Intelligent Agents in Software Reuse Repositories

Barry G. Silverman
Nabil Bedewi
Alfredo Morales

Institute for Artificial Intelligence
George Washington University
Staughton Hall, Room 206
Washington DC 20052
Barry@seas.gwu.edu

ABSTRACT

This article explores the role for intelligent agents in "reuse supported problem soving" in general, and in software reuse repositories in particular. As reuse repositories grow in size and extent, the "traditional" browse and surf mode becomes less effective, and simple search agents alone are not the answer. This article postulates a human information processing model of reuse supported problem solving, and then suggests a suite of cooperating agents that anticipate and better support the needs of repository users. Empirical results are presented from three separate repositories that compared the traditional vs. suite-of-agents architectures. These results show the intelligent agent architecture offers many benefits if agent parameters are properly tuned for degree of intervention and control, purpose and intentionality, emotional impact, and knowledge of reuse processes/domains. Readers will gain insight into (1) what agents should be included in the architecture and (2) effective settings of these agents' parameters as a function of user skill and agent purpose.

1) Introduction

Reusable software artifact repositories are an important asset for improving team productivity and product line quality. As a result, one can observe the "mainstreaming" of reusable software artifact repositories across the spectrum of Computer Aided Software Environments (CASEs) and fourth generation languages. From simple clip art and reusable script libraries of desktop applications, to the more complex domains of software parts reuse libraries managed by CASEs, the paradigm is shifting from programming-from-scratch to reusable-artifact-facilitated-program-editing: e.g., see Lowry (1992), Winograd (1995). This trend has demonstrated significant benefits. Yet to date it has relied upon direct manipulation and hypermedia interfaces to allow users to browse, surf, retrieve and reuse repository artifacts. As each of these repositories grow in size, they become too large for direct manipulation interfaces alone to support all of the needs for reuasable artifact identification, indexing, searching, transfering, transforming, adapting, and so on.

Fortunately, the technology is now here, in the form of intelligent agents, to help eliminate this bottleneck. Search agents are an early and widely used example of such technology. But search is not the only function acceptable for agent services. Besides, keyword based search is often an inefficient and frustrating experience. Specifically, the research questions of interest here are to explore:

- What services do repository users need from and want of intelligent, conversational agents?
- Do reuse repositories with suites of intelligent agents improve user productivity and performance relative to repositories with browse and surf (and search agent) designs? What agents should the suite include?
- What should be the settings of agent parameters such as degree and mode of intervention/control, intentionality and purpose, emotional impact, and knowledge of the reuse process and repository domain?

These are the types questions this research set out to investigate.

Since the late 1980s the lead author has directed four separate repository design and evaluation projects. This paper will examine lessons learned across these past projects in an attempt to discover what answers already exist to these questions vs. what directions further research should be aimed at. The remainder of this section presents: (1) an overview of the model of the reuse process that agents may need to be aware of (Section 1.1);

(2) background on intelligent agents and why further guidelines must be researched (Section 1.2); and (3) an overview of the four studies of intelligent agents that use this model to varying degrees to support repository users (Section 1.3). Section 2 then examines the results across the past repository efforts, while discussion of results are in Section 4.

1.1) A Human Information Processing View of Reuse-Supported Problem Solving

This research focuses on reuse-supported problem solving (RSPS) and the tools that facilitate it, particularly in groups. With the exception of discovery, most human problem solving involves some form of reuse of what is known and already developed. The light bulb existed before Edison, the car before Ford, and the desktop metaphor before Jobs. In following RSPS, professionals anchor on a standard, reuseable set of problem solving methods and solution strategies, and adjust them for each new situation. One can see this anchoring and adjusting in all aspects of the software field (e.g., requirements authoring, designing, code writing, test planning, and maintaining). The goal is to better understand these RSPS processes, so as to design tools that help assure the best anchors get selected, and the proper adjustments get performed.

This research views RSPS as spiraling through three basic processes: constructing, adjusting, and learning. The basic human information processing view places these into the following steps:

- **Information Acquisition Step** -- Initially, to get past the "blank page" one must create and construct. A draft problem statement and a solution are constructed, both with the help of appropriate reuse artifacts or anchors.
- **Information Processing Step** -- Once a draft is constructed, adjusting processes take hold. Partial failures of reuse artifacts are noticed and refinements to both the problem statement and solution are attempted.
- **Continuous Learning Step** -- For an organization to continually improve, it must capture the newly constructed and refined artifacts for future reuse. Decisions and tradeoffs that lead to the design of the artifact also need to be captured, as well as outcomes from the implementation of the artifact.

In general, spiraling through the first two of these steps may repeat for a given project until a quality, budgetary or time limit is reached. Then the third step needs to be accomplished in an ongoing manner. Silverman (1985, 1987) provide further detail on this model.

2) Background on Intelligent Agents for Reuse Processes

The motivation for the current paper arises from the lack of guidance in the literature on intelligent agents concerning agency for reuse-supported problem solving. Straightforward search and retrieval agents have received a fair amount of attention in digital libraries, and are widely used with the Internet and World Wide Web repositories: eg., see Fox et al. (1995). But this is a micro-task in the RSPS model.

Looking past digital libraries, the general literature shows that intelligent or conversational agents are computer programs that recognize organizational contexts, teamwork interaction needs, and humans' information management activities. They autonomously anticipate and carry out (low level) tasks that improve productivity in these settings. To the extent possible, they hide the details of these tasks from the users, interacting with users at the overall goal level. These agents may, at times, need to ask the user for specific instructions, explain how they're conducting a task, or ask for clarification. In general though, an intelligent agent should be able to determine intentions; manage unexpected changes in the conduct of its tasks; and adapt and learn over time: e.g., see Etzioni (1994), Sheth & Maes (1994).

