Software Development in the CDL2 Laboratory, Software Engineering Environments, H. Hunke, Editor, North-Holland, 1981.

Paper presents a programming environment that supports the CDL2 programming language. The components of the system include a command interpreter, a dedicated editor/with formatting and cross-reference, a program database (underlies all tools), a file system (underlies the program database), a local and global analyzer, an intermodule interface checker, a local and global optimizer, a segment builder, an abstract code-generator, and a concrete code generator. The paper includes an in-depth discussion of the system architecture.

[Boeh84] Boehm, Barry W. et. al. A Software Development Environment for Improving Productivity, Computer, Vol. 17, No. 6, June 1984.

Paper presents the background and status of TRW's Software Productivity System (SPS). The background includes discussion of a 1980 software productivity study and corporate motivating factors (increased demand of software, limited supply of software engineers, rising software expectations, reduced hardware costs). Based on an internal assessment, an external assessment, and a quantitative assessment, the productivity study recommends that TRW initiate a long range effort to develop a software development environment. In the short term, the study recommends the development of a prototype environment. The architecture and components of the prototype that was developed (called SPS-1) include: the work environment (improved office conditions), the hardware (a network of VAX's, LSI-11/23's, terminals, and communication equipment), a master project database (composed of a hierarchical file system, a source code control system, and a relational database), general utilities (menu, screen editor, forms package, date/time, report writer), office automation and project support (tool catalog, mail system, text editor/formatter, calendar, forms management, interoffice correspondence package), and software development tools (requirements traceability tool, SREM [Alfo81], program design langauge, Fortran-77 analyzer). Experience with the use of the prototype shows a definite improvement in productivity and also that immediate access to a good set of tools has the highest payoff.

[Bran81] Martha A. Branstad, W. Richards Adrion, and John C. Cherniavsky. A View of Software Development Support Systems, Proceedings of National Electronics Conference , Chicago, IL, October, 1981.

Paper presents views and examples of support environments and proposes four classes of support levels. The views include the toolbox approach, the VHLL (very high level language) approach, and the software development support system approach (an integrated system with an underlying database). Examples include PWB/Unix, Interlisp, Howden's environments [Howd82], and the Ada Programming Support Environment (APSE). The proposed four classes of support levels are: D1 (what most operating systems provide), D2 (the capabilities of D1 plus integration with an underlying database), D3 (D2 plus support for the entire life cycle), and D4 (D3 plus features that are current research topics). Extensions of each level are also proposed for critical applications.

[Bran81a] Martha A. Branstad and W. Richards Adrion, Editors. NBS Programming Environment Workshop Report. National Bureau of Standards, NBS SP 500-78, June 1981.

Report presents the results of the Programming Environment Workshop, Rancho Santa Fe, CA, April 1980. The goals of the workshop were to assess the current technology, indicate needed standards for tools, develop guidance for near term environment development, and determine future research directions. The workshop attendees were divided into four groups: (1) contemporary software development environments, (2) software environment research - the next five years (3) advanced development support systems, and (4) high level language programming environments. The results of each of the four groups is reported by their respective leaders, W. Howden [Howd82], L. Osterweil [Oste81], T. Standish, and M. Zelkowitz.

[Buxt80] J. Buxton. Requirements for Ada Programming Support Environments: STONEMAN. U.S. Department of Defense, Washington, DC, February 1980.

Report lays out the requirements for Ada environments. The general requirements include support for the entire lifecycle, integration of tools, and exploitation of modern hardware. The more specific requirements include: the Ada Programming Support Environment (APSE)/Minimal APSE (MAPSE)/Kernal APSE (KAPSE) model, database support, and tool support at the different APSE/MAPSE/KAPSE levels. The APSE/MAPSE/KAPSE model is a four level model where level 0 is the host level, level 1 (KAPSE) is a standard interface to level 0 that supports database interactions, communications, and run-time, level 2 (MAPSE) is a minimal tool set (editor, translator, linker, debugger, configuration manager), and level 3 (APSE) is a set of tools for full support (life cycle, documentation, and management support).

[Buxt80a] J. Buxton. An Informal Bibliography on Programming Support Environments, ACM Sigplan Notices, Vol. 15, No. 12, December 1980.

