DETRACT: A DEsign TRACking Tool

TR028

by

Torab Torabi, Karl Reed

Amdahl Australian Intelligent Tools Program

Department of Computer Science and

Computer Engineering

La Trobe University

Bundoora, Victoria, Australia 3083

Abstract

This report is a review of the paper presented by G T Jayaputera and K E Cheng [50]. The report compares the Jayaputera's paper and his prototype, SoftEAM, with DETRACT [18] project implemented in AAITP. Finally we will discuss the extent to which each system would be able to help the designers during their development of a design.

1. Introduction

DETRACT, DEsign TRACking Tool, is part of HyperCase [*] project in AAITP. Our study of DETRACT has been extended to a Design Tracking and Reasoning Tool. DETRACT Model is designed to not only be able to record design decisions, but also to be able to reason about the decisions made and alternatives rejected. While Design Recording provides for a better design documentation, Replay of Design History, and support for Maintenance; Design Reasoning will provide for Design Consistency and Integrity, Designer's support for decision making , and Design Reusablity. With above objective the design recording is carried out with respect to other important information in design life cycle.

2. Representation

The DETRACT model consists of the following artifacts and the relations among them:

- Design artifact: This is any kind of design document or object in design.
- Issue/ Problem Artifact: This artifact represent an issue or problem relating to a design artifact.
- Decision Artifact: The Decision artifact represents a decision made regarding an issue.
- Alternative Artifact: This alternative decision could be chosen regarding an issue.
- Constraint artifact: This artifact represents any constraint relating to a design artifact.
- Goal artifact: This artifact represents goal pertaining to a specific design artifact.
- Agenda Item: This is an issue regarding a design artifact which has been encountered, but decision making for the issue is postponed to the future in design process.

Therefore we consider 2 types of artifacts, Design artifact and Design Reasoning artifact. On the next page you can see an abstract representation of the Design Decision environment and the relationships between the design artifacts:

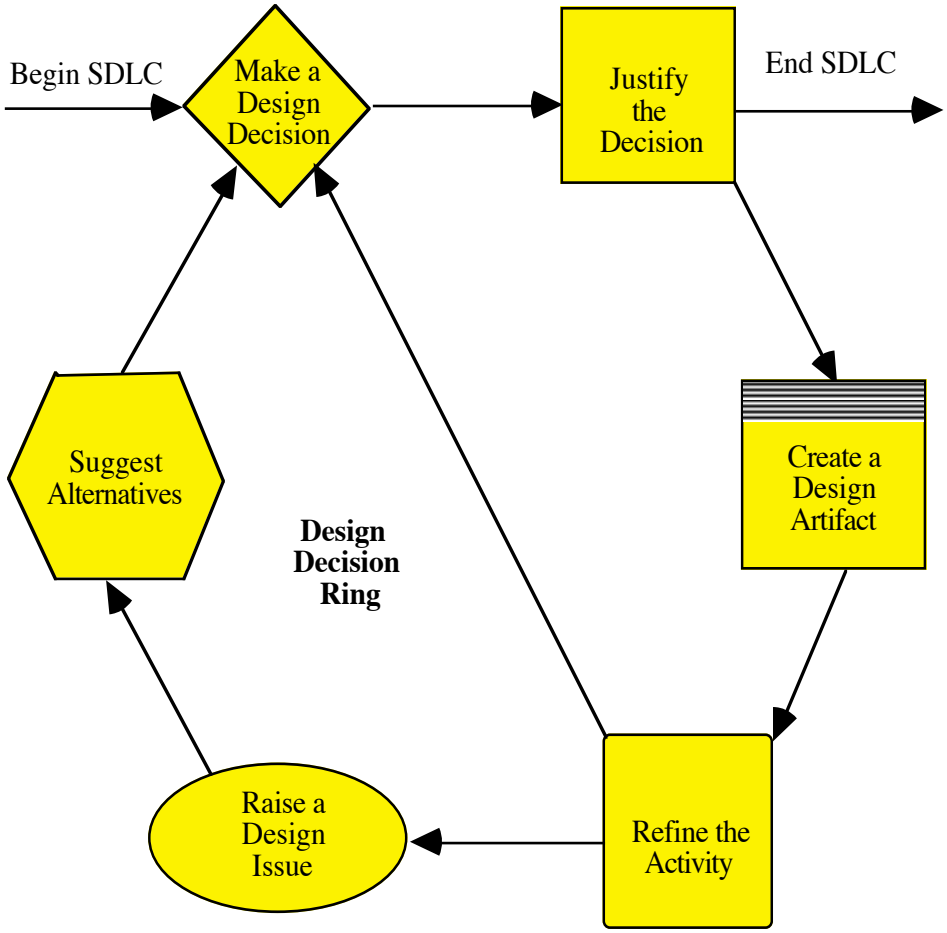

Figure 5: Abstract Model for Design Decision

3. Design and Reasoning Repository

For recording design and Reasoning information two separate repositories are considered. The design repository contains all information regarding the design. The reasoning repository consists of all information resulted from the relationships between artifacts and information pertaining to the reasoning artifacts.

