

Creating Value Under Uncertainty¹

Adele Goldberg, Neometron, Inc.

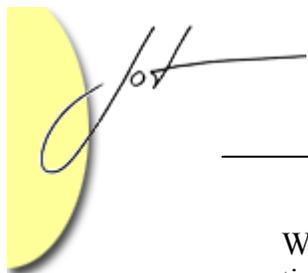
Abstract

Two disciplines—decision analysis (sometimes called value analysis) and scenario planning—provide a level of rigor to the task of looking into the future. Both combine information available in the present with explicit assumptions, interpretations, and consequences. Decision analysis models consequences of actions, identifying where uncertainty divides single events into multiple options, each having some likelihood of occurrence. Scenario planning is a form of science fiction writing that gives a gestalt view of the future evolved from the present under a model of how environment, technology, economics, and politics may change. Scenario planning is used to produce stories that identify patterns of behavior. Decision analysis applies mathematics to quantify sequences of choices, such as cost tradeoffs, and to compute the future value of assets created by following any particular sequence.

Why are these disciplines of interest in software engineering? First, they offer tools for modeling assumptions and expectations. They provide ways to construct plans in situations dominated by uncertainty. Second, they provide a framework for anticipating the consequence of selecting a next step. And third is that they offer techniques for designing collaborative technologies to support project planning. These same techniques are currently being used to design online support for secondary school teacher professional development and for drug development programs.

1 INTRODUCTION

I bring to my professional career an interest in two areas: the use of computers in education, and an interest in understanding how to help both individuals and teams work more productively through the use of computer-based communications and collaboration tools. These are not independent interests. Productive work is most often carried out in a learning context, and teams—of authors as well as software engineers—are needed to create and deploy effective uses of computers in education. I have worked on projects that relate to one or both of these interests in two contrasting situations: basic research and commercial product development. In each case, two issues always arise. The first is the personalities of the team members, and the second is development planning in the face of considerable uncertainty. Where these projects have failed miserably is when there was a mismatch between the job to be done and the personality of the software developers hired to do the job.



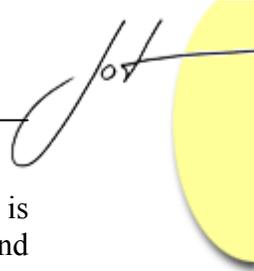
What is the difference between a researcher and a product developer? I pose this question not as the start of a joke, but rather as a serious inquiry into patterns of behavior and expectation. While both personalities seek to create value for their organizations, they do so with different tolerance for working in the face of uncertainty. While both personalities understand that each project offers an opportunity to learn, they do so with even greater differences in their willingness to work without clear specifications. Yet both researcher and product developer want to know in advance whether their actions will result in progress, regardless as to whether the target outcome is well specified. Both plan their actions, if only because they need to coordinate with collaborators.

The particular solution to planning under uncertainty that I describe is a part of the research around “model-based collaboration” [Goldberg, 2002]. The goal is to handle uncertain knowledge in a rigorous way when creating development plans. The idea is to apply the tools of decision analysis, in which models of the work (work process and task sequencing) are combined with models of the context of the work to form dependency maps and inference diagrams. Since the actual tasks and sequences cannot be fully determined, alternative pathways must be proposed. Decision analysts assign some measure of the likelihood of success for each of the alternatives, and also assign a value that could be attained as a consequence of succeeding with each alternative. These two numbers—likelihood of success and value produced—are determined by considering the target market for the development outcome. Market factors that contribute to the decision modeling include market size and competition, cost of the work, resource availability, and barriers to future competition. The bit of mathematics involved in this analysis produces an expected net present value (NPV) for any particular pathway. Optimizing NPV is traded off against likelihood of success. It is also important to look at the particulars of the pathway and consider whether the future glimpse into the history of the project makes sense as a whole. Scenario analysis thus complements the deconstructive approach of decision analysis.

Scenario analysis has a larger role to play. Scenario analysis, or story telling, is a technique whereby various assumptions are taken to their logical conclusion at some future date. The conclusion is described by telling a story in prose of the future situation. We can use the techniques around eliciting assumptions and constructing stories in order to design the knowledge structures that should be created and evolved during the course of the project. We also use such techniques to identify where and when it would be important to have project team members contribute data or commentary that would be used, in the future, to generate such a story.

2 RESEARCH OR PRODUCT DEVELOPMENT

At one point in my career, I was primarily a researcher and a manager of researchers. I then transitioned to become the head of a company seeking to introduce new methodology and technology into the market of software development tools. The transition was a leap of faith on the part of my investors, since the core development team was made up of lifelong researchers, such as myself. Why should there be any concern?



Research begins with a question—one that may never be answered. Success is measured by demonstrating progress towards an answer and by the discovery and publication of new and preferably novel ideas. Research measures success by papers published and speaking invitations received, which often translates into research grants received. Software engineering is a legitimate area of research, although its answers are often in forms that appear much like the outcomes of commercial efforts and, as such, confuse how we determine the value of the research. The software engineering researchers demonstrate their ideas in the form of actual software processes and process support. They demonstrate their prowess by distributing their software and counting the number of active users. Indeed, we look back at software engineering research milestones not so much in terms of the powerful ideas—for which there are many notables—but in terms of distributed results that ultimately changed how we view the materials of our work, including programming languages, tools, development and maintenance methodologies, algorithms, architectural patterns, and the standards we adopt to improve our ability to share results.

Research, by its nature, cannot require specific results on a particular timeline. Nor does it require that a market price be set, with a target market size that nets target margins. In fact, negative results are of interest, although not often reported.

Basic research is about the right to fail.

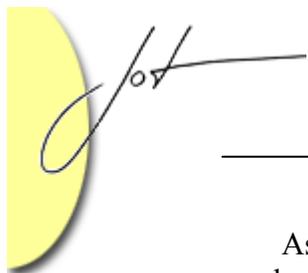
Starting a company is a presumption of success.