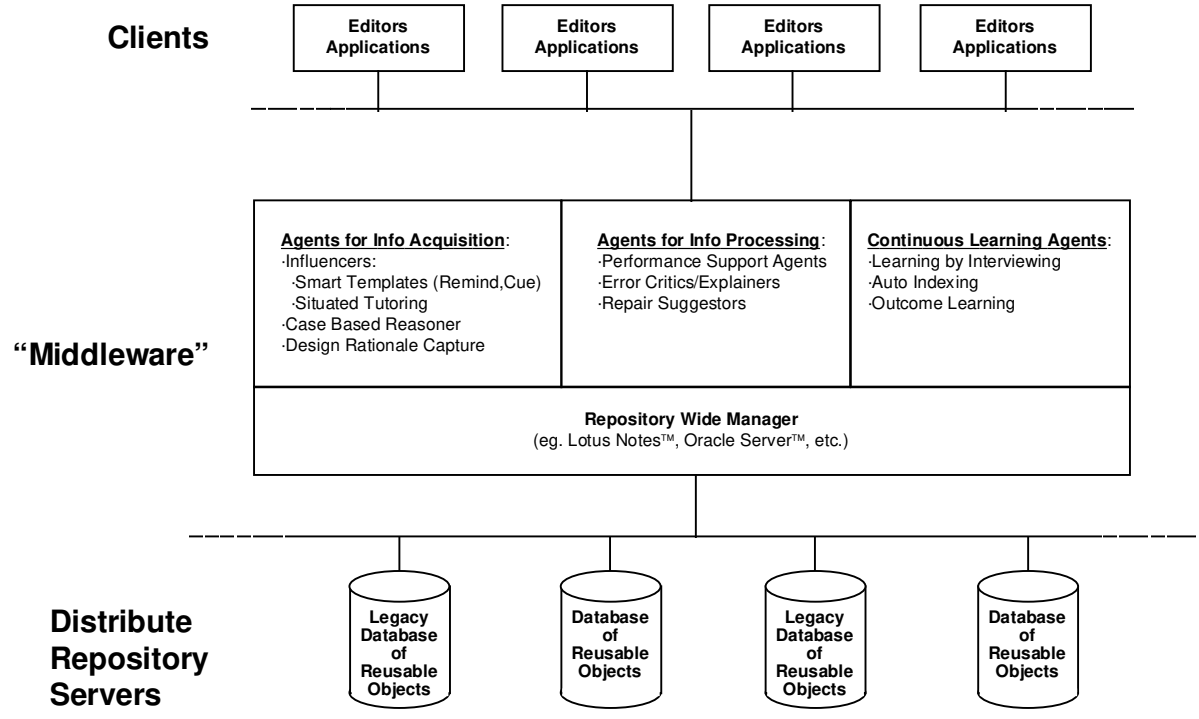
Some general purpose examples of agents' tasks are anticipating/carrying out simple information searching and handling tasks, and providing reminders, alerts, and coordination in the team-oriented setting. For instance, a group authoring environment such as Lotus Notes can include agents that variously route work to proper recipients, remind team members when document versions are due, and alert members when changes are added by others. Still other agents help a user keep track of team members and how to reach them. Similarly, a CASE environment performs similar "conversational" services for the programming team. A number of process-oriented, computer environments exist for cooperative work: eg., see Pitt et al (1994).

From the perspective of RSPS though, an important area of contribution lies in semantic services provided by yet other agents that are aware of the domain that the team is working on and that could act proactively. Thus, particularly as the group repository grows larger, there may be a need for agents that could anticipate users' needs, automatically suggest artifacts for reuse (reduce inappropriate anchoring without user-specified search parameters), and support users in adapting artifacts to make them more useful to the current context: e.g., , Silverman et al. (1989, 1994). Such agents might provide a range of services such as individualized daily news, knowledge-based editing, problem definitions, decision criticisms, solution suggestions, situated tutorings, and other support for the performance of the task itself.

2.1) Architecture of the Intelligent Repository

The difficulty is that there are too many choices. One needs to order and prioritize the list of potential agents that could support a repository. One way to do that is to model the human information processes and to focus research on agents that support the most important processes first. That is the reason for the RSPS model, and the design of the studies conducted here. From that perspective some of the major agent approaches of interest to RSPS include those listed in the middle of Figure 1 and as discussed below. Figure 1 shows a three tier client/server architecture. The server tier holds the repository artifacts in distributed databases, some of them legacy, some new. The client tier permits distributed users to access the repository artifacts and to bring them into their electronic workspaces, editors, and local applications. In most client/server architectures, the middle tier insulates the clients so that choice of platform and network access path (wired or wireless) becomes immaterial. The middle tier is widely defined to include lower level gateway/hub, security, and access services. From a repository view it also includes repository management software such as provided by Lotus Notes, Oracle Server, and so on. The architecture pursued here is a straightforward extension that adds a layer of RSPS-relevant intelligent agents to the middle layer, including the following groupings.

Fig. 1- Conceptual Architecture of Intelligent Agents for Reuse Repositories



Critiquing is a way of presenting a reasoned opinion about a *product* or *action* and in engaging in a dialogue about that opinion. It is analogous to human critiquing in real life. Critics are composed of rules or specialists that present "for/against" arguments and explanations for different issues on a given product or action the user has intitiated. They are suited for ill-structured and complex problem domains since their purpose is not to present a primary source or complete solution but to interact as a set of cooperative specialists working with the user towards a solution. One tends to associate a critic with the Information Processing or adjusting step of RSPS, yet critiquing can include:

Influencers -- These are strategies that prevent errors and inappropriate anchoring, shift mental models of the task, encourage good habits through smart templates, and provide on-demand, situated learning and tutoring. Influencers can also include during task cuing, reminding, and alerting.

