Death by UML

ALEX E. BELL, THE BOEING COMPANY

Fever

Self-diagnosis and early treatment are crucial in the fight against UML Fever.

A potentially deadly illness, clinically referred to as UML (Unified Modeling Language) fever, is plaguing many software-engineering efforts today. This fever has many different strains that vary in levels of lethality and contagion. A number of these strains are symptomatically related, however. Rigorous laboratory analysis has revealed that each is unique in origin and makeup. A particularly insidious characteristic of UML fever, common to most of its assorted strains, is the difficulty individuals and organizations have in self-diagnosing the affliction. A consequence is that many cases of the fever go untreated and often evolve into more complex and lethal strains.

Death by UML Fever

Little has been published in medical annals on UML fever because it has only recently emerged as an affliction. *The New England Journal of Medicine* has been silent on the disease, as has research produced by the world's most prestigious medical institutions. The content of this article represents many years of on-the-job research and characterizes all known strains of UML fever, as well as many of the known relationships recognized to exist between them. The article will conclude with disclosure of the only known antidote for the many and varied strains of UML fever.

Before commencing with the characterization of UML fever and its associated symptoms, it is important to emphasize that UML itself is not the direct cause of any maladies described herein. Instead, UML is largely an innocent victim caught in the midst of poor process, no process, or sheer incompetence of its users. Through no fault of its own, however, UML sometimes does amplify the symptoms of some fevers as the result of the often divine-like aura attached to it. For example, it is not uncommon for people to believe that no matter what task they may be engaged in, mere usage of UML somehow legitimizes their efforts or guarantees the value of the artifacts produced.

This article exploits the fact that the presence and associated severity of many software-related maladies on a program can often be observed and measured in terms of UML: too much, too detailed, and too functional, for example. Some readers may be quick to suggest that the same exploitation could be made regardless of a program's selected modeling approach. There may be some truth here, but no other technology has so quickly and deeply permeated the software-engineering life cycle quite like UML.

THE METAFEVERS

Extensive research has shown that UML fever can be categorized into four well-defined groups, known as *metafevers*. Their common laboratory names are *delusional*, *emotional*, *Pollyanna* (a person regarded as being foolishly or blindly optimistic), and *procedural* (see figure 1). Each of these metafevers is described in the following sections, as are the strains associated with them. Although much more is known about each of the strains than written, the objective of this particular article is to describe them to the extent that they are characterized and distinguishable from the others.

Delusional Metafever. The delusional metafever comprises UML fever strains that are considered by many to be among the most deadly. This metafever is best known by its devastating effects on the thought and judgment processes of otherwise healthy managers and engineers. It is very common for the fevers in the delusional category to damage the human immune system to such an extent that the body becomes susceptible to many other UML fever strains (see figure 2).

Utopia fever. Subjects afflicted with utopia fever typically believe that UML is a radical new technology with almost divine origins. Mutterings such as, "How did we get where we are today without UML?" and "Just think how much more advanced our technological revolution would be if we only had UML 20 years ago?" are common among those afflicted. Other symptoms of this fever include an amnesia-like condition causing people to forget that many complex software-based systems have been successfully built over the years without the benefits of UML.

This fever's direct symptoms are relatively benign on their own, but contracting utopia fever will most certainly result in affliction of more dangerous strains, particularly 42 fever (described later) where UML is believed to be the answer to all problems. A good litmus test for probing suspected carriers of utopia fever is to ask if they know of UML's origins or what methodologies engineers were using before UML to design complex software-intensive systems.

> Reality is that which, when you stop believing in it, doesn't go away.—Philip K. Dick

Blind adopter fever. This strain is recognized in those afflicted by a loss of judgment when it comes to assessing appropriate usage of available technologies and processes for their own programs. As opposed to tailoring or rejecting, victims of blind adopter fever tend to accept what other people have done on other programs even though it may not be applicable to their own.