The provision of separate repositories provides for logical separation of design documents and reasoning information. It also enables the designer to access the design information without disturbing reasoning repository. Information in Reasoning repository can also be accessed in a new design as a reusable data base or guidelines for decision making.

4. Recording

Using an event mechanism, every decision made by the designer during the design process is recorded in design decision artifact. Every Design Decision artifact is linked to the Issue artifact which "required" this decision to be made, and also linked with any artifact resulting from that Design Decision. These relations are automatically recorded in each artifact (see figure 6).

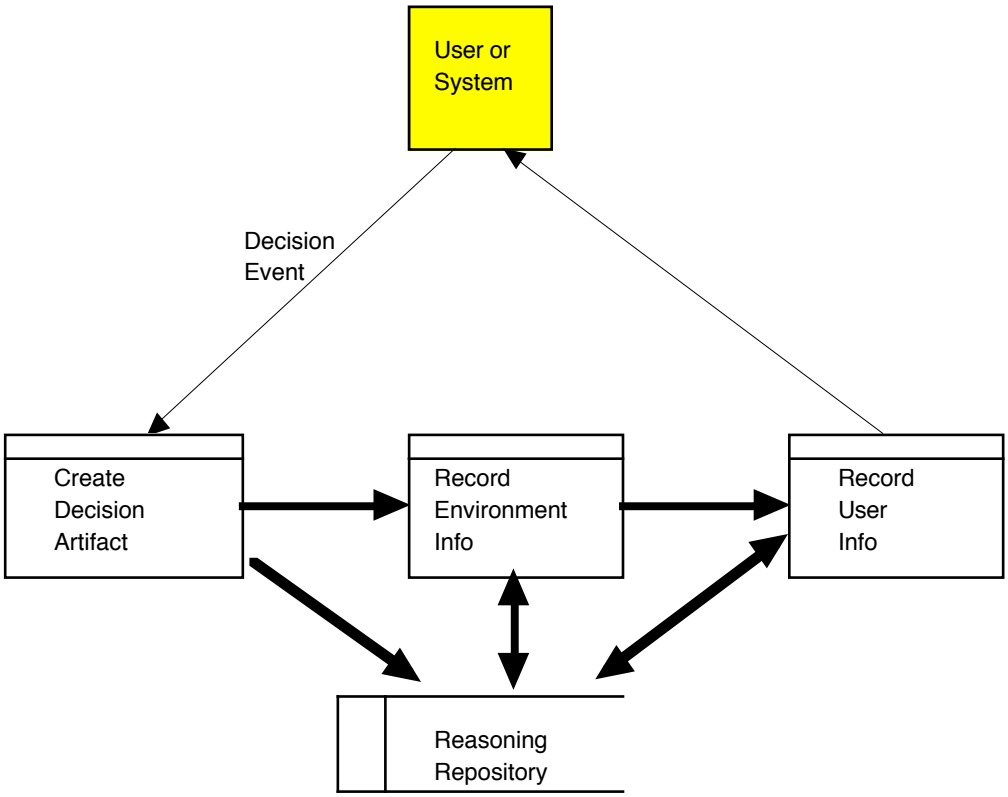

Figure 6: Recording Design Decisions

The information recorded in this process would provide a hierarchical and relational design history. This information is also augmented with further information provided by the designer.

Decisions tracked during design are categorised into different levels of hierarchies. Decisions recorded in one level are saved as a set of Decision Transactions. The decisions in a transaction are processed only when a decision belonging to the upper level, which this Transaction relates to it, is made. This would signal the end of decision making process in the lower level. This process will ensure that a unique instance of a specific decision would be present in a Decision Transaction. This type of processing of decisions is very important, otherwise the Design Decision Repository would be saturated with repetitive information.

This categorisation of decisions also provides the design history with well organised and structured information.