Basic researchers explore in the space of unknowns. It creates value by demonstrating that a theory to explain phenomena is correct, or not. Basic computer research creates value by inventing new ways to enable better processes, better productivity, or better science, in fields that benefit, for example, from computation, data management, or data mining. Research resulting in new communications technologies and collaborative work environments strikes at all three targets.

What basic research is not required to do is create a result that sells in the marketplace, nor to do so within some budget and timeframe (with the exception that one can view the research granting agencies as a marketplace). Software engineering research, of course, includes studies of how to improve our ability to create outcomes that sell in the marketplace, and that complete within some budget and timeframe. But the studies do not have to be successful to constitute good research.

Another personality distinction is that the researcher, once a solution direction is understood and appears feasible, concludes that the hard part is done and often loses interest—it is just a simple matter of programming. In contrast, the product developer knows that the hard part is ahead.

The product developer works in a more constrained situation. Faced with the same development uncertainties as the researcher and, more to the point, the same desires to offer creative solutions that capture market attention, the product developer is nonetheless expected to accurately predict costs in terms of time and resources required. In the ideal situation, the product developer is expected to set up and work within well-defined deterministic projects, with small amounts of uncertainty.



Ask a product development manager and team about their schedule for a new product release. The product developers I have worked with generally responded, “can’t answer the question; we don’t know. We have never done this before so there is no basis for answering.” Or, having done it before, they still say, “don’t know”, and for the most part they are answering accurately because every situation is indeed a little different.

I learned to take such a response as an opportunity, and to retaliate with my own simple question: “what don’t you know?”

I will avoid here the usual debate among the marketers and engineers, in which they sort out definitions of requirements and focus only where the goals can be made clear, broken into quantifiable and time-targeted objectives. The product developers quickly learn that their first step is to ask for such specifications. They deal with what looks like territory minutiae, but which results in a negotiated vocabulary that permits some semblance of productive interaction.

In contrast, basic researchers prefer ambiguity in order to retain both flexibility in the range of exploration, and flexibility in the ability to claim innovation—to be able to stumble onto a new discovery regardless of original focus. (The dilemma faced by many commercial managers is that many of the product engineers feel the same way!)

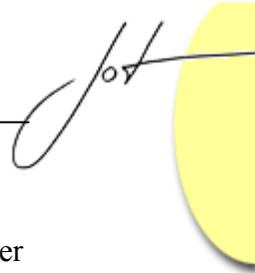
I eventually learned to divide product development planning into three tiers, tied to a requirements inventory:

- Tier 1. Features that have to be included to make the result worthy of market attention. You cannot ship without these.
- Tier 2. A prioritization of features that should be included if there is sufficient time. Usually some pet desire of an engineer is included.
- Tier 3. Features that will not be included and for which the marketers should stop asking.

The uncertainty of development requires that we treat time as the enemy. Time, and not the actual feature set, defines completion. So time is the primary known planning factor. The rest provides considerable discomfort since the thoughtful product developer will inevitably say: “I do not know.”

What is unknown and do we have viable ways to plan development in the face of such uncertainty? Among the unknowns in software engineering projects are:

- Whether the desired result can be built at all.
- Whether the result, when built, will in fact solve the problem—in particular, be accepted in the target market.
- Which architecture best meets quality objectives (including maintenance expectations).
- Which development tools will allow the team to work effectively and productively.
- Whether development team members have sufficient skills, in the domain or using the selected tools.
- What whims of the developers might defocus an agreed-to pathway.



- Whether the feature set will remain competitive at the time of delivery.
- Whether the specification leads to a result that creates a sufficient business barrier to competition.

The unknowns are compounded when the development team is formed from employees of different organizations, either inside or outside of the funding organization. Differences in development culture prevail: the nature and form of gate-keeping, requirements for documentation, testing style, tool preferences, and reward structure. All of these impact how communications, coordination, and issues management can be successfully handled. Add to the problem space that these team members remain physically within their organizations, work at significant geographical distances, and are often unaccustomed to pro-active reporting.

Web-based solutions abound today and seem to have some positive impact.

I can find a general interest group using tools similar to or the same as what I am using, and get advice in a timely way.

I can find and download examples that, assuming I am sophisticated enough to be willing “to read” in order “to learn”, both teaches and gives me code I can incorporate and modify.

When I cannot solve a problem, I have a worldwide community to which I can direct my questions and get answers.

Although of help, these web-based solutions are individual in nature. They do not tackle the larger questions around geographically dispersed teams: establishing and maintaining focus, identifying and resolving issues in advance of a crisis, planning work incrementally in the face of acknowledged unknowns, and doing so while maintaining awareness of work schedule and instilling confidence in those charged with management oversight. Management confidence can be improved by making economic and social assumptions explicit through decision analysis and scenario planning. Our particular interest, however, is to move these techniques much lower down—to tactical development planning.

3 A QUICK LOOK AT TWO DOMAINS

I will tell two stories to illustrate the ideas. Each of these examples is part of a larger effort to understand how to design online communities to support project teamwork, what in past writings I have called *project communities*. It shares its goals with CSCW efforts generally, but its approach is more akin to those researchers using simulation nets as a tool for planning projects, in the sense that there is a model of the work that permits an evaluation (or simulation) to produce some future view of progress. The model that forms the basis for community building is the same one used to construct scenarios and is the core of the model to be used by the decision analysis. In this way, there is some discipline in gathering information used by the decision analysts.

Most of the prior work that I have uncovered derives from the most basic contribution to planning, PERT networks, and is seen in an early effort called GERT (Graphical Evaluation and Review Technique [Swindle *et al*, 1979]). There is also prior work in using decision analysis to select directions for software investment (acquisition and development), but not at the detailed level of incremental development planning.

