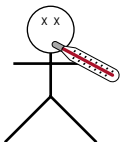
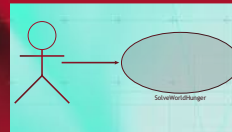


ALEX E. BELL, THE BOEING COMPANY

The Institute of Infectious Diseases has recently published research confirming that the many and varied strains of UML Fever¹ continue to spread worldwide, indiscriminately infecting software analysts, engineers, and managers alike. One of the fever's most serious side effects has been observed to be a significant increase in both the cost and duration of developing software products. This increase is largely attributable to a decrease in productivity resulting from fever-stricken individuals investing time and effort in activities that are of little or no value to producing deliverable products. For example, afflictedees of Open Loop Fever continue to create UML (Unified Modeling Language) diagrams for unknown stakeholders. Victims of Comfort Zone Fever remain glued in the modeling space, postponing





Acknowledgment is only the first step toward recovery from this potentially devastating affliction.

Fever

Diagnosis and Recovery

the development of software. And those suffering from Gnat's Eyebrow Fever continue creating models that glorify each and every Boolean value of prospective software implementations.

Research has shown that the failure to recognize or act upon UML Fever affliction is largely the result of factors such as denial, desperation, or a poor understanding of its symptoms. One of this article's primary objectives is to help overcome this failure by describing the fever's most commonly observed symptoms for the purpose of facilitating its diagnosis at both individual and organizational levels. Beyond promoting recovery in those actually stricken, this article is also focused on the codependents that perpetuate UML Fever in their organizations by ignoring its symptoms and failing to take corrective



UML Fever Diagnosis and Recovery

action. It is important to understand that individuals in leadership positions who allow UML Fever-related atrocities to occur on their watches are equally responsible for the fever's devastating effects as those actually stricken.

While afflicttees and enablers of UML Fever may once have had limited recourse in confronting their problems other than to do so in response to "hitting rock bottom" (program cancellation, perhaps), this is no longer the case. This article enables much earlier diagnosis of the fever as the result of providing symptom descriptions as never before documented. While new prospects for recovery may be encouraging, they are only within reach of those who admit to either having or enabling UML Fever and subsequently commit themselves to a corrective course of action. Even with admission and commitment, the road to recovery is not an easy one. It is pocked with temptation, anger, doubt, and isolation, and there are no guarantees of restored technical health. The likelihood of recovering from UML Fever, however, is greatly improved by committing to and following the 12-step recovery program premiering in this article.

SYMPTOMS OF UML FEVER

Any hope of recovering from UML Fever must be preceded by a recognition of its symptoms in oneself or one's organization. This important step is possible, however, only if the fever's symptoms are clearly described and readily available to those who have taken the bold step of entertaining the slightest possibility of a possible problem. A number of the UML Fever's primary symptoms are described in this section with the specific intent of enabling the diagnosis that is so vital to begin recovery. To avoid launching widespread UML witch hunts, however, it is important to emphasize that these symptoms

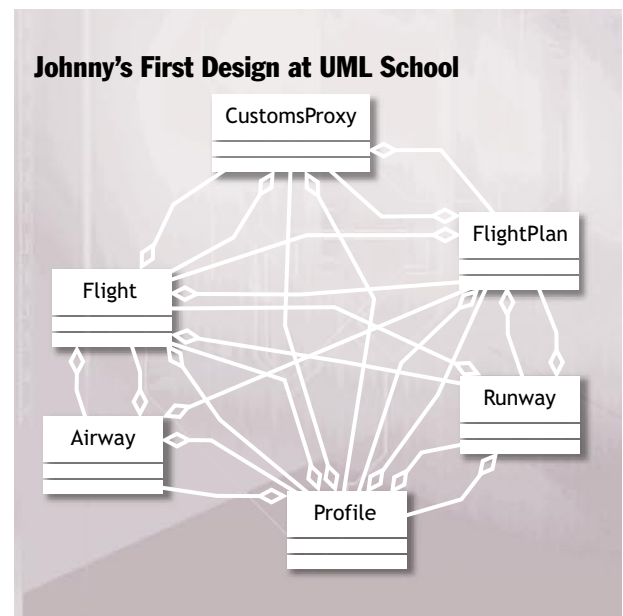
A Note from the Author.