We divide decisions into two types of Implicit and Explicit. Implicit Decision occur and are recorded when the designer creates or refines a design artifact. Such a decision is recorded without designer's intervention. The instance would contain all the environment design information related to the decision. Explicit Decision is recorded when the designer makes a decision after he/she is prompted with possible alternatives for resolving an issue.

5. Navigation

Because the reasoning artifacts and the design artifacts are related together using various type of relations, it is possible to navigate to desired design artifact. The navigation of design artifacts provides a design history of the system. The DETRACT provides number of ways of navigating between reasoning and design artifacts.

Based on relations between the different artifacts of the design represented in figure x the following kinds of navigation is provided in the DETRACT:

- Within a set of artifacts of a specific type.
- To all artifacts immediately resulted from a specific artifact.
- To an artifacts where a specific artifact immediately resulted from it.
- To the index for a specific artifact
- Within different indexes

6. User query

The designer should be able to ask a question regarding the design decision he/she has made and to receive cross-referencing information regarding different artifacts or concepts in the design.

Type of queries the designer may ask can be categorised to the following groups:

- Ordering Design Decisions: Decisions can be ordered by different attribute of design decision such as: time, person, level of Design Life Cycle, artifacts they relate to, ...

- Finding Design Decisions: Design Decisions can be located by the same attributes mentioned earlier.

- Finding Design Artifacts: Different Design Artifacts can be located with the respect of their relations to each other and to Design Decision.

- Making Future Design Decisions: Agenda Items serve as a resource for answering queries regarding future decision making.

This information would provide the designer with a picture of design issues involved during different stages of the design and kind of actions was taken for issues. Thus providing the designer a better understanding of the design.

7. Reasoning

We believe that Reasoning regarding the design decision can be provided if design decision are recorded and relations between different artifacts of the design is retained. This reasoning would constitute a "justification" for individual decisions were made, why some alternative decisions were rejected, and how these decisions affected the design.

Recording justifications during the design also enables DETRACT to verify that design decision would not conflict with the design constraint and do satisfy the design goals.

During this process certain issues are postponed for decision making. Those issues are recorded as Agenda Items. Agenda Items would be given priority based on number of criterias such as stage of design, latest time for resolving an issue, ...

8. Architecture

DETRACT, Design Tracking and Reasoning Tool, is been implemented on Macintosh using HyperTalk Event-Based language.

Below is a schematic of the architecture of the DETRACT Showing its relation with different part of the system. In final implementation the DETRACT would be integrated with the HyperEdit, the graphical design editor of the AAITP.

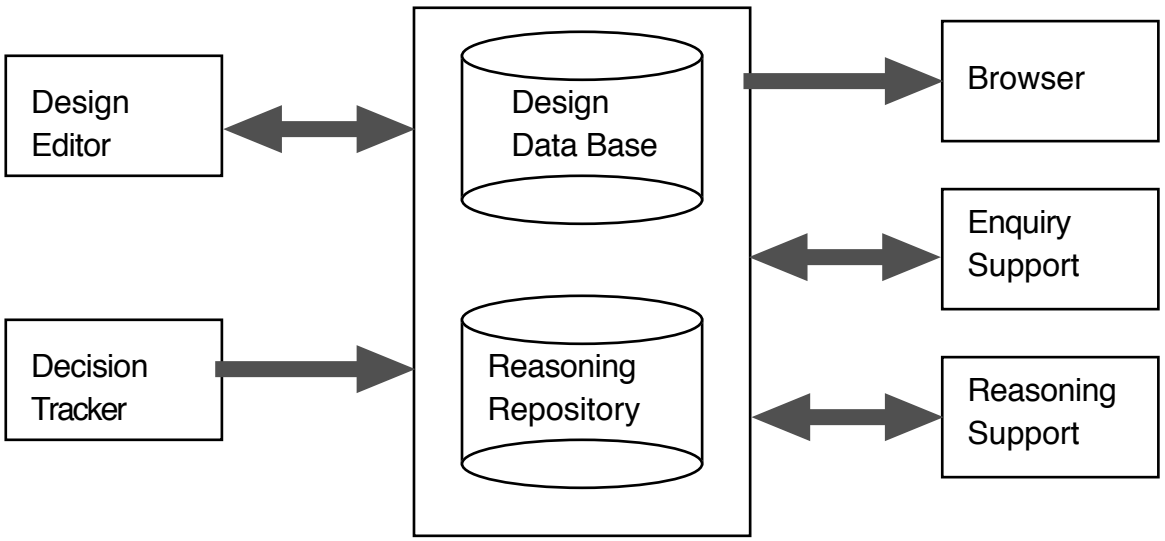

Figure 7: DETRACT System Architecture

9. Support

Design reasoning sub-system provides a support environment for the designer. Some the functions provided by sy**stem are:**

- Component Re useability: Since the user can query to look for the type of decisions made, therefore he/she would know what functions have been designed or decided on. The designer, if necessary, can use the reusable components related to those decisions further down the design.