Debiasers -- These are after-error feedback strategies that identify known faults, explain causes and effects, and offer repair suggestions.

Directors -- If errors still persist, these are performance support routines that can take control away from the user and carry out the task in a normative or prescriptive fashion. Generally, these will be deployed for safety-critical situations or where the underlying expertise model knows a normative or prescriptive solution.

Case-Based Reasoning (CBR) may be used as a low-intrusion agent with a number of the characteristics mentioned above and ideally suited to support constructive and learning steps: e.g., see Silverman et al. (1983), Kolodner (1993), Silverman, Chang & Feggos (1989). Basically, a CBR system examines the user's description (however incomplete) of the target problem, compares it with a repository of previous cases, and retrieves the most similar case(s). The similar cases are offered to support the user and as contrast to users' unaided anchoring judgments. A CBR-based agent also could contain an adaptation facility (or critic) for adjusting between the previous case to the current problem, or it could be left to the user to perform the refinement step. There are two added strengths to a CBR-based repository agent. First, it only needs partial knowledge of the domain, which is very useful in ill-structured domains such as design or estimation. Second, its performance improves over time as new (good and bad) cases are added to the repository.

Learning Methods -- Aside from the learning feature of the CBR approach just mentioned, there are several other learning agents that seem appropriate and that have been included here such as design rationale capture, learning by interviewing, and analytical learning after tracking and studying outcomes.

Early in this research we thought it would be important for many of the agents to operate simultaneously and opportunistically. So we created the middle tier as a blackboard architecture and added a Chaiman agent to coordinate and negotiate task assignments between the cooperating agents: e.g., see Silverman, Chang, & Feggos (1989). During the course of the research described in the next section, however, it became clear that most agents could work in sequence, or simultaneously for clearly constrained intervals. Further, the agents needed to perform much of their work on clients that were desktop PCs. As a result, there is now only a weak blackboard remaining in the architecture in the form of a working memory space that diverse agents write intermediate results to for operation on by subsequent agents. The chairman algorithm is similarly "gutted" and is little more than a terminal stay resident process. This Chair process gets embedded in client applications and it swaps agents in and out of memory in response to activity progress made (or direct manipulations) by a given user in the application. Activity progress is detected through the use of templates and scripts that users follow, and/or through the use of user toggled icons in the GUI.

2.2) Agent Parameters

Another way to examine the intelligent agent topic is to examine alternative settings for agent parameters. For example, intelligent agents must understand what aspect of a problem to support, when, and how to support the user: Silverman (1992). Three parameters that need be to considered when developing agent strategies include: Intrusiveness & Emotional Impact, Agent Primary Purpose, and Intervention Techniques.

AGENT PRIMARY PURPOSE -- During the early stages of development, it needs to be established if an agent's primary purpose is to assist Education or Performance. An educational agent's primary strategy is to maximize information retention and motivation; e.g., cut back on successive critiquing. A performance agent's primary strategy is to support construction and evaluate (and suggest repairs to) a product or an action

as a whole under user defined policies and/or criteria. For example, a Code Analyzing Critic could be instructed to critique for Efficiency rather than Readability.

INTENTIONALITY & EMOTIONAL IMPACT -- Intelligent agents must avoid both over-interference and insufficient help. Intentionality requires modeling users' preferences and goals so as to balance frequency of feedback, complexity and purpose of task, and cooperation with the user. User feelings about a computer as an intelligent assistant need to be considered. Users also need to be aware that computerized agents provide privacy. When these features are done well, users adopt agents and increase their dependency and trust.

INTERVENTION/CONTROL TECHNIQUES -- There are two techniques that agents could employ: Active and Passive Intervention. During active intervention, the agent retains control and interrupts at the appropriate time, hence allowing it to influence and respond to individual user actions. This technique is advantageous since support occurs when the problem context is still in the user's mind. However, timing is critical and every time support is offered, it potentially disrupts a user's cognitive processes. During passive intervention, the user invokes the agent when desired. The disadvantage of this technique is that the delayed messages generated by the agent could be out of context. Even worse, the user could have already gone down a "garden path".

KNOWLEDGE OF REUSE PROCESS AND DOMAINS -- Should agents have deep vs. shallow knowledge of the reuse process itself and of the specific domain of the repository in which they exist? Shallow or no knowledge implies the agents are unaware of each other, of the overall reuse process, and of domain specifics. The advantage of shallow or no knowledge lies in reduced cost and effort to develop and deploy the agents. The disadvantages may be that the agents are less intelligent, more automaton-like, intervene inappropriately, misunderstand intentions, and lose user trust.

3) Results from Studies

Table 1 provides an overview of the results from four studies of intelligent repositories. The set of intelligent agents of interest here are listed down the left side of Table 1. The four studies correspond to the next four columns. The results are contained in the body of the table and are discussed in the following four subsections, one for each study. After that Section 5 concludes this paper with a summary of cross-study findings.