Before readers accuse me of demonstrating flagrant insensitivity to those struggling with substance dependencies or mocking the 12-step programs intended to help overcome them, let it be known that I am fully aware of the pain associated with substance abuse and this article is in no way intended to make light of this very serious issue. While some parallels do exist, there is no comparing the devastating impact of substance abuse with that of UML Fever.

should be used as guidelines and must be considered in the context of specific program circumstances.

A UML-CENTRIC TRAINING CURRICULUM

A software organization's training curriculum should be focused on addressing any gaps identified as the result of comparing employee skill sets with the skills they require to do their jobs. It is a raging display of infection when comprehensive UML training campaigns are launched without a skill-set analysis showing them to be warranted, even more so when the employees are very weak in truly core software engineering skills.



As opposed to UML training, it is a much better investment to send inexperienced software designers to classes where they will learn about object-oriented analysis/design, composability, and abstraction. The value of UML syntax courses must also be weighed against user interface or database design training that may be much more important to an organization for developing its core competencies.² This is not to say that some UML training is not necessary, but it must be scheduled in response to need, not fever-induced delusion. Check your organization's UML Fever symptoms today by verifying that UML training is based on need as opposed to a vendor-declared crisis.

DESIGN BRAINSTORMING DEGRADES INTO UML SYNTAX FREE-FOR-ALL

A scene common to the development of many systems is that of a group of software designers collectively brainstorming their ideas on a whiteboard. The circles,

rectangles, cylinders, and arrows drawn on whiteboards are just a few of the symbols used to convey thoughts of dependencies, information flow, and control sequencing of the software under consideration. It is not unusual for the products of such brainstorming sessions to become the foundations of subsequent detailed design and implementation activities.

While the whiteboard has long been known as a setting associated with the informal exchange and evolution of ideas, the spread of UML Fever threatens to desecrate this long-hallowed site of software engineering creativity. Such desecration may occur when UML homeboys (or girls) strong-arm the focus of brainstorming sessions into religious debates of UML syntax and usage. Invaluable nuggets of wisdom such as "That is invalid UML syntax," or "That should be an activity diagram instead of a sequence diagram," are just a few of those signaling the beginning of a brainstorming session's end. The level of infection is particularly severe when the UML becomes forbidden as an allowable syntax for use during brainstorming sessions because of the unacceptably high probability of resulting argument and distraction. Is there a correlation between your lament of the UML's invention and the presence of particular individuals in brainstorming sessions?

EXCLUSIVE FOCUS ON STATIC SOFTWARE DESIGN

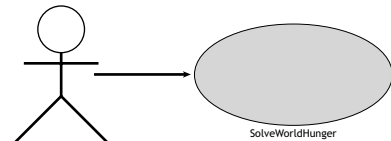
UML Fever is in obvious control of modelers who spend their time myopically focused on the static dimensions of software design. It is a wonderfully simple world that the stricken create for themselves by ignoring the complexities caused by having to consider software's dynamic behavior. Considerations of information locality, data coherency, control flow concurrency, and latency are often casually ignored or considered to be annoyances serving only to slow diagram production. In many cases, the afflicted don't explicitly ignore the dynamic aspects of software design, they are simply oblivious to its significance. After all, the dynamic behavior of software can't be all that important if the UML does not offer a rich syntax with which to express it succinctly. Systems engineers are equally vulnerable to this symptom by focusing strictly on functional flows and ignoring the -ilities that are so important to framing the collective solution space.

The description of this symptom should not be misunderstood to suggest that the static perspective of software design is unimportant or that dynamic behavior should not sometimes be abstracted. Instead, the point being made is that the comforts of limiting the modeling activity to drawing lines between static elements come

at the risk of the emerging design's validity. Specifically, ignoring the dynamic design will sometimes cause breakage of the static design as a result of its inability to meet performance, availability, or concurrency objectives. Ask a UML commando near you how they are considering the dynamic behavior of software in their diagrams as a test of this symptom.

