Developing Software is like Talking to Eskimos about Snow

John McDermott

Digital Equipment Corporation
111 Locke Drive, LM02-1/K11
Marlboro, Massachusetts 01752

Context

It's not clear to me exactly what people in AI do, but one thing that is talked about a lot is how important knowledge is. One of the things knowledge is supposed to be good for is solving hard problems. The idea, as I understand it, is that one characteristic of a large set of problems is that their solutions are radically contingent on the peculiarities of the various situations in which the problems are instantiated. To anyone who is ignorant of most, or even many, of the peculiarities, those problems appear hard. It is only to an agent who has knowledge of almost all of the relevant peculiarities that the problems appear straightforward.

So what does this have to do with software development? Well, Bob Balzer claims to believe at least three things [Balzer, 90]:

1. AI was expected to make software development less hard.
2. It hasn't.
3. The reason AI has failed is its reliance on isolationist technology and approaches.

AI doesn't own very many problems. It does own making software development less hard. It is usual to expect disciplines to make progress on the problems they own. So I share Balzer's first belief.

With respect to his second belief, it is clear that AI has not yet done much to make software development less hard. And AI certainly has done some things to make software development harder. So the only difficulty I have with Balzer's second belief is that it understates the case.

The Disagreement

Would that Balzer had only two beliefs. His third belief -- that AI has failed because of its reliance on isolationist technology and approaches -- shows a complete lack of appreciation for why we have failed. We have failed because we can't yet think about the software development problem clearly. The problem is essentially one of mapping from task features, described at some appropriate level of abstraction, to program features, described at some appropriate level of abstraction. Or in other words, the problem is mapping from knowledge level objects to symbol level objects [Newell, 80].

What we don't yet know is what the helpful abstractions at the knowledge level and the symbol level are. Happily, a whole bunch of work in AI is focused on just this question (eg, [Bachant, 89], [Clancey, 89], [Lowry, 89], [Marcus, 88], [McDermott, 89], [Musen, 89], [Rich, 90], [Yost, 89]. Over the past decade or so, the concept of rapid prototyping as an aid to software development has come into vogue. The idea is, of course, a wonderful one -- probably forever, but for sure in these primitive times when we don't yet know even how to talk about either tasks or programs in a way that doesn't obscure the mapping from the one to the other. Rapid prototyping is helpful because, given an approximation to a desired program, users can at least point to inadequacies the program has. Soon, hopefully, we will be able to communicate with words -- with words for patterns which at the moment we don't know we see.

An Example

I've recently come across a piece of work that is moving us exactly toward where I think we have to go [Roth, 90a; Roth, 90b]. I'm going to give an example from this research to try to convey at least the spirit of the enterprise. The focus of the work has been on how to automate the design and graphical presentation of information. The system developed, SAGE, is an intelligent interface which receives information from an application program and designs a combination of graphics and text that effectively conveys that information. SAGE embodies a way of thinking about information and about users' goals, and a way of thinking about graphical displays, that makes the mapping from one to the other fairly straightforward.

Figure 1 Information Provided to SAGE

Data

Activities

Activity	Start-Date	End-Date	Duration	Resource
Act-1	34	38	4	Sun
Act-2	2	16	12	TI
Act-4	30	34	4	Sun
:	:	:	:	:

Departments

Department	Responsible-for
CAD-Dept	Act-1, Act-2, Act-4, ...
Structural-Dept	Act-19, Act-20, Act-23, ...
Assembly-Dept	Act-33, Act-34, ...