Table 1 - Overview of Results from Studies of Intelligent Agents for Reuse Repositories

(a) Agents in the Architecture and User-Desired Agents

	Combat Simulator Repository	Plan/Report Generation Repository	College Courseware Repository	Software Project Repository
Agents for Info Acquisition:				
• Influencers:				
• Smart Templates (Remind,Cue)	Yes	Yes	Yes	Yes
• Situated Tutoring	No	Yes	Yes	Yes
• Case Based Reasoner	Yes	Yes	No (desired)	Yes
• Design Rationale Capture	No (desired)	Yes	Yes	No (desired)
Agents for Info Processing:				
• Performance Support Agent	Smart Editor	Embedd Agents	Smart Editor	Estimator Aid
• Error Critics/Explainers	Nonsimilarities	Plan errors	Programming	No
• Repair Suggestors	Yes	Yes	Yes	No
Continuous Learning Agents:				
• Learning by Interviewing	Yes (templates)	Yes	Yes,	Yes
• Auto Indexing	Yes by CBR	Yes by CBR	Manually	No (desired)
• Outcome Learning	No	Yes	Manual	No (desired)
			Yes	

Table 1 - Overview of Results from Studies of Intelligent Agents for Reuse Repositories (Cont'd)

(b) Agent Parameter Settings and Usage Results

	Combat Simulator Repository	Plan/Report Generation Repository	College Courseware Repository	Software Project Repository
Intervention/Control Options: Passive (P), Active(A)	Active, but user controllable	Active, some user control	Active, user controllable	Passive(Active desired)
Agents' Purpose: Education(E), Performance (P)	Performance	Education & Performance	Education & Performance	Performance
Intentionality: User Model(UM) Descriptive(D), Prescriptive(P)	UM Essential D, P	UM Preferred P	UM Preferred D, P	UM Preferred D
Process & Domain Knowledge: Shallow, Deep	Deep (Domain & Process)	Deep (Domain & Process)	Deep (Domain)	Deep (Domain)
Trust: System Accepted, Fielded	Accepted, Fielded	Accepted, Fielded	Accepted, Fielded	Accepted
Results: Statistical Improvement	Yes	Yes	Yes	Yes
Sample Size (N)	12	19	several dozen	33

3.1) An Intelligent Repository for Combat Simulators

In 1992, Silverman and Murray completed a study on factors affecting human-computer collaboration in a medium sized combat simulator repository (100s of reusable artifacts). Users in the domain must identify and reserve and coordinate the use of about 50 to 100 simulators and related test articles across numerous sites for each new military test exercise. Before the start of this study, the server level supported this activity with a legacy database holding information about the reusable aritfacts, sufficient to support reuse decisions (artifact characteristics), checking simulator availability over time, and making reservations for various reusable artifacts. The client level of the repository permitted remote access via ascii terminals and through a menu-driven interface.

This study attempted to follow a number of the architectural features and guidelines mentioned earlier. A combined client/middle tier was constructed to replace the ascii terminals with a graphical user interface and a set of intelligent agents. The GUI permitted users to depict the combat scenario they wanted to simulate by drawing it on a terrain map with the help of a map library, a set of draggable combatant icons. This activity was supported by a smart scenario generation template, and a conversational intelligent agent (case based reasoner) that collects the reusable components from the repository and/or engages in dialogues to help the users find suitable substitutes: see Murray et al. (1987), Murray (1989). The smart template consisted of a graphical lattice of the objects to include in a robust scenario. A library of past scenario templates serves as coherence influencers and completeness reminders to drive the anchoring and problem defintion processes. The template also supported question asking to fill in the data for the objects needed in the scenario. The case based reasoning agent, in turn, would use the data stored in the scenario objects as a statement of the "problem" to be solved. It used a similarity scoring routine to drive the search of the repository for reusable artifacts that are available during the needed time frame. User preferences for degree of similarity in the search process could be elicited via a slider icon for each parameter of interest. Alternatively, the users could set the CBR agent to verbose mode to interact during each artifact search and selection procedure.

In terms of the agent parameters, this approach permits variable degree of agent intervention (active question asking vs. direct manipulation of scenario lattices, plus terse vs. verbose CBR agent). User intentionality is supported via the introduction of the more natural (descriptive) interface metaphor plus the sliders for user preferences. In terms of depth of knowledge, the agents posses significant domain knowledge (CBR algorithm and heuristics are highly tailored to this domain) and process knowledge (smart template with prescritptive guidance). If successful, these features should lead to acceptance of the new design, and feelings of dependence and trust. In point of fact, the intelligent repository design was adopted by the repository organization, yet specific reactions and performance data could not be collected due to military classification once the system was fielded.

In lieu of that, a laboratory study was designed that compared reactions of 12 users in the original repository database browse and query mode to their reactions with the intelligent repository: e.g., see Silverman (1992), Murray (1989). The results show statistically significant improvements occur when repositories are agent-engineered well enough to: (1) replace standard menu interfaces with familiar metaphors; (2) replace unstructured artifact browsing and repository navigation with an ecological, whole-task-structuring interface (smart scenario template); and (3) substitute conversational, task-knowledgable agents support of the construction and adapting steps. Major importance for acceptance was ascribed to plasticity of control, user changable intervention capabilities, and

3.2) Intelligent Repository for Plan/Report Generation

The largest version of the intelligent agent architecture of Figure 1 we created to date supports plan/report generation for part of the materiel acquisition arm of the US Army: e.g., see Silverman (1991). This application supports Army personnel at 17 sites nationwide who must write hundreds of reports per year, one report for each new piece of materiel being acquired by the Army. Each report is fairly logical and well structured, almost like a knowledge base, and there are many hundreds of rules governing its content. A military standard and a training manual exist on how to write the reports. The problem is that the Army delegates the authoring task to low level officers who have technical expertise in the subject of the report (e.g., communications engineers, artillery experts, and bio-chemical weapons specialists), but who lack experience in the field where materiel must inter-operate, and be usable in adverse conditions. That is, the authors overly focus on personal, vivid knowledge about gadgetry, and overlook abstract information about operation, use, and support in the field. They would write completely unacceptable first draft reports, despite the availability of the standard, a training manual and course, and dozens of past "good" reports on any given area of materiel acquisition across the Army.