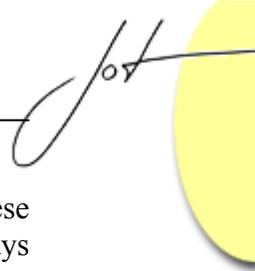
Historically, duration for a task is either deterministic (a single number presumably from experience) or probabilistic (weighted average of three times: worst, average, best; or most pessimistic, most likely, most optimistic). GERT was a probabilistic network analysis technique in which it was possible to maintain parallel development activities, leading to a probabilistic decision point that, in turn, directed the future course of development. The work on GERT was motivated by a desire to incorporate statistical uncertainty of activity durations, since such uncertainty alters the outcome of a critical path analysis and was proving to underestimate mean time to completion and to overestimate variance, affecting expenditure planning estimates. These concerns were also addressed in Barry Boehm's COCOMO estimation techniques, although COCOMO is more consistent with a particular model of software development than with a general theory of probabilistic modeling. (The classic reference is Boehm, 1981, but of course the early COCOMO efforts have been extended in COCOMO II and COCOPRO. Perhaps more interesting to the context of this paper is Boehm's more recent "model-based architecting" efforts. See the [MBASE web site](#).) The success of the COCOMO family of work points to the need for such explicit modeling before any computational approach to estimation can be considered.

Authoring and Delivering Web-Based Professional Development Services

The ThinkFive®² project is researching the efficacy and broad application of a set of Internet-delivered tools that together constitute an authoring, publishing, and learning management system. The tools were designed to support, in the United States, the teaching and assessment of learning in calculus and statistics for advanced high school students and the professional development of their teachers. The tools are the heart of a service focused on building the capacity of schools and school systems to expand and diversify enrollment in rigorous college-preparatory courses.

There are a number of commercial learning management systems currently on the market. These systems were designed as communications (e-mail and threaded discussion) and information sharing systems for corporate and university campuses. They create virtual extensions of a university classroom in which teachers and students can engage in asynchronous dialogue, view slide presentations often redundant with classroom lectures, exchange assignments, and conduct tutorials.

In the U.S. K-12 world, learning management systems are relatively new and are marketed as tools for diverse purposes ranging from direct administrative support to classroom teachers to supporting more efficient communications between administrators, teachers, parents, and students. Perhaps because these systems seek broad applicability, their designers have chosen not to emphasize or focus on instructional design.



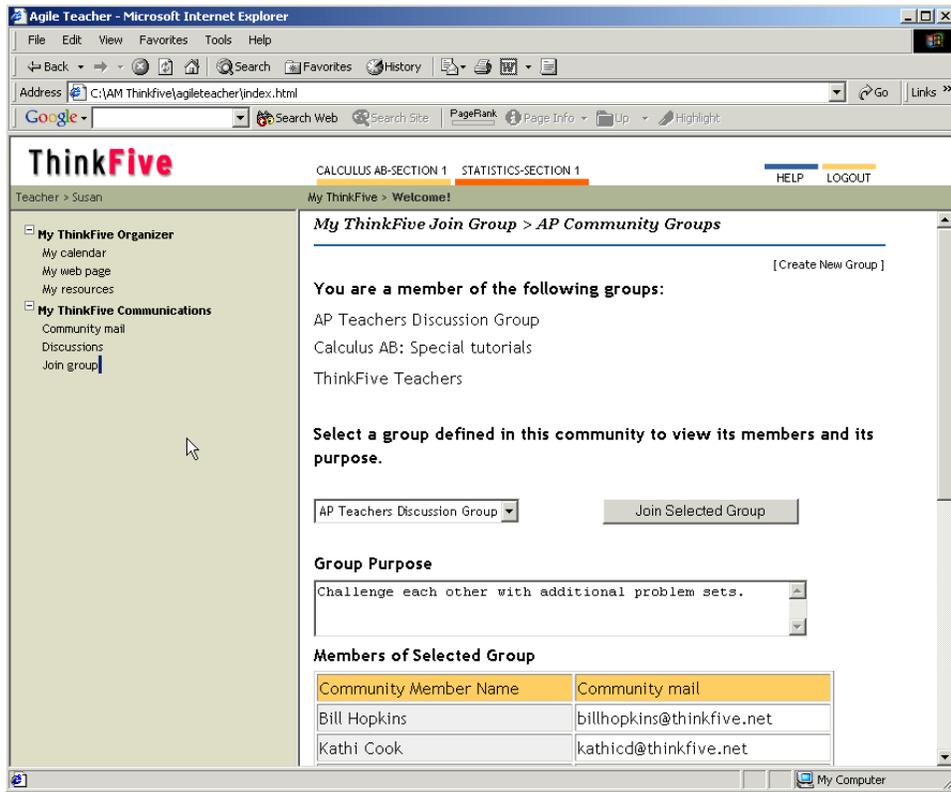
Unlike their computer-assisted instruction counterparts of the 1970s, these commercial systems do not draw on fundamental research in learning theory for the ways in which they support content development and delivery, behavior tracking and reporting, and, perhaps, most important, teacher support. In contrast, ThinkFive tools are designed to support teaching and learning in ways that reflect modern learning theory. The content management system—for authoring, reviewing, editing, and delivering learning material—is built to guarantee substantive and appropriate content availability for both students and teachers.

Central to the content management system is a content specification vocabulary that allows all of the content to be flexibly represented in XML. The tools include a software generator that interprets the XML content files and generates an Internet website. The generality of this approach allows the same XML files to be used to generate the students' website, as well as a different teacher-based website that embeds the student material in the context of annotation, lesson plans, teaching tips, and answers and solutions to test practice.

The tools are innovative in the way in which they embed the content delivery in the context of an Internet community. Like members of many such communities, students and teachers have personal websites, within-the-community mail exchange, and conversation support. Scheduling, assignments, and syllabus planning are among the several productivity features for the teachers. But, unlike services for other communities, the existence of an explicit model for instruction allows us to provide specific assistance to the teacher in understanding and acting on student interaction—for example, knowing which students are progressing rapidly and might be challenged with additional assignments; knowing which students are struggling, with which topics, and why; and determining how students might be organized into study groups (where the ability to flexibly create subgroups for assignments or discussions is a core capability of the software on which the system is built). The first image below shows the “Join Group” page of the LMS.