Bibliography with brief notes and commentary on 40 papers that deal primarily with architectures of programming support environments. Included with the bibliography are short summaries of the following proposed and existing programming support environments: Cheatham's PDS Model, Cooprider's Thesis, CADES, C-MESA, Softech's MSEF, Stenning's Foundation System Model, and Tichy's Model.

[Camp84] Roy H. Campbell and Peter A. Kirslis. The SAGA Project: A System for Software Development, Proceedings of the ACM Sigsoft/Sigplan Software Engineering Symposium on Practical Software Development Environments, Gaithersburg, MD, April 1984.

Authors discuss the SAGA Project and its current status. The project proposes to develop a lifecycle support

environment for small to medium size projects. At the heart of the system is a proposed language-oriented editor generator that can synthesize a language-oriented editor for each life cycle language (i.e., requirements language, design language, implementation language, etc.). The current system is Unix based and includes a prototype for a language-oriented editor (implementation language), prototype version control components, and a production documentation tool.

[Cowe83] Wayne R. Cowell, and Leon J. Osterweil, The Toolpack/IST Programming Environment, Proceedings of SoftFair, IEEE Computer Society Press, NY, (IEEE Order No. 83CH1919-0), July 1983.

The paper discusses a portable, Fortran-oriented programming environment. The architecture of the environment includes a command interpreter, tool fragments (commands may invoke several tool fragments), and a virtual file system (files created by tools can be recreated by tools). The tools in the environment include: data flow analyzer, program instrumenter and debugger, documentation generation aid, program formatter, syntax-controlled editor, macro processor, text formatter, program structurer, Ratfor processor, portability checker, program transformer, and various file-handling utilities.

[Cox83] Brad J. Cox. The Message/Object Programming Model, Proceedings of SoftFair, IEEE Computer Society Press, NY (IEEE Order No. 83CH1919-0), July 1983.

A tutorial on object-oriented programming, the paper discusses the operator/operand model and the message/object model. It then presents a case study developed on Smalltalk-80.

[Deli84] Norman M. Delisle, David E. Menicosy, and Mayer D. Schwartz. Viewing a Programming Environment as a Single Tool, Proceedings of the ACM Sigsoft/Sigplan Software Engineering Symposium on Practical Software Development Environments, ACM, New York, (ACM Order No. 548840), April 1984.

Paper presents an interactive programming environment called Magpie. The user of Magpie deals with two states, the status of the source code and the status of execution. The environment features overlapping windows, a mouse pointing device, pop-up menus, a browsing capability, language-directed editing, incremental compiling, and debugging capabilities.

[Fair80] Richard E. Fairley. Ada Debugging and Testing Support Environments, Proceedings of the ACM-Sigplan Symposium on the Ada Programming Language, ACM, New York,December 1980

(see SIGPLAN Notices, Vol. 15, No.11, November 1980).

A review of the requirements specified in [Buxt80] and a discussion of the issues associated with them. Analysis considerations, source level support and debugging, KAPSE considerations, and data abstractions are covered.

[Feil81] Peter H. Feiler and Raul Medina-Mora. An Incremental Programming Environment, Proceedings of the 5th International Conference on Software Engineering, IEEE Computer Society Press, NY, (IEEE Order No. 81CH1627-9), March 1981.

Paper reviews the support required by programming environments and the traditional method for providing this support (i.e., editing, translation, linking, loading, and debugging). It then presents the Incremental Programming Environment which features syntax-directed editing, common representation, incremental program translation, and language oriented debugging. This is followed by a discussion of design and implementation issues for such an environment.

[FIPS99] Guideline: A Framework for the Evaluation and Comparison of Software Development Tools, National Bureau of Standards FIPS PUB 99, March 1983.

Guideline presents a framework (a taxonomy) for identifying, discussing, classifying, evaluating, and comparing software development tools and environments. The taxonomy includes almost 100 features that are presented in a hierarchy. At the top level, features are divided into input/output categories or function categories. The functions include transformation, static analysis, dynamic analysis, and management. The appendix includes a set of event sequences for the acquisition of tools.

[Fisc84] C. N. Fischer, et. al. The Poe Language-Based Editor Project, Proceedings of the ACM Sigsoft/Sigplan Software Engineering Symposium on Practical Software Development Environments, ACM, New York, April 1984.

Paper presents an overview of POE, a Pascal programming environment, and presents some of the technical issues associated with the development of the environment.