Engineers afflicted with blind adopter fever have been observed to blindly force state machine semantics into all of their classes just so they can take advantage of forward engineering technologies that convert UML diagrams into code. Another observed symptom of this fever includes

usage of software development processes or process frameworks right out of the box as opposed to tailoring them to fit the needs of their own programs. A side effect of using such processes is wasting time and money on producing many unnecessary artifacts.

With most men, unbelief in one thing springs from blind belief in another.—Georg Christoph Lichtenberg

Abracadabra fever. The most frequently observed symptom of those afflicted with abracadabra fever is an impaired sense of reality. Managers afflicted with this fever have been observed salivating profusely when told of technologies that automatically develop software from UML diagrams. Thoughts of improving code productivity metrics, previously dragged down by having to develop extremely complex business logic, also produce symptoms that include wide eyes and mutterings of large Christmas bonuses. One can only imagine the future symptoms of abracadabra fever when managers postulate automatic development of entire systems using MDA (Model-driven Architecture).

While managers are the primary demographic afflicted by abracadabra fever, engineers have also been known to be susceptible. A common symptom of this fever on the engineering level is the expectation of almost surreal information being derived from gargantuan UML models. Insights into throughput, fault tolerance, latency, and system safety, for example, are just a few that are expected solely from UML models without having to be bothered with writing code or doing engineering work to characterize comprehensive component behaviors. Abracadabra fever appears to be very infectious among engineers who have little practical experience using UML.

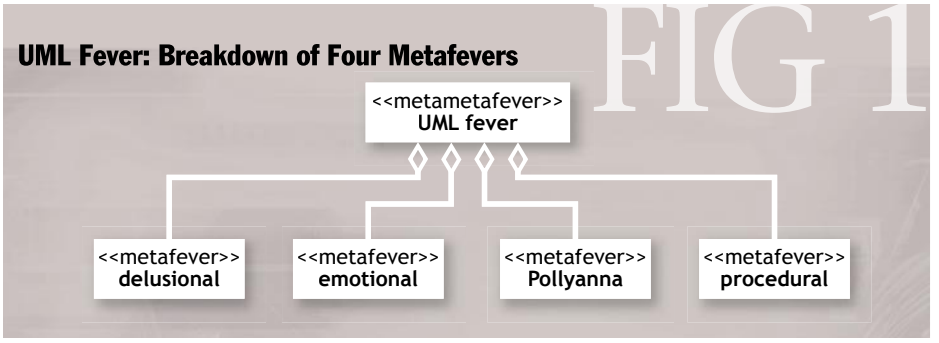
The truly educated man is that rare individual who can separate reality from illusion.—Author Unknown

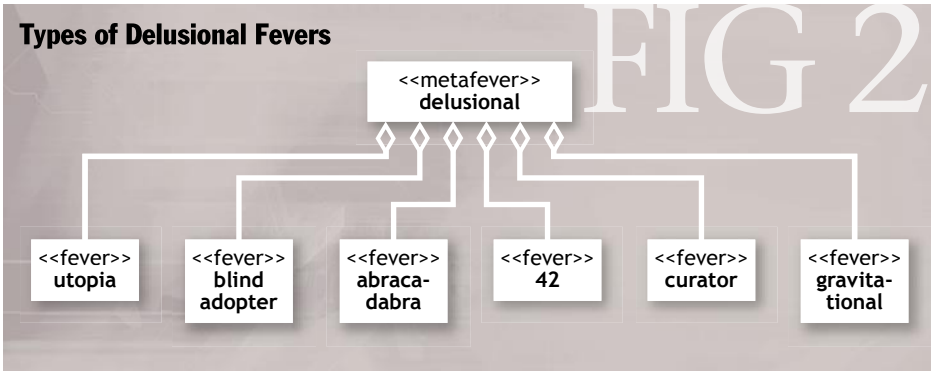
42 fever. As opposed to the celebrated "42" being the answer to any question about life or the universe, as suggested in Douglas Adams's *The Hitchhiker's Guide to the Galaxy*,[1] those afflicted with 42 fever argue that "UML" is actually the correct answer. The classical symptom of those afflicted with 42 fever in the sphere of software engineering is to have an a priori delusion that UML is the solution for all software-engineering problems. Research has shown that the delusion in victims of 42 fever can be significantly reduced by secretly playing subliminal messages in their work areas emphasizing that UML's creators did not intend for it to be the answer to all of software engineering's dilemmas.[2]