- Small-Scale Design Plan: Knowing the Agenda Items and priorities and other attributes of them, the designer can sort out a Small-Scale Dynamic Design Plan which would guide the designer for future activities.

- Design Integrity: The reasoning can cross check the design decision with the design Constraints. This facility would provide the design Integrity and consistency which is hard to provide in a large scale design.

10. Reflecting Decisions on Design

Decisions made during the design have a direct impact on present status of the design and also the future of design process. It is a valuable process to reflect the effect of design decision on relating design artifacts. The following three set of artifacts would directly be affected as a result of a design decision:

- Recent Design Artifact: The recent design artifact in process of creation or modification will be affected as a result of a decision made. For example when deciding which alternative indexing technique to be used, the selected technique specification/implementation will be added to the design artifact.

- Constraint Artifact: As a result of a design decision made, some constraints will be added to the design, an existing constraint artifact will be modified or a new one will be created.

- Goal Artifact: As a result of a design decision made, some goals or sub-goals may be removed or added to the design, an existing goal artifact will be modified or a new one will be created.

11. Conclusion

In DETRACT we introduced the concept of Decision Transactions. Decision Transactions not only provide for delta evaluation of design revisions, but also allows the designer to roll back to some point of previous design stage. Decisions Transaction concept also allows for unique copy of a decision in the design repository.

DETRACT provides different kinds of navigation between design artifacts and reasoning artifacts as mentioned in section 3.4. Recording the relationships between artifacts and the use of event handling in HyperTalk has provided a very smooth type of navigation between artifacts.

The provision of Relating design decisions to Alternatives, Constraints, Goals and Agenda Items enables DETRACT to reason about design decisions. The advantages of this provision is stated in more detail in sections 3.6 and 3.8.

User query sub-system in DETRACT provides the designer with different search and ordering facilities. In a large scale design and specially when the designer is working in a team, the above information can provide him with component reusablity, design integrity and better understanding of the large design.

The provision of reflecting design decision to the design itself and to the reasoning artifacts in DETRACT has a very important significance. This feature would keep the design up to date with regarding the decisions made, it will allow for a consistent and non conflicting decision making environment. Simulation of alternative decision reflection on design will provide with different design spaces, therefore a different design scenarios can be simulated.

12. Bibliography

[1] Kozaczynski W., Maciaszek. "Design Recording As Integral Aspect of Software Engineering", ASWEC 90, pp 87-92.

[2] Potts C., Bruns G. "Recording The Reason for Design Decision", ICSE, IEEE 1988, pp 418-427.

[3] Fisscher G., Girgensohn A. " Supporting Software Designers With Intelligent Domain-Oriented Design Environment", IEEE Transactions on Software Engineering, vol 18, no 6, l992, pp 511-522.

[4] Fickas S., Helm R. " Knowledge Representation And Reason In The Design of Composite Systems", IEEE Transactions on Software Engineering, vol 18, no 6, l992, pp 470-482.

[5] Setliff D., Rutenbar R. " Knowledge Representation And Reasoning In a Software Synthesis Architecture", IEEE Transactions on Software Engineering, vol 18, no 6, l992, pp 523-533.

[6] Boehm B., " A Spiral Model Of Software Development And Enhancement", Comp, May 1988, pp 61-72.

[7] Lor K.E., Berry D." Automatic Synthesis Of SARA Design mods From System Requirements", IEEE Transactions on Software Engineering, vol 17, no 12, 1991.

[8] Ramesh B., Dhar V. " Supporting Systems Development by Capturing Deliberations During Requirements Engineering", IEEE Transactions on Software Engineering, vol 18, no 6, l992, pp 498-510.

[9] Lee J. " Extending The Potts And Bruns Model For Recording Design Rationale", IEEE 1991.

[10] Tsai J.P., Ridge J.C. " Intelligent Support for Specifications Transformation", IEEE, Nov 1988.

[11] Fickas S., Nagarajan p. " Critiquing Software Specifications", IEEE, Nov 1988.