[Gutz81] Steve Gutz, Anthony I. Wasserman, and Michael J. Spier. Personal Development Systems for the Professional Programmer, Computer, Vol. 14, No. 4, April 1981.

This paper reviews the problems with existing development environments, proposes a programmer's personal machine, and examines the advantages and disadvantages of such a machine. The proposed programmer's personal machine

consists of: (1) an intelligent terminal with 1 Meg local storage, CPU and address space equivalent to a 32-bit mini, graphics capability, (2) hard disk (40 Megs) and floppy disk, (3) networking capability (with other PPBS's), (4) audio input/output, (5) pointing device (mouse, tablet, or light pen), and (6) capability to add more memory and other devices (e.g. a quality printer). The potential advantages of such a machine include constant response time, a comprehensive set of tools, less dependence on a single machine, integration of software development and office automation. The potential disadvantages include tool expense, training expense, communications, device dependence.

[Guya84] Jacques Guyard and Jean-Pierre Jacquot. MAIDAY: An Environment for Guided Programming with a Definitional Language, Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society, NY,(IEEE Order No. 84CH2011-5), March 1984.

Paper discusses an environment under development that is oriented around an object-oriented language and an algorithm design methodology. The environment enforces the methodology through a set of control functions. The user views a development session through a set of windows that present the development level, messages, the object being defined, objects remaining to be defined, the stored algorithm, and the current command.

[Hall80] Dennis E. Hall, Deborah K. Scherrer, and Joseph S. Sventek. A Virtual Operating System, Communications of the ACM, Vol. 23, No. 9, September 1980.

Paper presents a Unix-like environment that can be implemented on top of a vendor-supplied operating system to make in-effect a virtual operating system. The environment consists of four layers: (1) the vendor-supplied operating system (the innermost layer), (2) the virtual machine (consisting of primitives such as opening and closing files, reading and writing to files), (3) utilities (a set of tools written in portable Fortran including Kernighan and Plauger's Software Tools), and (4) an integrated command interface.

[Haus81] Hans-Ludwig Hausen and Monika Mullerburg. Conspectus of Software Engineering Environments, Proceedings of the 5th International Conference on Software Engineering, IEEE Computer Society NY, (IEEE Order No. 81CH1627-9), March 1981.

A paper which discusses the issues associated with the coverage of software engineering environments. The issues include support for full life cycle development, quality assurance, product control, management, specific

applications, specific methodologies, and representation schemes. Also discussed are issues related to the integration of tools and motivations for developing environments. The appendix defines the criteria that must be met for a system to be considered an environment. These include: a software engineering orientation, the use of at least one recognized scientific concept, applicability to more than one life cycle phase, and some level of tool integration. The authors present short summaries of environments that they feel meet these criteria. They include AIDES, APSE, ARGUS, ASSET, CADES, CDL2-Lab, COSY, DREAM, Gandalf, Gypsy, HDM, ISES, PWB/Unix, SDEM/SDSS, REVS, and SEF.

[Houg82] Raymond C. Houghton, Jr. A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE). National Bureau of Standards, NBSIR 82-2625, December 1982.

A review of the APSE [Buxt80], the ALS [Wolf81], and the AIE (the Navy's Ada Integrated Environment) based on [FIPS99]. The taxonomy includes a comparison of features in the areas of management, static analysis, dynamic analysis, transformation, and input/output. A set of underlying tool primitives is defined that support these features.

[Houg84] Raymond C. Houghton, Jr. Online Help Systems: A Conspectus, Communications of the ACM, Vol. 27, No. 2, February 1984.

Paper discusses online assistance that is provided by various types of environments. It includes a discussion of the types of assistance and the issues associated with the development of online help systems.

[Howd82] William E. Howden. Contemporary Software Development Environments, Communications of the ACM, Vol. 25, No. 5, May 1982.

Paper proposes four levels of tool support that could be provided by software engineering environments. For each level, the type of project, the estimated cost, and the support provided is detailed. For example, environment I has an estimated cost of $35K and is for medium-sized projects, while environment IV has an estimated cost of $3M and is for large scale projects. Requirements, design, coding, verification, and management tools and techniques are presented for each environment level.

[Huff81] Karen E. Huff. A Database Model for Effective Configuration Management in the Programming Environment, Proceedings of the 5th International Conference on Software Engineering. IEEE Computer Society,NY, (IEEE Order No. 81CH1627-9). March 1981.