The architecture of Figure 1 was utilized to better support this authoring activity. Performance support was provided by a knowledge interviewing/question asking agent that walked the authors through each subsection of the report (as required by the military standard). Authoring report passages was further supported by a network of critics (influencers, tutors, feedback debiasers, online version of the manual, restricted language understander, etc.) consisting of 1,500 rules, 1,500 objects, and 2,000 notecards. The CBR agent, in turn, was seeded with reusable passages from more than 300 past reports, indexed in many instances down to the sentence level to foster cross-document reuse of well-written passages. The CBR agent was also designed to capture new reports and index their passages in the repository, once they were approved by the senior officers. Seventeen servers were designed to hold the reports generated at the centers of technical expertise, while clients were the Zenith Z-248 desktop platform. The entire system was designed as a front end for creating the first draft report. Afterwards, this was converted to an ascii version that could then be imported into the word processor of choice for adding publication grade formatting.

In terms of agent parameter setttings, this is an example of a application with a prescriptive model and deep knowledge of the domain and process. These items are utilized by each of the agents to provide maximum intervention, and fairly rigid control, to shift the users' mental model from technical to operational, and to reduce their errors. Agent purpose is situated, education and performance support, while intentionality is supported by a two-level user model -- newcomer and repeat author. Overall the system was accepted by the senior officers and fielded. In a field trial, 19 users found the system made authoring take longer on the first draft, and they initially resented the interventions and loss of control. In the end, however, their fustration was reduced (no senior officer lectures), they understood what to write the first time, and the overall document creation interval was reduced by several months due to the need for less rewriting.

3.3) Intelligent Repository for College Courseware

For several years the lead author has been teaching a college course entitled "Decision Support Systems." This course involves student-centered learning-by-doing and hands-on development with the teacher as facilitator-coach. The students learn to program in the critic shell and environment, called COPE, that has been mentioned as one of the agents in the Information Processing Step described earlier. They also collaborate in small teams to develop a prototype DSS using the COPE critic language and to conduct a user-evaluation study of their prototype. In early versions of this course the students had to learn COPE from teacher-lead

laboratory sessions and from hard-copy documentation. They then had to learn how to build their prototype and run an evaluation project from books and copies of past students' project reports the teacher kept in a library.

As summarized in the third column of Table 1, several changes were made to try and improve this situation and to offer a Computer Supported Collaborative Learning (CSCL) environment for the students: e.g., see Silverman (1995), Silverman, Motta, & Rosenberg (1994), Mehzer (1992). First dozens of influencer, debiaser, and director critics were given deep knowledge of programming errors and were embedded in the COPE language. These are intelligent critic agents that help the programmer avoid (cues, reminders, online help, situated tutors) and repair (explain, suggest, implement fixes) errors in their applications. These agents may be toggled to user specified level of intervention (incremental before-task, continuous during-task, batch after-task, and no critic). To support their project work, the CSCL environment included a multimedia library/repository of past students' final presentations (digital video of design rational), digitized and scanned past project reports and related materials, and past students' COPE applications. All the parts of this CSCL environment are united in a GUI that presents the course as a travel through space to visit 14 planets and conduct missions on those planets, one mission per week of the course. As they conduct their missions, the students encounter the COPE critics (Agents for Info Processing), and anchor on and adjust from the various repository artifacts.

At present there are no repository "Agents for Info Acquisition" as the repository holds under a dozen past projects, other than direct manipulation browse and access via the GUI's space travel metaphor. However, the third author of this paper is currently conducting a thesis to add a CBR agent that will elevate the project prototyping into primarily a transformational-programming exercise. That is, the future students will simply specify the design and goals for their application, and the repository's agents will select and adapt reusable artifacts into a first draft of their DSS application. This transformational agent is being designed to search any repository, not just this one, and to assist users in importing reusable artifacts into the application.

Two forms of evaluation of the current CSCL environment have been conducted: overall assessment and critic evaluation. The overall assessment of the DSS course was 2.9 out of 4.0 by 13 students who took that course prior to the deployment of the CSCL. Since the CSCL has been added, the course assessment has risen to 3.5 out of 4.0 averaged over 47 subsequent enrollees: See Silverman (1995). Since the instructor and syllabus remain unchanged, the improvement in the course evaluation is attributed to the CSCL reducing the students' effort and frustration in conquering the COPE language and in designing and completing their DSS application projects. Thus in Table 1b, the "results" row reflects statistical improvement.

The other assessment study was conducted at a more microscopic level to determine subjects' performance with and reactions to the network of critics for the COPE language: see Mehzer (1992). These results are also summarized in the third column of Table 1b. A study was conducted of 37 course graduates performing programming tasks. The control group of 12 students had no critics, while 25 students performed the tasks with critic features available. The results show that, with critics, errors were eliminated and time to reach a correct outcome was decreased (by severalfold). However, 10% of the subjects dislike critics and turn them off, while another 20% of the subjects used them but think they program better without them. All subjects approve of the critics supporting an Education purpose for newcomers, and then switching to less active intervention and more Performance support as skill increases. A user model that tracks skill attainment and modulates critic intervention is preferred (at present there are only 4 skill stereotypes, and these are not individualized to specific skill capabilities).

3.4) Intelligent Repository for Software Project Estimation

Software project managers around the world make thousands of software cost and effort estimates a day using variations of the RSPS process. This is analogical reasoning from completed, similar component estimates: e.g., see Silverman 1983, 1985. That is most software project managers reuse prior estimates for analogical components that they are aware of. To support such efforts most project managers save past estimates for future reuse, or they obtain them from trusted colleagues who recently had to complete such an estimate. These saved estimates constitute informal repositories that exist in electronic spreadsheets, gantt charts, and documents on various managers' personal computers. As organizations migrate to CASE tools and groupware (e.g., Lotus' Notes), these repository artifacts are increasingly available to a wider set of users.