Although 42 and abracadabra fevers are similar and often afflict their victims simultaneously, some subtle differences are worthy of note. Those afflicted with 42 fever believe that UML is the correct answer to all questions, period. Those suffering from abracadabra fever, on the other hand, are not deluded that UML is necessarily the answer to everything, only the problems where they believe it to be the magical answer.

If the only tool you have is a hammer, you tend to see every problem as a nail.—Abraham Maslow

Curator fever. Much as a museum curator has a fascination and passion for paintings, those in the software engineering realm afflicted with curator fever have a similar absorption in UML diagrams. This absorption is fueled by curator fever's propensity to delude its victims into believ-





Death by UML Fever

ing that production of UML diagrams, as opposed to the engineering content behind them, is the single most important activity in the software-development life cycle.

A commonly observed behavior by those in the grips of curator fever occurs when domain analysts create volumes of use-case diagrams but remain oblivious to the fact that the most important artifact of use-case modeling is developing the supporting text.[3] UML interaction diagrams with messages analogous to "solve world hunger" between objects are of little value to any stakeholders. Use-case modelers afflicted with curator fever often declare software designers incompetent if they are unable to produce software designs based on extremely high-level diagrams. The only place a software designer would typically prefer a picture over a thousand words is in a museum.

> A painting in a museum hears more ridiculous opinions than anything else in the world.—Edmond de Goncourt

Gravitational fever. This fever causes delusions in those afflicted to believe that gravitational acceleration enables their UML artifact mass to have value. For those unfamiliar with Newton's second law of motion, those suffering with gravitational fever believe that the progress of software-engineering effort is directly proportional to the weight of the project's UML artifacts.

Research has shown that software managers are the demographic most susceptible to gravitational fever. A commonly observed symptom of this fever is for managers overseeing a code-hacking frenzy to direct development staff to reverse engineer their code into voluminous UML diagrams. These UML diagrams are subsequently passed off as design models, as opposed to the implementation models they really are.

Gravitational fever is often misdiagnosed as curator fever because of the similarity of the two afflictions. The subtle difference between these fevers, however, is that those afflicted by curator fever are very interested in the quality of UML diagrams, whereas those afflicted by gravitational fever care only about their weight.

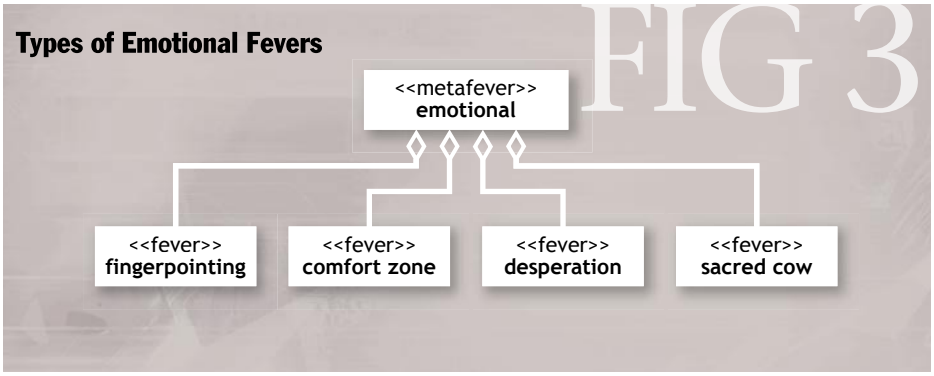
> Those who speak most of progress measure it by quantity and not by quality.—George Santayana

Emotional Metafever. The strains composing the emotional metafever group tend to attack and take advantage of the human body's emotional system. They include the fingerpointing, comfort zone, desperation, and sacred cow fevers described in this section (see figure 3).