Paper addresses configuration mangement issues (i.e., configuration identification and configuration control) in a software engineering environment and presents a model for effectively handling them.

[Hunk81] H. Hunke, Editor, Software Engineering Environments, North-Holland, 1981.

Book contains the proceedings of a symposium held at Lahnstein, Federal Republic of Germany, June 1980. Some of the papers include [Snow81], [Baye81], [Ridd81], [Alfo81], and [Mats81]. Papers related to some in the book are [Tayl84], [Stuc83], [Buxt80], and [Kern81]. Other papers discuss issues and tools related to software engineering environments including functional aspects of environments, computer aided design, support for concurrent and distributed systems, human factors, description languages, productivity, formal verification, performance, system decomposition, and version control. The book concludes with a bibliography by Hausen, Mullerburg, and Riddle that contains more than 350 citations from 1968 to 1980.

[Kern81] Brian W. Kernighan and John R. Mashey. The Unix Programming Environment, Computer, Vol. 14, No. 4, April 1981.

A paper which extols the benefits of the Unix programming environment. It reviews the underlying interface, i.e., the hierarchical file system and the seven primitive functions: open, create, read, write, seek, close, and unlink. It reviews the overlying interface (the user interface), i.e., available tools, input/output redirection, and various operators available to the user. It then discusses how a user can avoid programming by building a shell procedure of simpler tools available on the system. Finally, the attributes of the system are discussed, e.g., support for medium size projects, support primarily for the latter stages of software development, loose integration of tools and facilities, general support for all applications.

[Kuo83] Jeremy Kuo, Jay Ramanathan, Dilip Soni, and Markku Suni. An Adaptable Software Environment to Support Methodologies, Proceedings of SoftFair, IEEE Computer Society Presss,NY (IEEE Order No. 83CH1919-0), July 1983.

Paper describes an environment that can be tuned to support different software development methodologies. The control mechanism is based on the gathering of project data through a forms-based interface. The forms are defined at the start of development.

[Lebl84] David B. Leblang and Robert P. Chase, Jr. Computer-Aided Software Engineering in a Distributed Workstation

Environment, Proceedings of the ACM Sigsoft/Sigplan Software Engineering Symposium on Practical Software Development Environments, ACM, New York, April 1984.

Paper discusses an Apollo-based distributed software engineering environment. Because instances of the system can be running at various nodes in the environment, the system consists primarily of managers that keep track of what is going on. The managers include: a history manager (source code control), a configuration manager (version control), a task manager (tracks interrelationships among development products), monitor manager (watches user-defined dependencies), and an advice manager (tracks general project information).

[Love83] Tom Love. Experiences with Smalltalk-80 for Application Development, Proceedings of SoftFair, IEEE Computer Society Press, (IEEE Order No. 83CH1919-0), July 1983.

Paper extols the benefits of the single-user, single-language environment called Smalltalk-80. An example is presented of a graphics program that was developed using the object-oriented development methodology that is supported by the system. The "mode-less" user interface and the performance benefits of the interface structure and mouse are also discussed.

[Mage84] Kenneth Magel. Principles for Software Environments, ACM Software Engineering Notes, Vol. 9, No. 1, January 1984.

Paper which lists and discusses a set of environment principles that include the following: generality (full life cycle support), adaptability (portability), user orientation (designed for a specific community), tailorability (adaptable to many types of interface devices), extensibility (new tools can be added), consistency (consistent use from part of the system to another), unification (user can anticipate how unfamiliar tools will operate), abstraction (hide as many details as possible), aggregation (bigger tools from smaller ones), incremental preparation, efficiency, predictability, subsetable, ability to group resources, and recoverability.

[Mats81] Y. Matsumoto, et. al. SWB System: A Software Factory, Software Engineering Environments, H. Hunke, Editor. North-Holland, 1981.

Paper discusses a large scale software factory that consists of three physical buildings, 2000 employees, a methodology, a software environment, and management and quality control. The software products are for critical applications (nuclear power stations) and there is an emphasis on using reusable code. The software environment consists of a number of tools and techniques that emphasize

the latter part of the life cycle (language and library processors, editors, debuggers, etc.). The plans for the environment include the addition of tools for the front end (SADT, design languages, etc.).