ABSENCE OF STAKEHOLDER INPUT IN ARTIFACT DEVELOPMENT

A signature characteristic of UML Fever is for models and diagrams to be produced without a clear understanding of



intended stakeholders or their associated needs. Although the afflicted may emphatically insist that their diagrams are intended to "help the developers" or "facilitate integration," such empty suggestions are typically brazen advertisements that downstream needs are very poorly understood, let alone being addressed.

A good test of the symptom being described is to randomly select a suspected UML robot and probe them as to who, by name, is going to use the diagrams they are producing and for what purpose. Failure to produce a convincing answer is a likely sign of UML Fever in either the modeler or in the individual(s) responsible for developing the project plans sanctioning the diagram production activity.

PEOPLE WITH INAPPROPRIATE SKILL SETS IN CRITICAL POSITIONS

Software organizations are guilty of creating dangerous breeding grounds for UML Fever by staffing influential positions with individuals who are not qualified to serve in them. Just as we have long heard about the consequences of putting a fox in charge of a chicken coop, no less devastation should be expected when putting a design-challenged UML cowboy in charge of a software organization's modeling strategy or a UML ranger in a project management position who will define all progress milestones in terms of UML diagrams. No matter how well intended people may be, good intentions do not overcome poor qualifications and nonapplicable experience.

A recurring theme within fever-ravaged organizations is that the UML cheerleaders responsible for infecting them have never worked in positions that would illuminate the implications of their guidance. How can it be that beautiful UML models do not sing to the designers,



UML Fever

Diagnosis and Recovery

like angels, inspiring the development of perfect software that precisely meets customer needs? A classic example of this symptom is when the purported stakeholders of value-deficient models report their worth to the individuals mandating them and subsequently get lectured on lack of vision. What are the credentials of the drum majors leading your UML hit parade? Are you being led toward “UMLtdown”?³

A UML MODEL HAS TURNED INTO THE PRODUCT

The situation where producing a UML model becomes the focal point of a software organization's efforts is probably the most serious of all UML Fever symptoms. Disaster is inevitable for dysfunctional organizations allowing their focus to shift from developing software to production of magnificent models advertised as being only minor transformations shy of the deliverable source code.

A predominance of UML-centric activities in an organization is indicative of the fever's infiltration of fabric and culture, driving that organization away from the important goal of lean and agile software development. The signature symptom of an organization that is confusing a model for its product is the existence of a project plan primarily filled with milestones associated with the production of UML diagrams. What percentage of your program plan is filled with milestones associated with creation of UML diagrams as opposed to software?

UNREALISTIC EXPECTATIONS OF UML MODELS

It is a simple diagnosis to make when software organizations suggest that they will analyze their UML models to assess fault tolerance, performance, and the safety of complex systems without even writing software: rampant UML Fever. While such analysis is possible on very isolated levels, and it is true that UML diagrams can be used to illustrate safety concerns, these are very complex matters to reason about even using tools specifically designed to support their investigation.

Those afflicted with the fever often campaign for the UML as the solution for complex problems and usually have little trouble convincing the similarly desperate or naïve of the legitimacy of doing so. Is your UML model going to ensure that there are no logic errors in the corresponding software implementation? If you think so, consider yourself potentially infected.

THE UML IS USED IN WAYS OTHER THAN INTENDED

One of the most common symptoms of misapplication is to use the UML for modeling software to painstakingly fine levels of detail. While some follow this practice because they believe it will increase the probability that the resulting code will be more correct, others follow it by design because their organizations follow a sequential development process. A telltale extension of this symptom is when people purport that they need not be bothered with constructing software design models because reverse-engineering their detailed implementations overcomes the need for doing so.