Fingerpointing fever. This fever coincidentally strikes those who are in the final stages of recovering from more serious fevers previously contracted. The severity of fingerpointing fever appears to be directly related to the amount of time and money previously wasted developing unnecessary UML artifacts while being ravaged by other fevers. A frequent symptom of fingerpointing fever is for its afflictees to unjustifiably blame a software-development process or framework for advocating the development of too many UML artifacts. Another common symptom of this fever is to blame UML itself for being too expressive and encouraging design artifacts to be modeled to unnecessarily low levels of detail.

> It is no use to blame the looking glass if your face is awry.—Nikolai Gogol

Comfort zone fever. Victims of comfort zone fever typically enjoy a hypnotic sense of tranquility while they are engaged in activities focused on creating UML artifacts. Clinical analysis has shown that any attempt by its afflictees to migrate from creating UML diagrams onto software development activities later in the life cycle causes this tranquility to abruptly and traumatically cease. As a result, the victim's UML diagrams become large in num-



ber and extremely detailed (much to the delight of those suffering from gravitational fever).

Comfort zone fever is recognized by resistance in its victims to depart from the comforts of UML diagram creation. Program risk is often amplified in the presence of comfort zone fever since the proposed designs are not validated as early as they should be in the form of implementation.

The scholar who cherishes the love of comfort is not fit to be deemed a scholar.—Confucius

Desperation fever. Extensive clinical research has identified a correlation between the occurrence of desperation fever and the existence of project traumas such as slipped schedules, low productivity, and poor product quality. A symptom of those plagued with desperation fever is flattened ears that result from spending inordinate amounts of time on the telephone speaking with vendors in search of products that will solve all known project woes.

Victims of desperation fever often purchase expensive UML-centric products only to discover later that correct usage of those products does not align with their absent or broken software-development processes, often the root of their problems in the first place. The severity of desperation fever typically escalates as the result of highly paid consultants telling afflictees that newly purchased products will not bring benefits without major overhauls to existing software-development practices.

It is characteristic of wisdom not to do desperate things.—Henry David Thoreau

Sacred cow fever. Those afflicted with sacred cow fever develop intense emotional attachments to UML diagrams and refuse to allow any that have outlived their usefulness to die with the dignity they may deserve. Sacred cow fever results in many costs to a program including the cost of maintaining obsolete artifacts, misinformation propagation, and unnecessary consumption of storage resources. Clinical research suggests that this fever causes its victims to believe that by throwing away UML diagrams they are somehow negatively impacting the forward progress of the program (much to the delight of gravitational fever afflictees).

Treatment for victims of sacred cow fever should include a counseling regimen reinforcing that the value of UML diagrams is often transient and discarding those that are no longer of value is encouraged.[4]

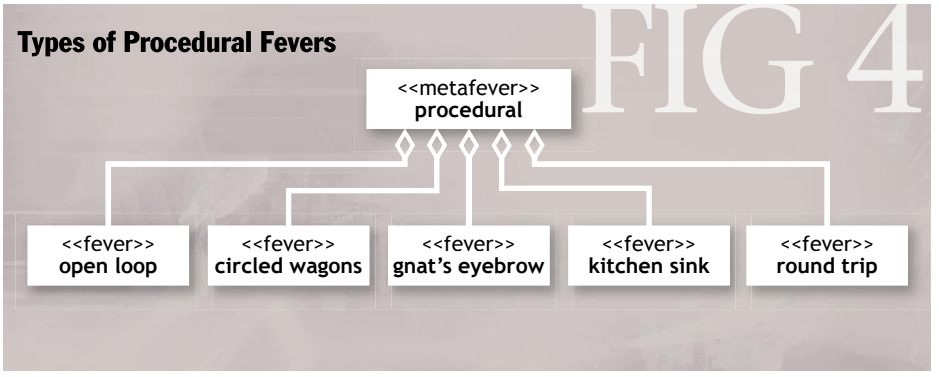
It is a very sad thing that nowadays there is so little useless information.—Oscar Wilde

Procedural Metafever. The UML fever strains belonging to the procedural metafever (see figure 4) tend to impair their victims from recognizing that they are not following a development process or that they may be following a very bad one. The procedural metafever strains are known as open loop, circled wagons, gnat's eyebrow, kitchen sink, and round trip.