Critical to the ThinkFive mission, the explicit model of instruction provides a basis for evaluating system usage and effectiveness. The model represents our hypothesis of how best to enhance the teaching and learning experience. The behavior tracking and data analysis capabilities that we put in place will allow us to both test this hypothesis and improve it.

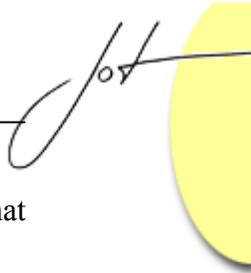
The second image below is a sample screen from the Statistics course, showing an animation in the topic Correlation. The third screen is from Calculus, and shows an assessment exercise from the topic on Related Rates.



One of the Community Communications Pages: Join Group

The development plan for ThinkFive is challenged by Time, the developer's primary enemy. The project, although almost 2 years in conceptualization and fund raising, was just funded at the end of March. School started mid August. In those four-and-half months, the following development had to be accomplished:

1. The first semester for two courses, Statistics and Calculus, had to be fully authored, reviewed, edited, and the media assets (including animations) had to be completed. At the start of funding, only a little more than half the material had been authored.
2. An authoring system, designed for XML editing, content and media tracking, and task assignment, had to be created and deployed in time for useful application. It had to reflect the way in which authors, editors, reviewers, and producers work together to prepare quality content according to the constraints of the instructional design.
3. The XML representation of the curriculum content includes mathematical expressions written using LaTeX, and online exercises that call on Flash MX players with changing content. This representation is interpreted by a software system that generates the content html pages. It will be rewritten (in truth several times) to keep pace with changing requirements for innovative uses of media



- assets, and the usual discoveries about how truly hard it is to create websites that run on the many machines and browsers that schools use.
4. A Learning Management System (LMS) for enrollment, syllabus planning, assignment management, community mail, discussions, and behavior tracking and reporting on student scores and problem areas had to be fully designed and implemented. The LMS is really a community of practice for both the students and the teachers, as it supports forming mentoring or study groups, and provides community teachers with access to teachers in other schools across the country to share teaching strategies
 5. The XML represents annotations and extensions to the core content targeted to the professional development of the teachers. These include notes, lesson plans, teaching tips, and special services such as video tutorials, essays, and live expert consultations. Additional products had to be identified and integrated into the LMS to support these services.

An Animation from the Statistics Course

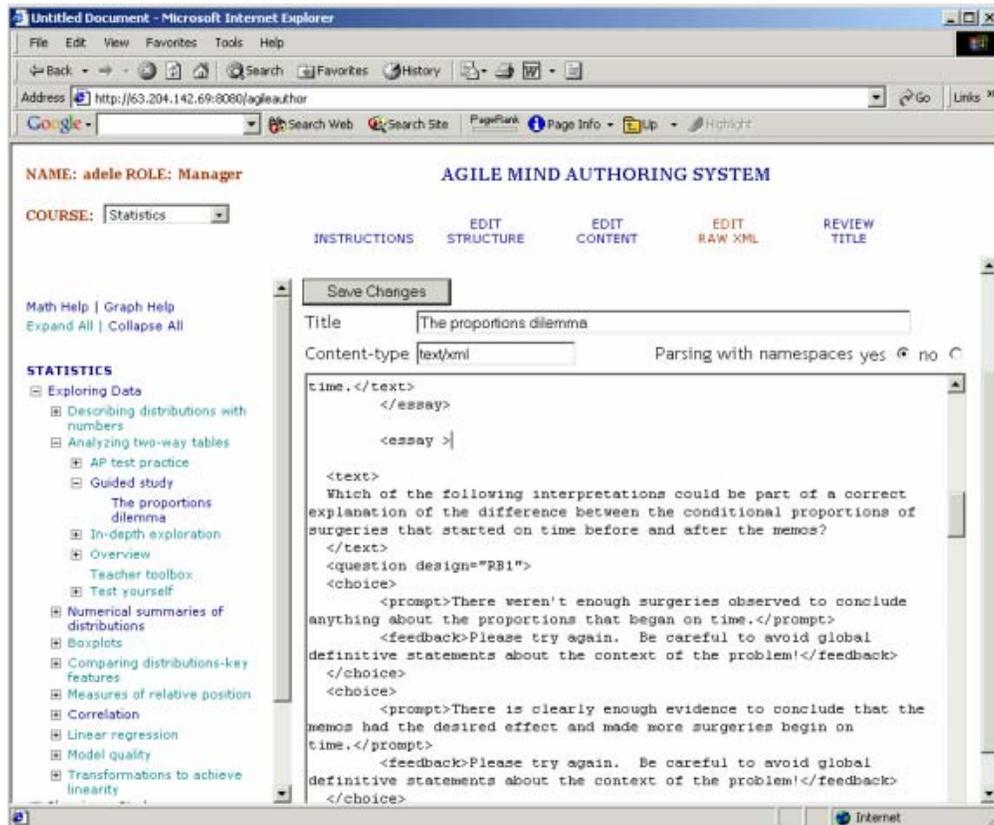
The primary good news in this otherwise ambitious enterprise was that there are only two developers (the author and Dennis Allison) for the authoring system, website generator,

and LMS. The implementation was done in ZOPE (an open source system based on Python [Latteier and Pelletier, 2002]), for which they had no prior experience. There was therefore no engineering confusion around organizing a team, and full local control over any expansions beyond the Tier 1 feature set. As a result, there was, after the available four months, a fully functioning LMS (as shown in the first images), a functioning authoring system to handle the XML and content development tracking (shown in the next two images), a completed and fully documented XML curriculum language, and a completed generator. In truth the latter two were mostly done before the project formally started but have been and will be continually revised.