Systems engineers can be equally guilty of UML misuse by using it beyond its purpose as a tool to help identify and allocate requirements through use case development. The fever is rampant when armies of modelers continue to decompose use cases into even finer-grained ones well beyond a point where requirements could have been identified and allocated. The symptom being described is most prevalent in organizations where “the UML” is the immediate answer for solving any issue. Does your organization use the UML to exist or does it exist to use the UML?

THIS ARTICLE IS PERCEIVED TO BE AN ATTACK ON THE UML

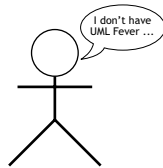
Many who are afflicted with the fever will perceive this article to be a cowardly attack on the UML likely inspired by the author's ignorance and bitterness. Such ignorance and bitterness must certainly be the result of trauma related to his diagrams being criticized in UML training class or having been beaten with rolled-up activity diagrams as a child. Beyond sentiments of cowardice, the afflicted will also feel very angry about this article's message. For example, modeling zealots will be angry that their UML empires are being threatened, software managers will not be pleased with suggestions that they may have been duped into staffing diagram production factories, and recent graduates of UML training may be outraged to read that course completion does not qualify them as software designers.

The angry are often too ravaged by the fever to recognize that this article is not about hammering the UML or rejecting the value it brings to the many software organizations that use it wisely. Instead, this article is focused on identifying symptoms of outrageous UML misuse and providing guidance for corrective action, objectives that no fever-free individual would be angry with. Are you angry?

COMMONLY HEARD MUTTERINGS FROM THOSE WITH UML FEVER

In addition to the behavioral symptoms already described, there are a number of common mutterings heard from those afflicted by UML Fever that help facilitate its diagnosis. While no single utterance necessarily leads to a conclusive diagnosis, it is the accumulation of repeated mutterings, as well as exhibition of behavioral symptoms, that provide an excellent basis for diagnosing the fever.

- “I don’t have UML Fever.”
- “I can stop creating diagrams any time I want to.”
- “We need to hire more modelers.”
- “We need more UML training.”
- “Alex Bell is a UML heretic.”
- “We need to model that!”
- “If I didn’t create UML diagrams, I’d be doing something much worse.”
- “Now is not a good time to stop modeling.”
- “Don’t delete that diagram!”
- “I use only 95 percent of the UML.”
- “We’ll reverse-engineer the design model from the code.”
- “I don’t want to be left out; everyone is creating UML diagrams.”
- “Our UML tool vendor has validated our development process and modeling approach.”



BEWARE OF SELF-INVENTED AD HOC TREATMENT REGIMENS

Pain and helplessness are among the dominant feelings when the healthy observe their peers or colleagues struggling with UML Fever. These feelings often inspire the well-intended to invent and apply their own treatment regimens with the objective of returning the afflicted to be positively contributing members of their software organizations. The most effective approaches for treating the fever are those that encourage the stricken to recognize and admit to their afflictions by themselves, much like the 12-step program described in the next section. The least effective approaches, on the other hand, are those that promote anger or estrangement, or push the afflicted into even deeper dysfunction. The treatments described here are examples of such ineffective approaches and should be avoided:

Don’t threaten the afflicted. Using threats as a tactic to rid the afflicted of UML Fever is like trying to put out a fire with gasoline. Threats of destroying the plastic UML diagram stencils of the afflicted if they don’t move out of

their UML communes or don’t resign from UML activist groups, for example, will only strengthen their modeling resolve.

Don’t disable modeling tool license servers. This tactic has very little impact on the afflicted beyond being a nuisance. Not only are fraudulent licenses readily available on the Internet, but freeware or trial versions of UML toolsets can also be downloaded to overcome a temporary lapse in diagram production by the afflicted.

Don’t use UML interventions except as a last resort. Abducting fever-afflicted peers or colleagues for a UML intervention should not be considered except under the supervision of specially trained treatment counselors. Even then, colleague interventions should be considered only as a last resort and should focus on how the abductee’s UML Fever has impacted the lives of the abductors.

Don’t try to control peers of the afflicted. Any attempt to control the peers with whom UML Fever afflictives cavort is counterproductive to promoting recovery. Declaring the cubicles of UML evangelists to be off-limits, for example, will only cause the afflicted to be even more drawn to them.