[12] Kaiser G.E., Feiler P.H., Popovich S.S. " Intelligent Assistance for Software Development and Maintenance", IEEE Software, 1988, pp 40-49.

[13] Nielson J. " The Art of Navigating Through Hypertext", CACM vol 23, no 3, 1990, pp 298-310.

[14] Conklin J. " Hypertext: an Introduction and Survey", IEEE Comp, 1987, pp 17-39.

[15] Rich C., Feldman Y. " Seven Layers of Knowledge Representation and Reason in Support of Software Development", IEEE Transactions on Software Engineering, vol 18, no 6, l992, pp 451-469.

[16] Schneidewind N. " The State of Software Maintenance", IEEE Transactions on Software Engineering ,vol 13 ,no 3, 1987.

[17] Arango G., Bruneau L." A Tool Shell for Tracking Design Decisions.", IEEE Software, 1991, pp 75-83.

[18] Cybulski J., Reed K. " A Hypertext-Based Software Engineering Environment", IEEE Software Mar 1992, pp 63-68.

[19] Selby R., Basili V., Baker F. " Cleanroom Software Development : An Empirical Evaluation", IEEE Transactions on Software Engineering, vol 13, no 9, 1987, pp 1027-1037.

[20] Blum B. " A Ten-year Evaluation of an Atypical Software Environment ", Software Engineering Journal, Sep 1991, pp 347-354.

[21] Barstow D. " Artificial Intelligent and Software Engineering", ch 16.

[22] Rich C., Waters R. C. " The Programmer's Apprentice: A Research Overview", IEEE Comp 1988, pp 11-24.

[23] Baxter I. D. " Design Maintenance Systems", CACM, vol 35, no 4, 1992, pp 73-89.

[24] Kozaczynski W. " The 'Catch-22' of Re-Engineering", IEEE ICSE, Nice, France 1990, pp 119.

[25] Maicchi M. " Re-Engineering: Can a Program Put Intelligence In Stupid Program", IEEE ICSE, Nice France, 1990, pp 123.

[26] Blum B.I. " Thoughts on the Software Process", ACM SIGSOFT, vol 11, no 4, 1986.

[27] Ambras J., O'Day V. " Microscope: A Knowledge-Based programming Environment", IEEE Software ,1988, pp 50-58.

[28] Meyers R. J., Parrish J. W. " The Macintosh Programmer's Workshop", IEEE Software 1988, pp 59-66.

[29] Sneed H.M. " The Myth of "Top-Down" Software Development and its Consequences for Software Maintenance", IEEE Conf. on Software Maintenance 1989, pp 22-29.

[31] Sommerville I. "Software Engineering", Third Edition, Addison-Wesley, 1989.

[32] Reed K., McLaughlin A. " A Software Development Design Tracker", Dept. of Comp Sc and Comp Eng, La Trobe Uni, Jun 1991.

[33] Borgida A., Jarke M " Knowledge Representation And Reasoning In Software Engineering", IEEE Transactions on Software Engineering, vol 18, no 6, l992, pp 449-450.

[33] Madhavji N.H., Schafer W. "Software Process and its Support", SE Journal, Sep. 1990, pp 229.

[34] Hoffnagle G .F. "Engineering the Software Development Process", SE Journal, Sep. 1990.

[35] Osborne W. M., Chilkofsky E. J. "Fitting Pieces to the Maintenance Puzzle", IEEE Software, 7(1): pp 11-12, Jan 1990.

[36] Chilkofsky E. J., Cross II J. H. "Reverse Engineering and Design Recovery: A Taxonomy", IEEE Software, 7(1): pp 13-17, Jan 1990.

[37] Bsdili, V. R. "Viewing Maintenance as Reuse-Oriented Software Development", IEEE Software, 7(1): pp 19-25, Jan 1990.

[38] Rugaber S., Ornburn S. B., LeBlanc R. J. "Recognising Design Decisions in Programs", IEEE Software, 7(1): pp 46-54, Jan 1990.

[39] Rich C., Wills LAM. "Recognising a Program's Design: A Graph-Parsing Approach", IEEE Software, 7(1): pp 82-89, Jan 1990.

[40] Samuelson P. "Reverse Engineering Someone Else's Software: Is It Legal?", IEEE Software, 7(1): pp 90-96, Jan 1990.

[41] Arnold R. S., Martin R. J. "Software Maintenance", IEEE Software, 3(3): pp 4-5, May 1986.