[Metz83] J. J. Metzger and A. Dniestrowski. PLATINE: A Software Engineering Environment, In Proceedings of SoftFair, IEEE Computer Society Press,(IEEE Order No. 83CH1919-0), July 1983.

Paper describes an environment that consists of a methodology (the PLATINE methodology) and a set of tools (the PLATINE tools). The environment supports the entire life cycle, is adaptable to product size, supports different types of users, and supports host/target development. The methodology consists of defining a software structure hierarchy (software structuration) which produces typed abstract objects which are then associated with elements (source, listing, binary, map, nomenclature, or status). The methodology also includes the production of software (merging of the elements), project management, and evolution. The user interface consists of a command language and a set of full screen panels. The tools include LSTR (specification of real time embedded systems, derived from PSL/PSA), SDL (system design representation), Metacomp (YACC like), EPCS (a project management tool based on DEC's project control system), a formatter (DEC's runoff), a screen editor (DEC's EDT), documentor (editor from source code), mail (DEC's), crossrf (data dictionary cross reference), complex (a complexity measure), a configuration controller, and a comparator.

[Oste81] Leon Osterweil. Software Environment Research: Directions for the Next Five Years, Computer, Vol. 14, No. 4, April 1981.

Paper discusses research issues associated with software engineering environments, in particular, the breadth of scope and applicability, user friendliness, reusability of internal components, tight integration of tool capabilities, and use of a central database. A five-year research plan is presented which includes studies of existing support systems, tool fragment studies, data base studies, construction, and test beds for configuring environments.

[Oste82] Leon J. Osterweil. Toolpack - An Experimental Software Development Environment Research Project, Proceedings of the 6th International Conference on Software Engineering, IEEE Computer Society NY,(IEEE Order No. 82CH1795-4), September 1982.

An implementation of [Oste81]. Paper presents an

environment under development that concentrates on tight integration of tool capabilities (use of tool fragments) and an underlying central database (virtual file system). See also [Cowe83].

[Pren81] Dan Prentice. An Analysis of Software Development Environments. In ACM Sigsoft Software Engineering Notes, Vol. 6, No. 5, October 1981.

A paper which emphasizes the problems. The issues discussed include lack of hardware support, high cost, user resistance to change, and poor user interfaces.

[Ridd81] W. E. Riddle. An Assessment of Dream, Software Engineering Environments, H. Hunke, Editor, North-Holland, 1981.

Paper reviews the DREAM system. DREAM is oriented to the development of concurrent systems using DREAM Design Notation (DDN), a language that can be used to model a total system including hardware, software, and wetware (people, etc.). The model reflects the externally observable characteristics of a system and is an adequate basis for preparing implementation plans. The DREAM system tools include a data base core that stores DDN fragments, bookkeeping tools (entry and retrieval), and decision-making tools for paraphrasing (a re-structured presentation), extraction (simulation), and consistency checking. The paper concludes with lessons learned and problems for the future.

[Ridd83] William E. Riddle. The Evolutionary Approach to Building the Joseph Software Development Environment, Proceedings of SoftFair, IEEE Computer Society Press, NY (IEEE Order No. 83CH1919-0), July 1983.

Paper reviews an effort to build an environment that was cut short due to the closing of Cray Labs. The Joseph environment was 30% completed. The paper includes a user scenario of the proposed environment which includes capabilities to view database information, extract database information, perform notation-directed editing, analyze changes, and deposit information into the database. The paper then discusses the work that was accomplished which includes a layered environment with Unix at the core, a set of integrated tools in the next layer (the crypt), and a requirements definition tool and a design definition tool in the outer layer (pharaoh and oasis). Pharaoh and oasis include viewing, notation-directed editing, and version control capabilities of requirements and design specifications. They use a notation that consists of key-words and description fragments.

[Rube83] Burt L. Rubenstein and Richard A. Carpenter. The Index Development Environment Workbench, Proceedings of SoftFair,

IEEE Computer Society Press,(IEEE Order No. 83CH1919-0), July 1983.

Paper presents a methodology and an associated environment for building application systems (informations systems for business applications). The methodology divides an application system into a dialogue manager, a database processer, an output processor, an action processer, an extended data dictionary, and a control monitor. The environment includes facilities for data dictionary specification, structured graphics, screen definition, output processing (for developing mock-ups), defining control between components, and generic data entry. An example of the use of the system is presented.