The screenshot shows a web browser window titled "Agile Learner - Microsoft Internet Explorer". The address bar shows the URL: `C:\AM ThinkFive\agileteacher\calculus\topics\topic33\topic_index.html`. The page is titled "ThinkFive" and displays a navigation menu on the left with categories like "Course Professional Services", "Course Topics", and "Integrals". The main content area is titled "Leaking cone problem" and contains a text-based problem description. Below the text is an interactive diagram of an inverted right cone tank with a radius of 6 feet and a height of 8 feet. The diagram includes a water level indicator and a rate of change of $-1/6$ ft/min. At the bottom, there are input fields for the derivative $\frac{dv}{dt}$, the rate of change $\frac{dh}{dt}$, and the height h .

A Page from the Calculus Course: An Interactive Assessment Exercise

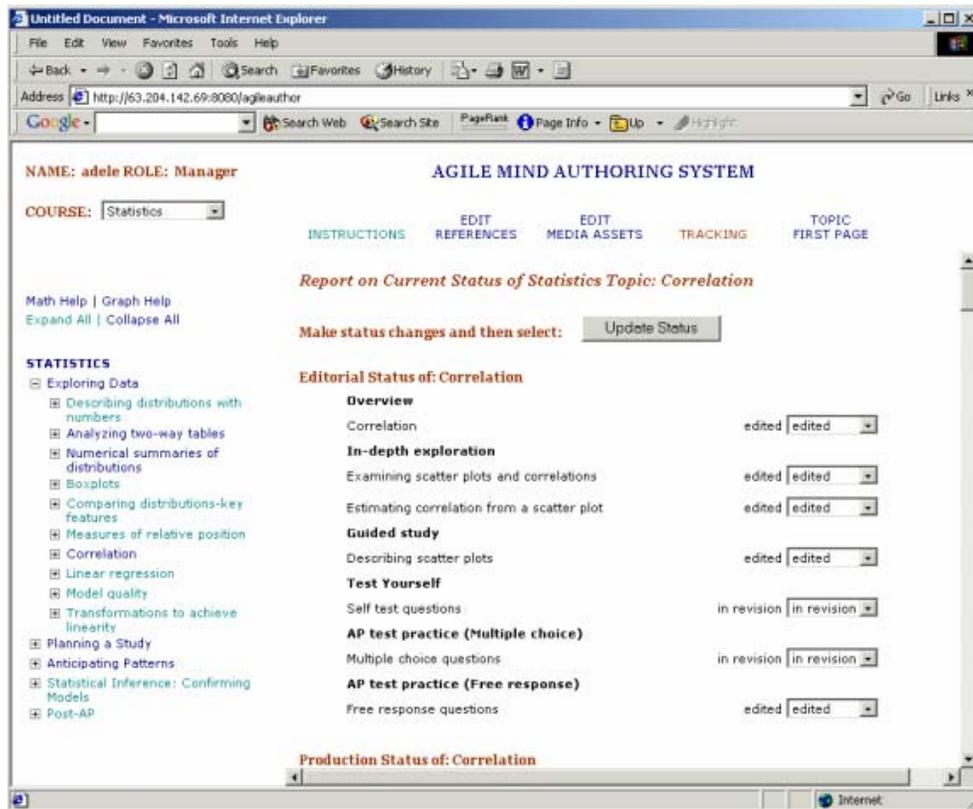
Uncertainty lies in the quality of the LMS solution of course. But the real uncertainty comes because the content development team is entirely composed of contract workers—authors, editors, reviewers, artists, and animators. All work is done via telecommuting, so there is considerable reliance on using technology for coordination and collaboration support—starting with the telephone, but also using a voice over IP solution that includes whiteboard functionality, a Wiki (which seems to be only of interest to the engineers), and the authoring system itself.



An XML Editing Page from the Authoring System

The rapid progress experienced was fundamentally feasible because the instructional design itself provided a foundation for a project model, and the basis for the vocabulary in the communications with development team members who live in more than four different parts of the United States. This project model served as the basis for team assignments, and measuring progress towards the completion targets. There was no uncertainty on the day the system went live in over 50 schools; there was considerable uncertainty as to whether the various parts would come together as envisioned. The incremental staging and delivery of each part attempted to allay concerns, but could not and did not remove the obvious risks.

The story told to the market was well-received, but the value created remains a matter of future data analysis. As additional resources become available to the project, it is simply not clear where those resources can be deployed to best meet subsequent milestones for delivering content and professional development services, and new LMS functionality. Without some level of formal analysis, the balance between development planning and business planning will not be understood other than intuitively. Where luck reigns, intuition might suffice, but it is a risk.



An Editorial and Media Asset Tracking Page from the Authoring System

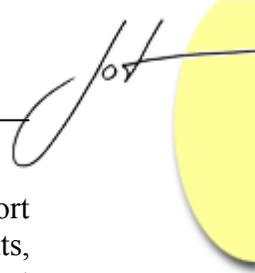
Managing Drug Development Programs

The second domain example has less development time pressure but not less ambition. The goal is to improve the process by which drugs are developed, from first indication to delivering information about the drug to the regulatory agencies. The specific project management features of this process are:

1. The information about the resulting marketable drug must be packaged according to regulatory requirements for a new drug application.
2. A targeted management scoreboard must continually present an evaluation of both completion status and progress based on an agreed to set of objectives and measures.
3. A complete history of the drug development program must be created that can be interactively queried and used to immediately transfer knowledge about the drug.

The containers or knowledge structure for these three outcomes are created at the outset of the project and continuously updated as a direct result of the process by which team members collaborate. As a result, the true status of activities, plans, and milestones should be consistently and persistently available to team members as appropriate.