Don’t force 12-step program participation. Afflictives of UML Fever must voluntarily participate in 12-step recovery programs for those programs to be effective. Forced participation will only trigger resentment and anger in the afflicted, both complicating and delaying the recovery process.

THE 12-STEP RECOVERY PROGRAM

The 12-step recovery program described here is specifically designed for those who have earnestly admitted to being under the control of UML Fever and are committed to pursuing a healthier engineering lifestyle. The program should be diligently followed, step-by-step, to rid both individuals and their software development organizations of the fever’s dilapidating effects. Software organizations should encourage and sponsor the creation of recovery groups for afflicted employees, and consider the absorption of program costs as part of employee health packages.

These 12 steps are designed to attack the fever from both sociological and technical perspectives. They should be followed in a strictly sequential order, with each step started only as the result of completing the previous.

1. *We admitted we were powerless over our rampant misuse of the UML—that our software projects had spun out of control with misguided focus on model and diagram production.* This is the first and most critical step for recovering from UML Fever. It is only subsequent to problem



UML Fever

Diagnosis and Recovery

recognition that afflicttees are capable of committing themselves to recovery.

2. *Made a decision to turn our technical direction and development activities over to the guidance of experienced architects, designers, and developers.* A characteristic common to many fever afflicttees is a void of actual software design and development experience. The importance of this step is an admission by recoverees that a lack of such experience could indeed be a legitimate deficiency for understanding which artifacts may or may not be valuable for developing software products.

3. *Made a searching and fearless technical inventory of our skills and vowed to pursue the appropriate training, mentoring, and education to overcome any identified deficiencies.* This step requires those in recovery to address any identified shortfalls in regard to the skills required to do their work. Simply recognizing deficiency is not sufficient for recovery. A plan must be made and executed to overcome any identified shortfalls.

4. *Came to believe that developing and following a lean, iterative, and incremental software development process could restore our program to order.* UML Fever and a sequential software development process often go hand in hand. The perceived need for a software design to be completely spec'd out before proceeding with development leads to the construction of both large and detailed UML models. The completion of this step is important to the afflicted because it signals recognition that software development may productively commence without having first documented the entire design in the UML.

5. *Admitted to our supervisor, to ourselves, and to another colleague the exact nature of our misguided UML usage.* The importance of this step is to begin the restoration of a previously respected technical reputation soiled by the effects of UML Fever. A detailed confession of past UML impropriety serves to reinforce that a recoveree truly recognizes past errors in judgment.

6. *Made a commitment to align the production of all UML artifacts to stakeholders who are actively involved in vouching for their value and defining their content.* Those afflicted with the fever often believe that downstream stakeholders must not understand their own needs if the diagrams delivered to them are characterized as having little or no value. The completion of this step signals recognition and acceptance that models and diagrams

cannot be created based on what the modeler thinks may be of downstream value, but must be created based on the stakeholders' specified needs.

7. *Humblly asked that any anger we expressed against the well-intended trying to help us realize and overcome UML Fever in the past be forgiven.* It is not uncommon for those wallowing in the throes of UML Fever to have alienated themselves from the healthier side of the technical community as the result of ill-spirited dialogue occurring in the past. This step is intended to initiate the repair of any relationships that are still salvageable with those previously alienated.

8. *Made a list of all projects we had harmed with bad UML usage/guidance and became willing to make apologies to them all.* In addition to acting as a conscience cleanser, an important by-product of completing this step is to strengthen future employment opportunities by announcing and demonstrating recovery to past colleagues and supervisors previously harmed.

9. *Made a commitment to producing the minimal number of UML diagrams and models necessary to develop our product without sacrificing quality.* The completion of this step brings with it the realization that UML artifacts are enablers for developing software products as opposed to being the products themselves. It is a defining moment in the recovery process to switch from a mode where success is measured by the size of a UML model to a mode where the primary objective is to develop software.