Open loop fever. The effects of open loop fever stimulate the urge for rampant creation of UML diagrams with no traceable objective or having no obvious stakeholder. Victims of open loop fever believe that the act of creating UML diagrams alone is a guarantee of value-added activity. Clinical research has suggested that individuals most susceptible to open loop fever are those who have never been end users of UML diagrams and those whose ride on the software life cycle has been very limited.

Hypnotism has proven effective in easing the symptoms of open loop fever. Victims are programmed to tie creation of diagrams to program objectives[5] and to engage with diagram clients to ensure that their needs are addressed. Post-hypnotic interviews with victims of this fever have resulted in the discovery that UML diagrams are not always the preferred artifacts of those downstream in the life cycle.

Furious activity is no substitute for understanding.—H. H. Williams

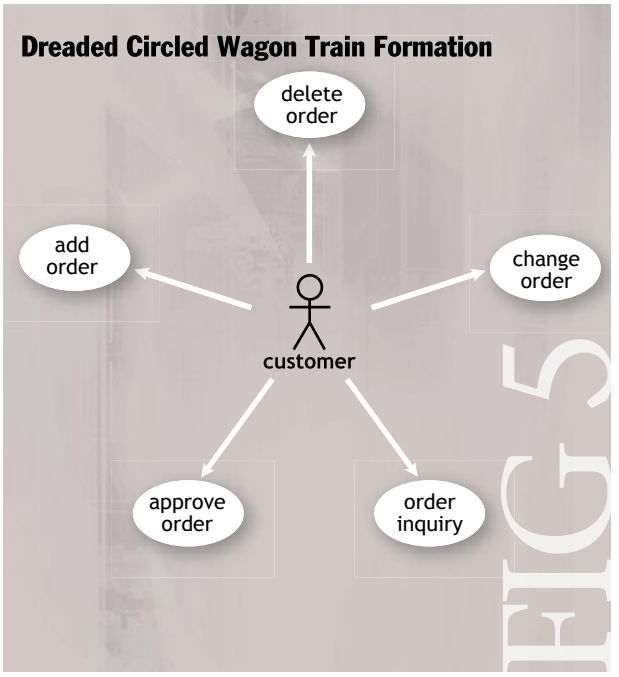


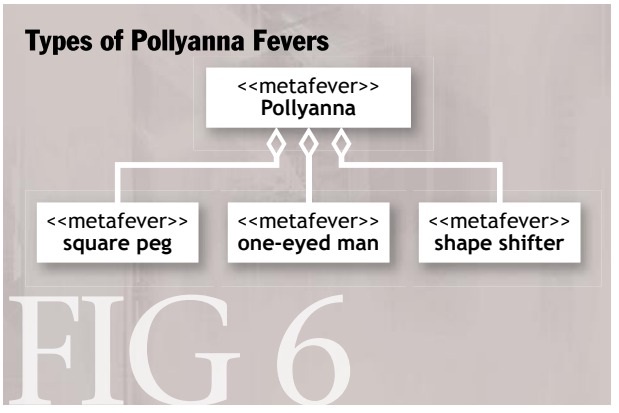
Circled wagons fever. Extensive clinical research[6] has led to the discovery of circled wagons fever. Its primary symptom is its victim's tendency to use UML use-case diagrams to capture fine-grained functional decompositions of their domain space. This fever's name is derived

Death by UML Fever

from observations that its victims have a propensity to create use-case diagrams in the dreaded wagon train formation, as illustrated in figure 5.