[Sava82] Ricky E. Savage, James K. Habinek, and Thomas W.. Barnhart. The Design, Simulation, and Evaluation of a Menu Driven User Interface, Proceedings of Human Factors in Computer Systems, Washington, D. C. Chapter, ACM, Gaithersburg, MD, 1982.

Paper discusses experiments relating to the user interface of an environment. An analysis of human errors led to the design of a system that provided an extensive hierarchy of menus for the inexperienced user and a variety of shortcuts to system functions for the experienced user.

[Shne80] Shneiderman, B. Software Psychology: Human Factors in Computer and Information Systems. Winthrop, Cambridge, Mass., 1980.

Book discusses a broad range of issues related to human factors in computers. Of particular interest to the developers of software engineering environments are the chapters on interactive interface issues and designing interactive systems. These chapters cover the user interface (control, response time, text editing, menu selection, error handling), the goals for interactive systems (simplicity, power, user satisfaction, reasonable cost), and the design process (from a human factors standpoint).

[Snow81] R. A. Snowdon. CADES and Software System Development, Software Engineering Environments, H. Hunke, Editor, North-Holland, 1981.

A review of one of the early large scale software engineering environments. The paper presents a history of CADES dating back to the early 1970's. CADES was established to provide a mechanism by which information relating to the structural model of a system could be made available to system designers early in the development process. Underlying CADES is a hierarchical database called PEARL. The establishment of CADES led to the

development of a structural modeling methodology: isolate functions, data, and constraints, produce data tree, produce function (holon) tree, consider next level of detail, code in Systems Description Language (SDL), and compile. Although CADES was developed to support the earlier phases of development, the author claims that CADES solutions are always sought for development or production problems and there is an increasing trend towards support for implementation.

[Solo84] Elliot Soloway. A Cognitively-Based Methodology for Designing Languages/Environments/Methodologies, Proceedings of the ACM Sigsoft/Sigplan Software Engineering Symposium on Practical Software Development Environments, ACM, New York, April 1984.

A paper that discusses issues relating to use of an environment. In particular, the author claims that environments should be developed based on a methodology that derives design implications based on tested hypotheses of why software developers work the way they do.

[Sten81] Vic Stenning, et. al. The Ada Environment: A Perspective, Computer, Vol. 14, No. 6, June 1981.

A paper that reviews the objectives (i.e., life-cycle support, open-ended environment, support for Ada, configuration control, project team support, portability, and host characteristics) and the architecture (i.e., the KAPSE/MAPSE/APSE approach, the database, KAPSE functions, the user interface, intertool interfaces, and tools) of the Ada Programming Support Environment (APSE).

[Steu84] H. G. Steubing. A Software Engineering Environment (SEE) for Weapon System Software, IEEE Transactions on Software Engineering, Vol. SE-10, No. 4, July 1984.

Paper presents a large scale environment called FASP that is hosted on multiple, large scale commercial computers. FASP primarily supports the latter stages of software development, but the extension to the requirements and design phase is discussed. The author attributes a two-fold increase in lines per month to FASP and an increase in software quality due to the tools, standards, and testing associated with the environment. The environment includes an underlying database made up of libraries: source library, object library, test library, interface library, production data library, and documentation library. The system is command driven where the commands consist of lower level system commands (command procedures). Testing is supported by the ATA (Automated Testing Analyzer) and there is support for multilanguages and multitarget computers.

[Stuc83] Stucki, Leon G. What about CAD/CAM for Software? The ARGUS concept, Proceedings of SoftFair, IEEE Computer Society Press, (IEEE Order No. 83CH1919-0), July 1983.

Paper proposes that software can be developed using a CAD/CAM approach with the aid of a software engineering environment. An overview of such an environment (ARGUS) is presented. ARGUS is a micro-based environment that was built on top of Unix. Argus is menu driven with a single key stroke approach. Six toolboxes are available at the top level; they are the management toolbox (scheduling tools, action item tracking tool, electronic spread sheet, and phone list update and retrieval system), the designer's toolbox (kernel CAD/CAM capabilities with a graphics/forms based approach), the programmer's toolbox (language-based, project-specific code template capability provided by a customizable editor and language specific syntax generation macros), the verifier's toolbox (analysis tools), and general/utility tools (general editing and communication tools). A noted CAD/CAM feature of ARGUS is the automatic projection of data to documentation and code templates from the underlying database.