10. *Were committed to taking a technical inventory of individuals in key positions to ensure that the appropriate skills are possessed to perform in them.* This step is primarily directed at the codependents enabling the proliferation of UML Fever within their organizations. No matter how committed individuals might be in their quest for recovery, UML health at the organizational level is impossible if those in leadership fail to complete this step.

11. *Sought through research and analysis to improve our understanding of the UML to understand its strengths and limits so we can apply it intelligently in our endeavors.* Completion of this step signals that recoverees are on the road to understanding how the UML should best be used, are capable of choosing the correct level of model detail, and are able to define the appropriate lifetimes of diagrams and models. Recoverees must remember that it is perfectly acceptable to dispose of UML artifacts after they have served their purpose.

12. *Having had a technical awakening as the result of completing these steps, we will carry this message to other afflicttees of UML Fever and practice these principles*

Agility Promotes Long Life

SCOTT W. AMBLER, RONIN INTERNATIONAL

Alex Bell has many very good points in his article, many of which have been previously recognized and overcome within the Agile Modeling (AM) methodology (<http://www.agile-modeling.com>). I suspect that many organizations are dying from UML Fever because of the IT industry's self-interests and dysfunctional views of modeling tools.

My observation is that lean and effective organizations conduct upward of 90 percent of all modeling as sketches on paper or whiteboards, with only a few using sophisticated software-based modeling tools. The reality is that the value is typically in the modeling effort itself, not in the model. Why is it that so few software modeling books or academic papers seem to focus on sketching in terms of recommended best practices? Shouldn't we help developers to get better at what effective modelers actually do in practice, instead of trying to inflict the questionable visions of tool vendors and/or inane academic theories on them?

My experience is that effective modelers do the following:

Create agile models that are just barely good enough.

The implication is that agile models are the most effective models possible—if a model isn't good enough, then you have more work to do; if it is more than good enough, then you clearly have invested extra effort that wasn't required. The challenge is that "good enough" is in the eye of the beholder.

Model with others. Software development is a lot like swimming: it's very dangerous to do it alone. Just as extreme programmers insist on programming in pairs, agile modelers know that the best models are developed collaboratively by several people.

Apply the right artifacts. The diagrams of the UML are an important part of your modeling toolkit, but you need more if you're building business software. Every single business system I've ever built has a user interface, implements business rules, and accesses either relational or file-based data sources. Yet the UML doesn't yet have official profiles for any of these things. I suspect that if the OMG (Object Management Group) were to eat its own dog food and develop a UML use case model describing how people actually go about building software, it would identify the gaping holes that still exist in the UML after all these years. At <http://www.agilemodeling.com/artifacts/>, I overview a wide range of potential modeling artifacts, including all 13 of the UML 2.0 diagrams.

Insist on active stakeholder participation. Your stakeholders—users, managers, operations professionals, and so

on—are the source of a system's requirements. If you choose to adopt inclusive modeling techniques that are simple to understand, such as sketches, CRC (Class Responsibility Collaborator) cards, and low-fidelity user interface prototypes, and that use simple tools, such as paper and whiteboards, it is possible for your stakeholders to be actively involved.

Use the simplest tools. Sometimes the simplest tool for the job is paper and pen, other times it is a sophisticated MDA (model-driven architecture)-compliant modeling tool. Just like your grandfather used to say, use the right tool for the job.

Not-so-effective organizations mandate the use of a documentation tool—oops, I mean modeling tool—thinking that the resulting documentation will be valuable. Yes, there are some great software-based modeling tools available and most offer good productivity enhancements if used correctly. Correct usage, however, is easier said than done.

The reality is that the value is typically in the modeling effort itself, not in the model.

Isn't it interesting how most vendors are desperate to avoid the term *CASE* these days? It makes me wonder when they'll start avoiding *MDA* as a marketing term. Perhaps we should listen less to the tool vendors' vision of what they want us to buy and instead focus on an accurate vision of how we can actually model effectively? AMDD (Agile Model Driven Development, <http://www.agilemodeling.com/essays/amdd.htm>) is one such vision. If you choose to live by agile modeling approaches, you will greatly reduce your program's chances of dying from UML Fever.