Objects & Relations

Domain	Relation	Range
Department	Responsible-for	Activity
Activity	Start-date, End-date	Date
Activity	Duration	Number-of-weeks
Activity	Requires	Resource

Characterization of Objects

Object & Sets	Ordering	Coordinate/Amount	Domain
Department	Nominal	--	--
Activity	Nominal	--	--
Resource	Nominal	--	--
Date	Quantitative	Coordinate	Time
Number-of-weeks	Quantitative	Amount	Time

Characterization of Relations

Relation	Coverage	Cardinality	Uniqueness
Responsible-for	No	Varied	Yes
Start-date, End-date, Duration	Yes	1	No
Requires	Yes	1	No

Characterization of Relationships Among Relations

Relationship	Role	Relation
Interval	Beginning	Start-date
	Size	Duration
	End	End-date

Characterization of User's Goals

Goal	Relations
Visualize-correlation	Responsible-for
	Start-date, End-date
	Duration
	Requires

SAGE attends to the following four kinds of task features:

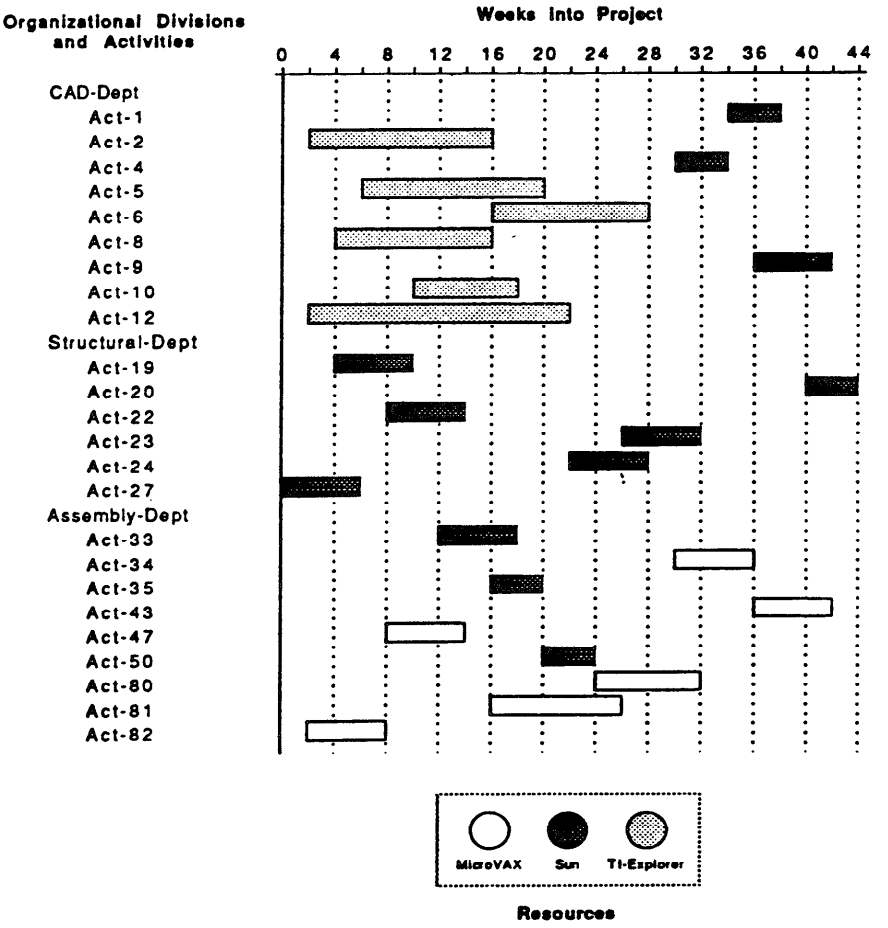
1. features which distinguish the kinds of information each graphical technique can express,
2. features which order graphical techniques based on how effective they are at conveying different information,
3. features which define users' purposes,
4. features which determine how information should be integrated within a display.

SAGE has a language for describing graphical displays and information which it uses to automatically produce many complex, creative pictures through the synthesis of simple techniques. Figure 2 illustrates the kind of picture SAGE can compose. Figure 1 displays the input that allowed SAGE to compose that picture.

In this case, the inputs to SAGE come from a query to a project management database. The manager who made the query was considering how to allocate computer resources to departments for an upcoming project. The data she requested included the departments involved in the project and the activities they would be responsible-for, the activity start-dates, end-dates, durations, and the required resources. Figure 1, in addition to showing the information retrieved from the project management database, contains a number of *data characterizations*. These data characterizations -- information about what features of the task have relevance -- provide SAGE with precisely the information it needs in order for it to exploit its knowledge of how to compose pictures.