Despite the noble intentions of its victims to conduct object-oriented domain analysis, research has shown that circled wagons fever amplifies its victim's natural tendency of breaking a problem down into smaller and smaller chunks to the extent of becoming a compulsive behavior. As opposed to simplifying the use-case modeling activity, victims of circled wagons fever actually complicate it by making the use-case model much more difficult to understand.





Circled wagons fever is often observed in victims having functional decomposition backgrounds. This fever knows no boundaries, however; even people with object-oriented experience sometimes fall victim. This is a very common fever, and its symptoms should be carefully monitored within the engineering staff, particularly in the early stages of a program life cycle.

Wisdom is knowing what to do next; virtue is doing it.—David Starr Jordan

Gnat's eyebrow fever is recognized in its victims by a very strong desire to create UML diagrams that are extremely detailed. Not to be confused with comfort zone fever, where detailed modeling is a side effect of emotional factors, afflictees of gnat's eyebrow fever emphatically believe that it is important to model to very low levels of detail because doing so increases the probability that the resulting code will be more correct. Because of variables such as flux in system requirements and dependent design activities occurring in parallel, for example, the time spent on low-level modeling is often better applied to actual implementation.

Clinical research has shown a high affliction rate of gnat's eyebrow fever in those modelers who have not actually participated in coding activities. A theory supporting the research findings suggests that the coding experience is very important to developing a sense of value that provides modelers with insight into what is and what is not valuable to downstream clients of the model.

Good judgment comes from experience. Experience comes from bad judgment.—Jim Horning

Kitchen sink fever. Victims of kitchen sink fever crave the idea of building gargantuan UML models that include all fine-grained design elements in their detailed splendor. Kitchen sink fever is often accompanied by abracadabra fever in victims who believe that in the absence of code, information can be derived by describing the low-level behaviors of interactions spanning the model's represented subsystems. Victims of kitchen sink fever typically spend significant amounts of time recovering from the effects of crashes of their modeling tools.

Clinical research has shown that one reason victims of kitchen sink fever desire all possible artifacts in their models is that they have a poor understanding of the

Life with UML: It's Still Work

PHILIPPE KRUCHTEN,
SOFTWARE ARCHITECT

Many of the fevers identified in Alex's "Death by UML Fever" are related to the software process, absence of a software process, or to fundamental misunderstanding of what a process is for. I hear comments such as: "Oh, we ran all the activities described by RUP (Rational Unified Process) and created all the UML (Unified Modeling Language) diagrams it prescribes..." or "There is this widget in UML, and I can't find how RUP says to use it." UML is a notation that should be used in most cases simply to illustrate your design and to serve as a general roadmap for the corresponding implementation. Unfortunately, some users of UML leave their brains in the lobby, get settled behind their keyboards, and get busy drawing UML diagrams because doing so is a much easier alternative than doing difficult software engineering work.

information that can be realistically derived from them. Research has also shown that those infected with this fever have typically never used a gargantuan model.

Men have become fools with their tools.
—Thomas Elisha Stewart

Round trip fever. The primary symptom has a very serious effect on those afflicted: a complete loss of the ability to use abstraction as a means of managing the complexities of software design. Victims of round trip fever often fail to recognize the difference between a UML design model and an implementation model that they reverse engineer from detailed code. Software architects responsible for conducting design reviews typically give failing grades if they are given implementation models in review packages.

The origins of round trip fever appear to be rooted in technology. Its victims typically start the traditional design cycle by creating very low-level implementation models so they can take advantage of reverse engineering toolsets. The demographic that round trip fever primarily targets is new graduates who are technologically-centric rather than architecturally-centric. Further research is required, but the downstream impact of what appears to be a serious deficiency of abstraction capability in our young engineers is a serious concern.

Our life is frittered away by detail.
Simplify, simplify.—Henry David Thoreau

Pollyanna Metafever. The strains associated with the Pollyanna metafever (see figure 6) are typically observed in managers and characterized as being the result of foolish or blind optimism. The Pollyanna metafever includes square peg, one-eyed man, and shape shifter fevers.