[Tayl84] Richard N. Taylor and Thomas A. Standish. Steps to an Advanced Ada Programming Environment, Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society NY, (IEEE Order No. 84CH2011-5), March 1984.

Paper presents a research environment for exploring concepts and issues related to software engineering environments in general and the Ada programming language in particular. The environment called Arcturus currently includes interactive Ada (a Pascal superset), template-assisted editing, performance measurement (histograms or color), mixed compilation and interpretation, and an Ada program design language. Some concepts and issues being explored include complexity (does it scale up?), AVOS (Ada Virtual Operating System, i.e. an Ada command language), user interface issues (an Ada shell), mixing interpretation and compilation, layered architecture (i.e., device level, user interface level, tool level, foundation level), and analysis, testing, and debugging of tasking programs.

[Teit81] Warren Teitelman and Larry Masinter. The Interlisp Programming Environment, Computer, Vol. 14, No. 4, April 1981.

Paper presents a look at the Interlisp environment. Interlisp is an environment for users of Lisp (a non-procedural list processing language). The environment is very much language dependent and is intended for use by Lisp experts. Some representative facilities in Interlisp include: file package, masterscope (helps analyze the scope

of a change), DWIM (do-what-I-mean spelling corrector), iterative expressions, and the programmer's assistant.

[Teit81a] T. Teitelbaum and T. Reps. The Cornell Program Synthesizer: A Syntax-Directed Programming Environment, Communications of the ACM, Vol. 24, No. 9, September 1981.

Paper discusses an interactive programming environment with facilities to create, edit, execute, and debug programs written in a subset of PL/I. Editing is syntax-directed with underlying tree structures, predefined templates, and phrases to fill the templates. Execution of programs can be gear-shifted forward or backward with various controls on speed.

[Teit84] W. Teitelman. A Tour Through Cedar, Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society NY,(IEEE Order No. 84CH2011-5), March 1984.

Paper presents the facilities of the programming environment called Cedar. The Cedar environment emphasizes the use of parallel operation, multiple windows on a screen, and user interaction with a mouse pointing device. The environment supports the use of an "industrial strength" Pascal-like programming language. The tour makes stops at the display (bitmapped and object-oriented with the use of icons), viewer window package (supports multiple levels of windows which are tiled on the screen), whiteboards (work windows), tioga editor and document preparation system (supports tree structured documents, editing with the mouse, syntax-directed templates), user executive (programming interface), interpreter (for debugging), automatic storage management (garbage collector), rope (string) interface, bug tracker, electronic mail, support for parallelism, and icon editor (pixel oriented, graph editor).

[Tomo84] Tomoharu Mohri et. al. PDAS: An Assistant for Detailed Design and Implementation of Programs, Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society NY, (IEEE Order No. 84CH2011-5), March 1984.

Paper presents an environment that uses a forms-oriented approach to standardize document format and to prevent inconsistencies between documents and programs. There are 10 types of forms for design which are based on a forms-oriented language. The system structure consists of a forms-oriented editing subsystem, a document generation subsystem, a program construction subsystem (generation is based on module algorithm descriptions), a design database, and a component database (interchangable program components). An interesting aspect of the environment is

automatic Japanese to English translation from algorithm descriptions.

[Wass81] Wasserman, Anthony I. Tutorial: Software Development Environments, IEEE Computer Society Press, NY, (IEEE Order No. EH0187-5), 1981.

Tutorial is a reference collection of 39 papers including most of the landmark papers on software engineering environments.

[Wass83] Wasserman, Anthony I. The Unified Support Environment: Tool Support for the User Software Engineering Methodology, Proceedings of SoftFair, IEEE Computer Society Press, NY (IEEE Order No. 83CH1919-0), July 1983.

Paper presents an overview of the User Software Engineering methodology and the tools in the environment that support the methodology. The methodology involves users in the early stages of development and addresses user interactions with information systems. The tools in the USE environment include: the Troll relational database (underlies and is used by other tools), RAPID (rapid prototyping tool oriented to the development of information systems), PLAIN (a procedural language oriented to the development of information systems), Focus (screen-oriented editor and browser, and IDE (a software management and control tool).