To better support this estimation effort, under NASA's sponsorship we designed and built a protoype estimation repository with intelligent agents as listed in Table 1a (last column) and as now will be further described: e.g., see Silverman, & Bedewi (1993), Bedewi (1995). The repository is managed by a piece of groupware (Lotus Notes) that lets distributed user clients share a server's reuse artifacts. This component also supports the repository's primary client user interface, e.g. task descriptions (hypothetical projects) are presented to the subjects through this interface, along with the Q&A for the new project problem descriptions and attributes' values. Subjects also provide their project estimates through this interface. Through this client interface, the users may directly access the reuse artifacts and the various media these artifacts come in, or that help explain details about an artifact (derivation, assumptions, purpose, etc.).

In the middle there are two intelligent agents built in Esteem. Esteem is an off-the-shelf case based reasoning tool that was embedded to test the intelligent agent features of interest in the repository. It is used to hold the domain structural knowledge (attributes), pointers to the experimental domain's reusable cases, the functionality for changing the case's attribute weights, a rule base for estimate adjustment, and the capability to "launch" other applications. Basically, Notes sends a DDE[1]message containing target case information to Esteem whenever a subject presses a button for retrieving previous cases, weight adjustment or estimation adjustment. In response, Esteem provides its own user interface for the appropriate function (button) selected. Esteem's strengths are two fold. First, it numerically ranks the problem descriptions of previous cases for similarity to a new target problem based on attribute matching. The user can manipulate a number of attribute parameters that impact the search result, however, due to time constraints they were only exposed to changing the attributes' importance (weight) in the trials described below. Second, is its capability to invoke a rule base for authoring added conversational agent support. For this research, the rule base was developed as the "estimation assistant" to adjust the cost & effort of the selected "best" matching previous case to the target case. This is a further example of including deep knowledge about the domain in the architecture.

An evaluation of this architecture was conducted using 33 professional project managers/leaders who volunteered from NASA, AT&T, MCI, and related organizations: e.g., see Bedewi (1995). The primary purpose of the evaluation was to assess the role of multimedia for the design of reuse artifacts, however, a number of insights about the intelligent agents are also possible. Specifically, the repository was seeded with cases from the sorting algorithm domain. Each case contained an extensive number of artifacts covering past estimates (text, tabular, graphical), sorting program component attributes (lines of code, languge, computer, and so on), and a rich set of multimedia on the derivation and rationale of each past component estimate. The subjects each performed two hypothetical component estimation tasks, one without the repository and one with the repository and all its agents.

The results (see final column of Table 1b) include several interesting insights. First, even though sorting algorithms are of the size (and familiarity) that most managers could intuitively estimate, only 42% of the subjects felt they had sufficient information for the task without the repository, as compared to 79% with the repository. Similarly, subjects who thought their estimates would be met rose from 42% to 82% when the repository was added. At the same time their mean group cost estimates rose by 62% and the standard deviation in the group estimate dropped in half. These numbers indicate the subjects are being given strong anchors, ones they trust, as well as estimation assistance they find acceptable (they don't adjust in different directions). Indeed post-usage questionnaires and debriefing results reveal 91% of subjects feel the repository improved their estimates, saved them time, and supported their acquisition and processing efforts. Out of dozens of media-artifacts they encountered, they consistently rated the acquistion and estimate adjustment agents the most important and useful features (system help was also important). In terms of the CBR agent, they valued its prioritization of past projects, but wanted an interface that let them display and compare several past project estimates simultaneously. In terms of the estimation assistant, they felt it was useful at the start of the estimate adjusting process (to get them going), but then they ignored its final recommendations and relied on their own judgment in the end. The lessons learned here give insight about how information should be organized, agent vs. human control of tasks, and what should be allowed to intrude, when.

[1]Dynamic Data Exchange

Overall, the subjects accepted the design. One surprise was their significant appetite for media-artifacts. For example, subjects indicated emotional impact is important and that the previous projects' artifacts should be personalized as much as possible. This includes:

1) Capturing "body language" on video,
2) Capturing lessons learned throughout the project cycle with the original voice "tones",
3) Capturing project participants' original comments, decision rationale, and issues throughout the project life cycle.

This personalized information was often cited as the most vivid encountered by the subjects when accessing previous cases while performing the project estimate. Subjects complained that the quality of the video presented should be improved and that a couple of the audio clips were hard to hear.

These types of comments will prove challenging for an organization attempting to capture personalized information mentioned above, to improve emotional impact, and to create comprehensive yet concise clips. Given that this information will be captured "dynamically" during the life of ongoing projects and by "amateurs," balancing between artifact content, quality, and production cost may be an obstacle not easily overcome. Further research would be useful here to develop templates of and procedures for the ongoing creation of such media-artifacts by project personnel. Results to date from the design rationale community may contribute to this effort.

5.0 Discussion of Results & Lessons Learned

It seems possible to generalize two sets of findings from the results: those concerning the architecture and those concerning the intelligent agent parameter settings. The architecture was created around the needs suggested by a human information processing model of reuse supported problem solving (RSPS). This model provided a strategic view of the range of agents needed. These model findings were incorporated into the architecture by including agents to support information acquisition (smart templates for cuing, reminding, situated tutoring, searching, case based reasoning, anchoring, and so on), information processing (performance support, artifact transformation, adjustment aids, etc.) and continuous learning (by interviewing, rationale capture, automatic indexing, outcome tracking, etc.). All repositories that tried the resulting intelligent agent architecture exhibited significant increases in user performance as well as user satisfaction. One can conclude this is a robust architecture for intelligent agents in repostiory applications. It is a cost effective upgrade for and extension to browse and surf (and search agent) repository interfaces. One can also infer that in general, for any class of agent applications, using a human information processing model is an effective way to help decide what intelligent agents are needed during architecture development.