The drug development task is inherently complex. Complexity is apparent in the need to manage large amounts of continually and dynamically created information, to



build both economic and scientific models that inform decision analysis, and to support knowledge acquisition and sharing among scientists, medical practitioners, patients, government regulators, and project management teams. These requirements are about technical complexity. There is of course additional complexity raised by socio-political concerns.

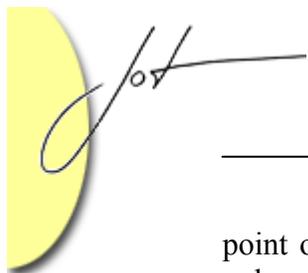
Learning must take place both intellectually and practically. Intellectually, collaborators each represent an area of cognitive skill, but together they must carry out coherent and responsible development planning and execution. What works and what does not work in support of such collaboration—whether it be communication tools, knowledge structuring applications, content management, or sophisticated decision analysis techniques—must be observed and preserved and shared with future teams. Practically, actions beget results that, in turn, have to affect the decision as to how to next act. Practically, each action has to be determined from a rigorous view of the whole development program as well as how the program fits into economic expectations for the program stakeholders. How can the distributed cognition of a team of professionals be made explicit and shareable so that actions are carried out efficiently and effectively, while continually learning so as to improve for future actions?

The answer potentially lies in scenario analysis and decision modeling. Much of the answer reported here is due to ideas worked on with Greg Hamm and Radomir Julina, colleagues at a new pharmaceutical company (Pharmaceutix).

The first step is to apply scenario analysis to design a target knowledge structure that can represent the evolving thinking with regard to the best positioning for the drug in the marketplace. The content of this knowledge structure will ultimately be translated into the formal label that will be presented to the regulatory agency. The knowledge structure is to a large extent simple, in that it holds possible labels, the messages in the labels that have to be proven, the actions that could or should be taken to construct that proof (including drug trials), and the knowledge garnered in carrying out these actions. The complexity comes from interpretation of this knowledge source, in that a single message can appear in multiple potential labels, actions can serve to support the claims of multiple messages, and the history of the work behind knowledge acquired has to be maintained and remain accessible long after the work is completed. It also comes from the large numbers of message claims, managing the data from trials and other efforts to prove the claims, providing access to the data when new claims ideas are postured, and doing so while maintaining several label options until a determination can be made that one is preferred.

Decision analysis contributes to the specifics of the development planning. Development planning involves determining, out of the many potential labels and messages with which to construct these labels, which actions to pursue. The incremental objective is to determine what is not known, determine what is optimal to learn next, and be able to predict, based on expected value of the learning effort, whether the attainable value is worth the effort.

Where are the uncertainties in drug development programs? Certainly, first and foremost is the question of the safety and efficacy of the drug itself. For a safe drug, there is still the question of what is the best indication to pursue from a therapy and a market



point of view (i.e., what problem should the drug be marketed to solve?) A number of unknowns about the drug exist and are studied in formal trials. But there are development risks aside from the drug itself. Logistics is one risk (can the drug be manufactured with sufficient and cost effective yield? will the supply be available at the times needed in the trials? will the study investigators obtain the study subjects at the right times?) Accuracy is another risk (will all the data and decision rationale be properly obtained and recorded and accessible?) Market positioning is yet another risk (since multiple labels are possible, the one with the best marketability is the one that needs to be pursued).

Development planning is made more interesting in that it has to take into account a number of factors, most of which have only estimated values associated with them. In the following list of factors, a “label” is an approved use of the drug.

1. Which label is best to pursue (provable and of market interest).
2. Which label the regulator is likely to accept.
3. Depending on who does the investigations, cost for learning (i.e., doing the trials or research).
4. Time and costs of the actual development work (which is often done by outsourcing partners, some of whom conduct the drug trials, and others who manufacture the drug and its delivery form; there may be as many as 12-15 different partners involved in a full drug development program).
5. Availability of candidate patients for the trials.
6. Whether the trials will be conducted cleanly.
7. Whether a trial will show up a serious drug side effect.
8. Whether logistically limited resources for a trial will be available, on time and in the quantity required.
9. What to do as result of current trials.
10. Timing of competitive drugs on the target market.

Traditionally, planning is done using Gantt charting and is based primarily on the experience and intuitions of the planners. Drug development suffers from the same problems as software development—the need to reuse data and results, the need to reuse patterns of behavior, and the need to reuse knowledge about worker effectiveness.

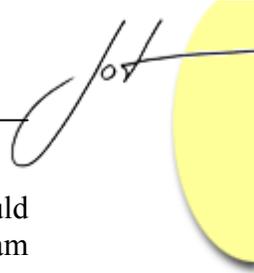
And Time is still the enemy—in this case, it is important to get done fast enough to maximize use of a drug patent, and it is important to discover early whether the best bet is to kill a drug without wasting additional resources.

4 A GENERAL FRAMEWORK FOR ONLINE PROJECT SUPPORT

The framework for thinking about development planning and community building has been documented elsewhere (see for example [Goldberg, 2002]).

The three basic tenets of model-based collaboration are:

1. **Work must proceed from a clear statement of goals and objectives, where objectives are time-based and quantifiable.** The skill required to design a model-based collaboration is good question asking—formulating those questions



- whose answers define the desired future story. The vision told in the story should indicate how collected data will be interpreted, and should justify to team members their taking the effort to provide the data.
2. **Work must be structured to reflect the structure of the knowledge to be acquired and shared.** The implications here are that we start modeling from the perspective of desired outcomes, and work backwards towards appropriate segmenting of these outcomes and the activities needed to produce each segment. The goal is a knowledge-flow model of work, not instead of, but as an integral part of a work process.
 3. **Workers have to be able to express opinions about how they work.** The requirement is therefore to provide the ability to customize the model dynamically. What we actually do not know is whether giving team members the ability to collaborate on such customization improves their willingness and agility at using online collaboration tools. Research in this area is ongoing, with Alejandro Fernandez and Joerg Haake, colleagues of mine in Germany. [Fernandez *et al*, 2002].