[Wert82] Harald Wertz. The Design of an Integrated, Interactive, and Incremental Programming Environment, Proceedings of the 6th International Conference on Software Engineering, IEEE Computer Society NY, (IEEE Order No. 82CH1795-4), September 1982.

A paper that presents the details of a proposed environment that integrates editing, executing, and annotating programs.

[Wirt81] N. Wirth. Lilith: A Personal Computer for the Software Engineer, Proceeding of the 5th International Conference on Software Engineering, IEEE Computer Society NY, (IEEE Order No. 81CH1627-9), March 1981.

Paper discusses the development, features, and architecture of the Lilith programming environment for Modula-2. The system provides a high bandwidth between the user and the system partly through the use of a mouse pointing device and the hardware structure. The display is suitable for text, diagrams, or graphics.

[Wolf81] Martin I. Wolfe, et. al. The Ada Language System, Computer, Vol. 14, No. 6, June 1981.

Paper discusses the Ada Language System which is currently

under development at SofTech. The system will provide capabilities at the MAPSE level [Buxt80]. Issues relating to the development of an Ada compiler are also discussed.

APPENDIX A

Categorization of papers

Overview of software engineering environments:

[Bran81] [Bran81a]
[Buxt80a] [FIPS99]
[Haus81] [Howd82]
[Hunk81] [Oste81]

Issues in building software engineering environments:

[Barn82] [Gutz81]
[Houg84] [Huff81]
[Mage84] [Pren81]
[Sava82] [Shne80]
[Solo84]

General software engineering environments:

[Boeh84] [Hall80]
[Kern81] [Kuo83]
[Lebl84] [Mats81]
[Metz83] [Tomo84]
[Ridd83] [Steu84]
[Stuc83]

Systems development environments:

[Alfo81] [Ridd81]
[Rube83] [Snow81]
[Wass83]

Programming environments:

[Baye81] [Buxt80]
[Camp84] [Cowe83]
[Cox83] [Deli84]
[Fair80] [Feil81]
[Fisc84] [Guya84]
[Houg82] [Love83]
[Oste82] [Sten81]
[Tayl84] [Teit81]
[Teit81a] [Teit84]
[Wert82] [Wirt81]
[Wolf81]

APPENDIX B

General References on Software Engineering Environments

1. Branstad, Martha A. and W. Richards Adrion, Editors. NBS Programming Environment Workshop Report. National Bureau of Standards, NBS SP 500-78, June 1981.

2. Hunke, H., Editor, Software Engineering Environments, North-Holland, 1981.

3. Proceedings of the 5th International Conference on Software Engineering, IEEE Computer Society, NY (IEEE Order No. 81CH1627-9), March 1981.

4. Proceedings of the 6th International Conference on Software Engineering, IEEE Computer Society, NY (IEEE Order No. 82CH1795-4), September 1982.

5. Proceedings of the 7th International Conference on Software Engineering, IEEE Computer Society, NY (IEEE Order No. 84CH2011-5), March 1984.

6. Proceedings of the ACM Sigsoft/Sigplan Software Engineering Symposium on Practical Software Development Environments, ACM, New York,, April 1984.

7. Proceedings of SoftFair, IEEE Computer Society, (IEEE Order No. 83CH1919-0), July 1983.

8. Wasserman, Anthony I., Guest Editor, Special Issue on Programming Environments, Computer, Vol. 14, No. 4, April 1981.

9. Wasserman, Anthony I., Tutorial: Software Development Environments, IEEE Computer Society, (IEEE Order No. EH0187-5), 1981.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See Instructions)	1. PUBLICATION OR REPORT NO. NBSIR-3113	2. Performing Organ. Report No.	3. Publication Date April 1985

4. TITLE AND SUBTITLE

Annotated Bibliography of Recent Papers on Software Engineering Environments

5. AUTHOR(S)

Raymond C. Houghton, Jr., and Dolores R. Wallace

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

National Bureau of Standards
Gaithersburg, MD 20899

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This document reports on the contents of fifty-five recent papers on software engineering environments. Several of these papers present an overview of software engineering environments. Other papers discuss issues to be considered in building software engineering environments. The remaining papers describe general software engineering environments, system development environments, and programming environments.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

distributed workstation environments; documentation; interactive programming environments; software engineering environments; systems development environments; user interfaces

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

24

15. Price

$7.00

USCOMM-DC 6043-P80