In terms of lessons learned about agent parameter settings, it is clear that having the right collection of agents in the architecture is necessary, but not sufficient to guarantee success. Significant effort needs to be expended to give these agents the behaviors that users want and expect. In particular, this research looked at acceptable and unacceptable behavior in terms of four parameters: knowledge of reuse processes and domains, degree of intervention and control, purpose and intentionality, and emotional impact. We discuss some lessons learned about each of these in turn.

In terms of knowledge of reuse processes and domain, at least one thing is clear. That is, that adding repository domain knowledge to the agents seems to be universally called for. For example, the CBR agent needs a domain model for its indexing function, plus its info acquisition functions had to be tuned to analyze and conduct dialogues about (1) low level attribute similarities in the combat simulator repository, (2) text parsing in the plan generator repository, and (3) high-level, cross-case, tabular comparisons in the project management repository. Likewise the influencers, smart templates, situated tutors, feedback critics, repair suggestors, and rationale capture agents all were rated more highly when they included expertise (albeit narrow) in the specific domain of the repository. Giving agents deep knowledge and awareness of the reuse process and of other agents is less clearly called for. For example, the interagent cooperation appears fairly straightforward in repository applications, and an intially sophisticated "blackboard cooperation model and chariman" was all but eliminated from the architecture. Also, most agents in the last two repositories (courseware and software estimation) omitted deep process knowledge, yet these agents were still used and accepted by most of the users. The agents in the first two repositories, by contrast, incorporated process

knowledge, could reason about what step the user was trying to take next, and could offer anticipatory and prescriptive support (influencers, reminders, and so on). Deciding whether or not to give an agent process knowledge is clearly tied to some of the other parameter topics about to be discussed.

In terms of parameters affecting agent intervention and control, when one has a significant descriptive or prescriptive model of the process/domain, and of where humans do and don't perform well, it is easier to see where agents should interrupt, take control, and provide support. Where this knowledge is absent, agents should generally be either passive (user started and controlled) or highly plastic and flexible in their interventions. For example, the agents in the estimation repository are only activated by user button selection, and their suggestions/findings are easily overridden. The value of this design philosophy was realized in this repository. Specifically, the project estimators enjoyed the luxury of intervention by the estimation aid at the start of that task, but they took over control once the agent had got them going. Even though they felt the agent was naive, they liked its intervention to get them started. Our descriptive model of this domain never lead us to expect this usage of the estimation agent. Fortunately we had designed it so users could override its advice.

By contrast, active intervention and rigid control requires significant process as well as domain knowledge (and intentionality sharing -- see below). For example, descriptive studies and interviewing of the simulator repository users showed they thought about the problem at the level of the battlefield map and in concrete terms. They were quite happy to have agents interpret their goals and take over much of the remaining activity. High degrees of agent intervention occurred during scenario construction, and agents exhibited significant control over the reusable artifact list adjustment and substitution step as well. Agent intervention and control still had to be reduced though for seasoned officers (higher domain and process knowledge) who also were computer literate users. As a second example, it was only in the planning repository, where we had the most prescriptive model of the domain/process, that aggressive intervention and rigid control were used. Here the senior officers had insisted the agents be designed to correct repetitive errors (of selective perception, of inappropriate anchoring and adjustment, of overconfidence, and of inadequate training) on the part of the junior officers who were the authors of the first drafts of the reports.

In terms of parameters affecting agent purpose and intentionality, as with the other parameters, there are few universal truths and many nuances to consider. The repositories studied here ranged in purpose across the spectrum from pure education (college courseware repository), to mixed on-the-job education and performance support (planning repository), to pure performance support (estimation and simulator repositories). In some regards, one would expect the education based repositories to have to expend more effort on modeling user skill progress, and in getting users to "verbalize" and think about their own goals and intentions. The plan report genreation repository had only two skill level settings, but did engage in much of the latter types of dialogue, aided as it was by the well developed prescriptive model of the domain/process, and of commonly occuring user errors. There are many dialogue sequences that elicit and help correct users' goals and intentions in the course of using the agents of this repository. These sequences tend to be time consuming and unpopular with the novice users, but they are proven to lead to dramatic performance increases (report quality, rewrite time reduction). Besides, they are mandated by the senior officers who must sign off on the reports in the end. The other educational repository primarily follows a student-centered discovery pedagogy and, thus far, it has few intention eliciting or sharing agents. The one area where it includes user skill level models is in the critics of the COPE programming language. Here the use of only four, rigid skill stereotypes was a source of dissatisfaction to many users, yet these critics demonstrated major programmer performance improvements. In summary, one interpretation of all this may be that intention sharing is difficult to do well in the eyes of education-oriented repository users, but performance improvements can be attained anyways.

Turning to performance-oriented repositories, one might expect the agents to have to be more adept at sharing, explaining, and being willing to alter their own intentions, even though they may need to do this infrequently. That is, when users are more proficient, one would expect they are less easily lead and more demanding for goal information from their collaborators. Many of our agents are totally unable to accomodate such dialogues. For example the agents in the estimation domain that are easily and often overriden. The case based reasoning agent in the simulator repository is probably the closest to meeting this expectation. That agent can be switched at will between "terse" and "verbose" modes. In terse mode it works autonomously, attempting to interpret scenario goals and trying to meet them with the most similar artifacts it can locate within the time and attribute constraints. When switched to verbose mode, it displays the subgoals and resoning paths it is pursuing and asks users if they want to alter them before each individual artifact reuse decision. This relatively

rich interchange is possible because the goal structure and attributes of the domain are well-understood (scenario templates, complete search problem attribute descriptions). It would be difficult, though not impossible, to extend this approach to a search agent in an unlimited repository, such as say the world wide web. The effort would only be worthwhile if the domain problem descriptions were to be used repeatedly.