Development Planning

Nothing can save us from the requirement to produce plans and allocate tasks. All we accomplish by taking a knowledge flow point of view is to emphasize what we value—that is, creating the target outcomes according to clearly stated objectives for the product, processes, and resources, and structuring our collaboration in terms of knowledge acquisition and sharing. We still have to determine how and when knowledge will be acquired, and what dependencies might exist in the ability to acquire the various segments of knowledge. This is the moral equivalent of producing activities with start, duration, and end times, with explicit sequencing, much the way workflow analysts would produce a work breakdown, milestones, and interdependencies.

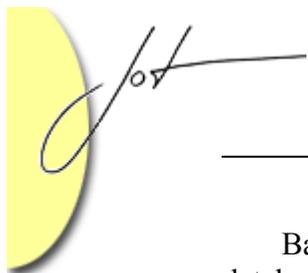
The distinction we wish to make, however, is that we can do development planning for ill-defined non-deterministic projects. And we expect to do so when there is considerable uncertainty in planning parameters such as duration and resource availability. These particular uncertainties are prevalent in both example domains.

The two development planning tasks we need to address are:

- to select what to work on, and
- to remove unknowns.

The order is a bit of a chicken-and-egg problem. We know that our product engineers said that they cannot provide a plan because they have never accomplished our particular set of objectives and therefore do not know what they will discover. In particular, to the exacerbation of any manager, they will say that because they do not know what will happen, they really cannot answer my question, which was “what don’t you know?”

So I learned a frustrating lesson. The question I really pose is, “What are your current thoughts on what you need to know?”



Based on an initial answer—architecture, tools, interface feasibility, efficient database design, and so on—our planning task translates into a careful assessment of what we need to know, that in knowing is calculated (via decision analysis techniques) to create value of interest to the organization and to produce the final stage of deterministic activities. Knowing what we need to know allows us to plan development according to a set of knowledge-seeking activities, and turns our planning into a future picture reminiscent of those produced by scenario analysts.

Knowledge Outcome

Model-based collaboration has one additional value—it produces a history of the work. That some form of history of actions and acquired knowledge must be produced should be obvious from the requirement to be able to learn from use and to apply that learning to improving the collaboration/project model.

Maintaining a useful history is a nontrivial idea. Many systems maintain records of online conversations and ask users to categorize conversation fragments including information used in making decisions. These personal and group information managers are often quite useful during the lifetime of a stable project, for the actual project participants. They fail at being useful by project outsiders, usually because the structure of the knowledge being collected is not itself well designed nor documented, and therefore not accessible. But they also fail to be useful because the terminology used for classification and for discourse is not well defined. Outsiders simply cannot find information in the historical recordings. There are two possible reasons: they do not have a mental model of how to browse, and no tools exist to explain the knowledge models.

5 RETURNING TO THE TWO DOMAIN EXAMPLES

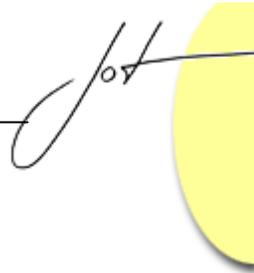
Let's now return to the two examples given at the outset and see a possible development planning and support solution for each.

Modeling Preparation of Web-Based Professional Development Services

Why is this area a good candidate for the model-based collaboration ideas? First, most of the work is outsourced—art production, animation, editing, reviewing, and authoring. The core team consists of business management, editorial leadership, and core engineering who live in four different cities. Some form of online communication and information tracking is required. Second, given time pressure, not everything can be done and decision-making needs to be carried out with a larger view of the value to be created. We previously discussed the uncertainty in software development. There is also uncertainty in the order, quantity, and timing of authored, edited, and reviewed materials, with regard to availability of editors and media producers.

Basically, the questions we need to ask are:

What is the project model and can it be used to define the structures under which collaboration take place?



- Is there a natural measure for success and for progress?
- Is there a special vocabulary?
- Is there a central and defining knowledge structure that must be constructed?
- Is there a natural flow of knowledge that can be defined and assigned for access to roles and activities?
- Is there a knowledge outcome?
- Is there an opportunity to learn?
- Will scenario analysis help in understanding the application of the project model?
- Can decision modeling assist in focusing and planning?

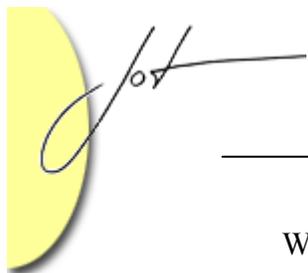
In this example, the project model is the instructional design model. It clearly defines the resulting product, but also defines the controls that have to be in place for coordinating the authors, editors, reviewers, and media producers. The authoring system being built (only a part of the system was completed for the first phase of work) fully reflects the instructional design, as does the learning management system. Both also have clear access control based on roles, to tracking input and output, content input and content flow into the publishing part of the learning management system. Some of the project vocabulary is standard, but the particular labeling of the instructional elements are new and need to be understood by all participants.

By designing a very targeted authoring system that reflects this design, we were able to turn content development over to the various responsible parties, and in parallel continue working on new features to increase the automation applied to the process. Scenario analysis was the driving technique to determine these designs. Decision modeling has not been used simply because time did not permit thinking before acting (a fact that necessarily added to, and continues to add to management angst).

Modeling Drug Development Programs

Why is this area a good candidate for the model-based collaboration ideas? As so much about a successful drug development program is delivering the right information to the regulators about the right label choice, the project model for drug development needs to be about the evolving design of the label and the decisions around design changes. The scenario analysis focuses on the story that has to be presented to the regulatory agency: what are the claims, how are they supported, who was involved in creating the support, what issues arose and how were they handled? Managing drug development strategically is possible because there are many outsourcing choices for both clinical research and manufacturing. The decision analysis has to provide the following evaluations:

- Which are the likely labels, where a label is a set of messages?
- Which messages are required to create a label with sufficient market value?
- What is the likelihood that any particular message can be supported?
- What is the likelihood that the regulatory agency will accept any particular message?
- Given the possible alternative ways to create support for a message, what is the likelihood that the regulatory agency will accept any particular approach?