SCOTT AMBLER is a senior consultant with Ronin International Inc. (www.ronin-intl.com). His most recent book, *The Enterprise Unified Process: Extending the Rational Unified Process*, was published in February 2005 by Prentice Hall PTR. He is also the author of *The Object Primer 3rd Edition: Agile Model Driven Development with UML 2.0* and *The Elements of UML 2.0 Style*, both published by Cambridge University Press.



in all of our software engineering affairs. The importance of this step is for the previously afflicted to become walking advertisements that UML Fever can be overcome. Individuals recovering from the fever are in a position to serve as inspirational icons for those still struggling with it.

SUMMARY

Many of us are aware from first-hand experience that some of life's pleasures can be enjoyed to excess with great detriment. Unfortunately, this detriment is not isolated to those guilty of the excess; it extends to peers, colleagues, and loved ones. Society endeavors to protect its citizens from the harms of excess by restricting activities such as smoking, consuming alcohol, or accessing controlled narcotics by establishing constraints such as minimum age requirements, having to pass examinations, or receiving authorization from government-approved agents. Much to the misfortune of many software organizations, however, there are no such constraints imposed on usage of the UML. Anyone capable of using a UML modeling tool is poised to become a diagram-making machine, regardless of competence in software design, systems engineering, or requirements analysis.

It is astonishing that software organizations task individuals who have little understanding of basic software design principles such as abstraction, concurrency, and composability to model the structures and behaviors of multimillion- or billion-dollar software systems. An interesting world it would be if civil engineering organizations were to allow similarly unqualified people to model bridges, roads, and tunnels without intimate knowledge of beam theory, stress, or loading. The complex art of software design is often trivialized into the creation of UML diagrams.

The UML Serenity Prayer

Oh great UML Spirit, please grant me the insight necessary to valuably use the UML;

the courage to challenge its misuse;

and the wisdom to recognize when UML Fever has robbed otherwise intelligent people of their good senses.

UML Fever does not disappear as the result of continuously reciting "The UML Serenity Prayer" (see sidebar) or desperately hoping that the afflicted suddenly see the light through divine intervention of the UML gods. Commitment and action are fundamental to recovery. Software organizations may need to reevaluate their development processes, reassess training curricula, and reconsider staffing profiles as part of a recovery regimen that begins with fever diagnosis. If software organizations are incapable of performing UML Fever self-diagnosis, it is their duty to hire experts, other than their UML tool vendors, to make independent assessments and to act upon the subsequent results.

Are any of the symptoms of UML Fever identified in this article recognizable in you or your organization? If so, are you committed to completing the 12-step recovery program designed to overcome them? Or, do you think it is best to wait for the problems to resolve themselves? Perhaps you think that this article is talking about UML Fever ravaging *other* organizations, not yours? Until software organizations take charge of their technical health by actively dealing with the enablers and the stricken alike, UML Fever will continue its indiscriminate attack on software programs across the planet. ☹

REFERENCES

1. Bell, A. E. 2004. Death by UML Fever. *ACM Queue* 2(1): 72-81. <http://www.acmqueue.org/modules.php?name=Content&pa=showpage&pid=130>.
2. Ambler, S. W. 2001-2004. Be realistic about the UML: it's simply not sufficient. Agile Modeling Essay; <http://www.agilemodeling.com/essays/realisticUML.htm>.
3. *yoo-em-elt down*: n. The condition where a program runs out of money building UML models before it can deliver a product.

RELATED LINKS

- Larman, C. 2004. What UML is and isn't. *Java Pro* (February); http://www.fawcette.com/javapro/2004_03/magazine/features/clarman/default.aspx.
- Meyer, B. 1997. UML: the positive spin. *American Programmer* (March); <http://archive.eiffel.com/doc/manuals/technology/bmarticles/uml/page.html>.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

ALEX BELL is a software architect with The Boeing Company.

© 2005 ACM 1542-7730/05/0300 \$5.00