In terms of parameters affecting agent acceptance, trust, and emotional impact, the most surprising results encountered in these studies were probably the extent of detail and media that users want from repositories. The estimation and courseware repository users want to see the body language and hear voices of the previous artifact creators' explaining design rationales, and the plan generator repository users would benefit from hearing the intonation and disappointment of their supevisors. If repositories of the future are to capture and retain such levels of detail, this poses high performance computing challenges for agents in every step of the reuse process -- acquisition, processing, and continual learning. As the telecommunications, video and computing fields continue to merge, the repository users may find the artifacts they want most are indeed in the repositories. The problem facing repository agent developers is how to use content-based search to retrieve only the most relevant clip segments, and to support the transformations and links to other media that are need for reuse supported problem solving.

References

Bedewi, N., Judgmental Forecasting in Software Projects: Exploration of Design Guidelines for Multimedia, Reusable Artifacts to Support Analogical Reasoning Heuristics, Washington DC: GWU/IAI Dissertation, submitted to University Microfilms, Inc., 1995.

Bloor, R., "Repository Technology," DBMS, v._, n._, Dec. 1991, pp. 17-23.

Etzioni, O., Weld, D., "A Softbot Based Interface to the Internet," CACM, v.37, n.7, July 1994, pp. 72-75.

Gennari, J.H., Tu, S.W., et al., "Mapping Domains to Methods in Support of Reuse," Int. J. Human Computer Studies, v. 41, 1994, pp. 399-424.

Gomaa, H., "Reusable Software Requirements and Architectures for Families of Systems," J. of Systems and Software, (accepted for publ.), 1994

Hodges , B., "Reuse Driven Software Process (RSP) - A Product Line Approach to Reuse," in The Navy/ STARS Demonstration Experience Report, Crystal City: US Navy (STARS CDRL A017R), 1995.

Lowry, M.R., "Software Engineering in the Twenty First Century," AI Magazine, v.14, n.3, Fall 1992, pp. 71-87.

Mehzer, T., Toward a Theory of Critiquing Human Errors in Computer Programming, GWU/IAI Dissertation, avail. form University Microfilms, Inc., Sep. 1992.

Murray, A., A User Oriented Approach for Assessing Multiple Paradigm Knowledge Engineering Practices, GWU/IAI Dissertation, avail. form University Microfilms, Inc., Nov. 1988.

Murray, A. Feggos, K. Weinstein, M., Silverman, B.G., "A Lisp Machine-Based Scenario Generator for the INNOVATOR Expert-System", in Military Applications of AI, ORSA, 1987, pp. 90-110.

Nabkel, J., Shafrir, E., "Blazing the Trail: Design Considerations for Interactive Information Pioneers," SIGCHI Bulletin, v.27, n.1, Jan. 1995, pp. 45-54.

Pitt, J. Anderton, M., Cunningham, J., "Autonomous Agents in Inter-Organization Project Management," In Intelligent Information Agents Workshop Papers, CIKM'94.

Seel, N.R., “Agents in the Human-Computer Interface,” IEE Colloquium on Intelligent Agents, London: IEE, 1991, pp. 1-5.

Sheth, B. Maes, P., “Evolving Agents for Personalized Information Filtering,” Proc. of Ninth Conf. on Artificial Intelligence Applications, Los Alamitos: IEEE Computer Society Press, 1993.

Silverman, B.G., “Computer Supported Collaborative Learning,” Computers & Education Journal, (accepted for publication), 1995.

Silverman, B.G., "Intelligent Electronic Collaborators for Mirror Worlds," in Distributed Collaboration Technologies for Software Engineering, ARPA Workshop Proceedings, Arlington: ARPA/SISTO, May 1994.

_____, Critiquing Human Error: A Knowledge Based Human-Computer Collaborative Approach, London: Academic Press, 1992a.

_____, “Human-Computer Collaboration,” HCI Journal, v.7, pp.165-196, 1992b

_____, "Criticism Based Knowledge Acquisition for Document Generation", Innovative Applications of AI, Menlo Park: AAAI Press, July 1991, pp. 291-319.

_____, "Software Cost and Productivity Improvements: An Analogical View," Computer Magazine, v.18, n.5, May 1985, 1985, pp. 86-96.

Silverman, B.G., Bedewi, N. E., "Toward the Development of a Rule Base for Critiquing Project Management Errors During Software Development," Expert Critiquing Systems Research Workshop Proc., Menlo Park: AAAI/MIT Press, 1993.

Silverman, B.G., Chang, J., Feggos, K., "Blackboard System Generator: An Alternative Distributed Problem Solving Paradigm", IEEE Trans. on Systems, Man, and Cybernetics, Mar. 1989, n. 19, n. 2, pp.334-355.

"INNOVATOR: Representations and Heuristics of Inventor/Engineers" Chapter 17 of Building Expert Systems for Business, Addison-Wesley, 1987, pp. 402-39 (with V. Moustakis).

Silverman, B.G., Moustakis, V., Liebowitz, J., Bokhari, I., Tsolakis, A.,
"Resource Planning by Analogy: The SCAT Support System" in Microcomputers: Tools or Toys?, ACM/NBS, June 1983.

Silverman, B.G., Motta, C., Rosenberg, M., “The Trekkie Environment for Computerized Collaborative Learning,” CSCW’94 Poster, avail from GWU/IAI, 1994.

Winograd, T., “From Programming Environments to Environments for Designing,” CACM, v.38, n.6, June 1995, pp. 65-74.