What is the cost of obtaining the support for a message, given alternative clinical research partners and manufacturing partners?

What is the market climate for delivering a label, which contributes to the potential market value?

What is the barrier to competition (timing of the compound's patent, timing of entry for competition)?

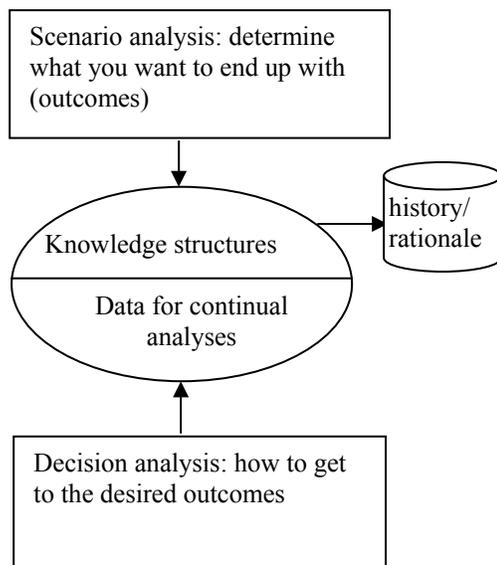
The project model then is a particularly rich representation of labels, claims for messages, potential actions to prove claims, and history of actions that turn claims into messages to be included or eliminated. All development works towards the focused goal of selecting the best label, completing the proof of those claims that comprise a marketable result, and maintaining the history for effective submission to the regulatory agency.

The decision analysis handles the complex evaluation of how to select the questions to be answered by clinical trials and which trials provide the best pathway to removing those unknowns that are barriers to selecting the label and messages.

A subset of the completed model is then deliverable to the regulators and other interested parties as a browsable source of knowledge about the drug. Moreover, models that are abandoned are still accessible as sources of reusable knowledge as the company makes decisions about future drug selection and development.

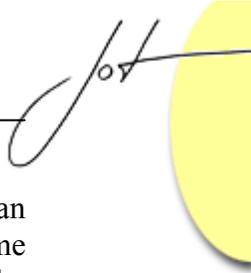
6 CONCLUSION

The figure here summarizes our proposed method (without showing the full project modeling or execution process).



In order to do the value analysis, (1) the decision analyst needs to determine the value of an outcome, (2) each outcome must be well-defined in the sense that there is a clear relationship between the probability of an outcome and the change in the probability that having the outcome will be accepted by the targeted recipient (i.e., a measure of whether obtaining the outcome adds value), and (3) an ordering that denotes the dependencies among tasks leading to outcomes and the probability that time and/or resources are available to do these tasks in the designated order.

The idea is simple. So if it is so good, does that mean everyone is doing it? For example, are the big pharmaceutical companies at least buying into the decision analysis part? They are, but only sporadically and only as additional input to the decision making process. The reason these techniques are



generally not a part of core processes is complex, having to do of course with an entrenched culture, large number of business units that contribute ultimately to the same drug submission result but that otherwise act independently. The companies lack the infrastructure for collecting information in a disciplined way that is accessible to the decision analyst. Most well-run projects of course have at least some database-backed issue management system. We can look to the keys and relations of the database to see some of the implicit knowledge model for the project.

The role of the decision analyst is to assist the developers in planning under uncertainty—to decide whether an investment should be made to pay for new information, and which source of information (which trial in the case of drug development) is the best choice given the expected uncertainties in the outcome. The incorporation of decision modeling into a knowledge management approach to collaboration provides a rigorous technique with which to select what to do and to select what not to do based on getting to an NPV with acceptable risk. In the end, a company needs to have a clear understanding of why development followed the pathways it did, both not to repeat failures as well as to improve the ability to succeed.

REFERENCES

Barry Boehm, *Software Engineering Economics*, Prentice Hall, 1981.

Alejandro Fernandez, Joerg Haake, and Adele Goldberg. Tailoring Group Work, *Proceedings of the 8th Annual Workshop on Groupware*, La Serena, Chile, September, 2002.

Adele Goldberg. *Collaborative Software Engineering*, in *Journal of Object Technology*, vol. 1, no. 1, May 2002, pages 1-19, http://www.jot.fm/issues/issue_2002_05/column1

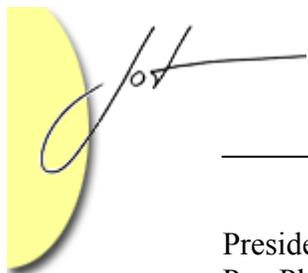
Amos Latteier and Michel Pelletier, *The Zope Book*, New Riders Press, 2002.

D. W. Swindle, E. H. Gift, F. M. Bustamante. Utilization of probabilistic network analysis in planning long-range engineering projects, *Proceedings of the 11th Conference on Winter Simulation*, Volume 2, December 1979.

About the author



Dr. Adele Goldberg is a director of Neometron, Inc. Neometron is a California-based consultancy working towards the use of virtual communities to support more effective teamwork. (<http://www.neometron.com>). Previously, she was the Chairman of the Board and a Founder of ParcPlace Systems, Inc. She served as CEO and



President from inception until 1991, and Chairman until 1995. Prior to the creation of ParcPlace, Dr. Goldberg received a Ph.D. in Information Science from the University of Chicago and spent 14 years as researcher and laboratory manager at the Xerox Palo Alto Research Center. From 1984-1986, she served as president of the ACM, the U.S. computer professional society.

¹ This paper is a revision from a lecture in the Distinguished Lecture Series at ETH, Zurich, Switzerland, June, 2002.

² ThinkFive is a registered trademark of AgileMind, Inc.