The example illustrates what it means to have helpful abstractions at the knowledge level and at the symbol level:

Figure 2 The Picture Produced by SAGE



- knowing that resources comprise nominal (unordered) sets and that number-of-weeks is quantitative enables SAGE to use color for the requires relation, but not for the duration,
- knowing that dates refer to time enables SAGE to honor the convention that time is visualized horizontally, not vertically,
- knowing that the responsible-for relation maps "uniquely" to activities (ie, each activity is associated with one department) enables the hierarchical representation of the vertical axis,
- knowing that the goal is to see the correlations among all the relations leads SAGE to encode them using different properties of a single graphical object: the color, vertical and horizontal position, and length of interval bars; (SAGE also can infer that the bars' vertical position reflects the department associated with each activity),
- knowing that start-date, end-date and duration comprise an interval, enables SAGE to integrate them as such in a single bar,

The knowledge that SAGE has that allows it to exploit the data characterizations includes

- constraint knowledge: for example, representing resources using different shapes instead of colors would have prevented display integration, because the interval bars are already constrained in shape; SAGE considers spatial and other graphical constraints when searching for a way to integrate a picture,
- picture organizational knowledge: for example, in the absence of direct goals to the contrary, SAGE used the order in which the relations were requested to determine that it should index (organize) the picture by department rather than by a different property (eg, resource),
- effectiveness knowledge: for example, SAGE knows that color is good for distinguishing among three resources, but not twenty.

Conclusion

AI almost has it within its grasp to make software development easier. Though it probably wouldn't hurt if we were less isolationist, the primary thing we need to do is identify helpful abstractions for knowledge level and symbol level objects so that program pieces can identify and compose themselves on the basis of immediately salient characteristics of tasks. If we look at our past just right, it's clear this is work we've been preparing ourselves for for decades. Now it's time to do something about it.

References

[Bachant, 89] Bachant, J., and E. Soloway. The Engineering of XCON. *Communication of the ACM*, 32, 3, 1989.

[Balzer, 90] Balzer, R. AI and Software Engineering -- will the Twain Ever Meet? Proceedings of the Eighth Conference of the American Association for Artificial Intelligence, Boston, Massachusetts, 1990 -- or if not there, personal communication.

[Clancey, 89] Clancey, W. J. The Knowledge Level Reinterpreted: Modeling How Systems Interact. *Machine Learning*, 4, 3/4, 1989.

[Lowry, 89] Lowry M., and R. Duran. Knowledge-Based Software Engineering. *Handbook of Artificial Intelligence*, Vol. 4, Addison-Wesley, 1989.

[Marcus, 88] Marcus, S. (ed). *Automating Knowledge Acquisition for Expert Systems*. Kluwer, 1988.

[McDermott, 89] McDermott, J. The World Would Be a Better Place if Non-Programmers Could Program. *Machine Learning*, 4, 3/4, 1989.

[Musen, 89] Musen, M. Automated Support for Building and Extending Expert Models. *Machine Learning*, 4, 3/4, 1989.

[Newell, 81] Newell, A. The Knowledge Level. *AI Magazine*, 2, 1, 1981.

[Rich, 90] Rich, C., and R. Waters. *The Programmer's Apprentice*. Addison-Wesley, 1990.

[Roth, 90a] Roth, S. and J. Mattis. Automatic Graphic Presentation for Production and Operations Management Systems. Proceedings of the Fourth International Conference on Expert Systems, Hilton Head Island, South Carolina, May, 1990.

[Roth, 90b] Roth, S. and J. Mattis. Data Characterization for Intelligent Graphics Presentation. Proceedings of the Conference on Computer Human Interaction, Seattle, Washington, April, 1990.

[Yost, 89] Yost, G. and A. Newell. A Problem Space Approach to Expert System Specification. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan, 1989.