[42] Letovsky S., Soloway E. "Delocalised Plans and Program Comprehension", IEEE Software, 3(3): pp 41-49, May 1986.

[43] Ourston D. "Program Recognition", IEEE Software, IEEE Expert 4(4): pp 36-49Winter 1989.

[44] Sneed, H. M. "Software Renewal: A Case Study", IEEE Software: pp 56-633, July 1984.

[45] Neighbours J. M. "The Darco Approach to Constructing Software from Reusable Component Software, SE 10(5): pp 564-574, Sep. 1984.

[46] Meyer B. "Reusablity: The Case for Object-Oriented Design", IEEE Software: pp 50-64, March 1987.

[47] Card D. N., Church V. E., Agresti W. W. "An Empirical Study of Software Design Practices", Transaction on Engineering SE 12(2): pp 264-271, Feb. 1986.

[48] Freeman P. "Software Design Representation: Analysis and Improvements", Software Practice and Experience, 8(5): pp 513-528, Sep. 1978.

[49] Freeman P. "Software Design Representation: A Case Study", Software Practice and Experience, 8(5): pp 501-512, Sep. 1978.

[50] Jayaputea G. T., Jayaputera K. E. "SoftEAM: A Design History and Justification Maintenance Tool" Proceedings os ASWEC, pp 185-196, Sep 1993.

[51] Bolognesi T., Brinksman E. "Introduction to the ISO Specification Language Lotos" Computer Networks and ISDN Systems, 14(1): pp 25-29, 1987.

[52] Tichy W. F. "RCS - A System for Version Control" Software Practice and Experience, 15(7): pp 637-654. July 1985.

[53] Rochkind M. J. "The Source Code Control System" IEEE Transactions on Software Engineering, SE-1: pp 364-370, 1975.

[54] Jackson P. "Introduction to Expert Systems" Second Edition, Chapter 3, pp 48-52, Addison Wesley Publisher Ltd, 1990.

[55] Shinghal R. "Formal Concepts in Artificial Intelligence" Chapter 12, pp 453-457, Chapman & Hall Computing, 1992.

[56] Shirai Y., Tsujii J. "Artificial Intelligence: Concepts, Techniques and Applications" Chapter 4, pp 53-55, John Wiley & Sons, 1984.

[57] Sommerville I. "Software Engineering" Third Edition , Addison Wesley Publisher Ltd, 1989.

[58] Harel D. "On Visual Formalisms" Communications of ACM, 31(5): pp 514-530, May 1988.

[59] Hull R., King R. "A Tutorial on Semantic Database Modelling" Research Foundation in Object-Oriented and Semantic Database Systems, pp 1-33. Printice Hall Series, 1990.

[60] K. Reed, "The Original Technical Program for the Amdahl Australian Intelligent Tools Program" AAITP Tecchnical Report TR001, Dept. of Computer Science & Computer Engineering, La Trobe University, 1988.

[61] A. Proestakis "HyperEDIT - Functional Description," AAITP Tecchnical Report TR007, Dept. of Computer Science & Computer Engineering, La Trobe University, 1992.

[62] J. Cybulski, A. Proestakis "HyperEDIT: An Object-Oriented Diagram Meta-Editor" AAITP Tecchnical Report TR009, Dept. of Computer Science & Computer Engineering, La Trobe University, 1991.

[63] K. Reed, J. Cybulski "HyperCASE: A Hypertext Based Software Engineering Environment" 1991.

[63] K. Reed, J. Cybulski "Integrating Hypertext and CASE: The Emerging Technology," AAITP Tecchnical Report TR010,

Dept. of Computer Science & Computer Engineering, La Trobe University, 1991.

[64] A. Proestakis "HyperEDIT: Architectural Overview and Distributed Building Process Description" AAITP Tecchnical Report TR0017, Dept. of Computer Science & Computer Engineering, La Trobe University, 1992.

[65] J. Cybulski "SODA: Software Designer's Aide" AAITP Tecchnical Report TR019, Dept. of Computer Science & Computer Engineering, La Trobe University, August 1992.

[66] D. Cleary, K. Reed "PROTRACT: A Computerised Tool for Tracking Software" AAITP Tecchnical Report TR020, Dept. of Computer Science & Computer Engineering, La Trobe University, 1993.

[67] A. McLaughlin, K. Reed "Design Rationale: The Missing Ingredient in Software Development" AAITP Tecchnical Report TR009, Dept. of Computer Science & Computer Engineering, La Trobe University, 1992.