Square peg fever. This strain causes its victims to believe that all project staff members are interchangeable regardless of experience, background, or education. Symptoms include placing requirements personnel into the roles of software designers and assigning anyone capable of using a UML modeling tool into the domain analyst role. Square peg fever has the propensity to cause rampant UML fever of all strains when methodologists are put into the roles of technologists and practitioners.

Research has shown that some cases of square peg fever are triggered by the ability of most anyone to create UML diagrams that resemble artifacts of value to desperate observers. Square peg fever is particularly prevalent in the face of staffing shortages or skill-set imbalances.

No amount of artificial reinforcement can offset the natural inequalities of human individuals.—Henry P. Fairchild

One-eyed man fever. We have long heard the adage that "the one-eyed man is king in the land of the blind." This adage is embodied in the realm of software engineering as one-eyed man fever and usually afflicts managers who place in positions people who do not have nearly the expertise required to perform in those positions. Victims of one-eyed man fever may be recognized by their selection of the project's UML visionary based solely on the number of half-day syntax classes previously attended.

This fever appears to have a high incidence rate in managers who don't understand the technologies under their jurisdiction to the extent required for making decisions about them. A very undesired side effect of one-eyed man fever is that the blind in the organization often mistake the one-eyed man's practices as best practices and adopt them.

A painting in a museum hears more ridiculous opinions than anything else in the world.—Edmond de Goncourt

Shape shifter fever. Victims of shape shifter fever demonstrate raging affliction by sending people to brief design tool and language syntax classes with the expectation that they return as experts in best practice. Afflictees mis-

Death by UML Fever

take the ability to navigate "File □ New □ Class Diagram" dropdowns as the signature quality of a software designer. The symptoms of shape shifter fever are particularly exacerbated when classes on tool and language usage are taught out of context from how they will actually be used on a program. As some believe "clothes make the man," afflictees of this fever believe "UML makes the designer."

Much like the other strains in the Pollyanna metafever category, shape shifter fever is most prevalent in times of budget constraints and staffing shortages.

> Education is like a double-edged sword. It may be turned to dangerous uses if it is not properly handled.
>
> —Wu Ting-Fang

THE BATTLE

The diverse strains of UML fever described are based on first-hand pain and observation as opposed to simply being the musings of a fiction writer. The lighthearted manner in which the fevers are described should in no way placate the reader into believing that they are not real or that their symptoms are not potentially having serious impacts on the success of their own software programs.

At the root of most UML fevers is a lack of practical experience in those individuals responsible for selecting and applying the technologies and processes underlying a program's software-development efforts. This lack of experience translates into both unrealistic expectation and misapplication of technology, often aggravated by nonexistent or bad software-development processes, a perfect breeding ground for UML fever. If a software organization's battle against UML fever is to be successful, it is absolutely critical that people with practical experience are in place driving the selection of technologies, as well as developing the processes for their associated usage.

The battle against UML fever is even further complicated by the difficulties some software organizations have in self-diagnosing their affliction. As previously suggested in the characterization of the delusional metafevers, some organizations can become so completely absorbed with UML that they lose sight of their primary objective, developing software, in favor of building gigantic models. In such cases, independent and expert help from outside of the organization may be the only option for initiating the UML fever recovery process. Program management must regularly evaluate staff in influential positions for UML fever because its onset is sometimes gradual. Failure to promptly diagnose UML fever may result in its spread at epidemic proportion with devastating impact.

Systematic diagnosis of UML fever is possible only if its symptoms are catalogued, characterized, and publicized—three explicit objectives of this article. Diagnosis, however, is only the first step in the recovery process. Afflicted software organizations must also identify and diligently follow appropriate treatment regimens if they are to rid themselves of UML fever's debilitating effects. The road to recovery is not always easy. Well-intended individuals attempting to launch diagnosis and treatment programs for their afflicted organizations may have to endure the unpleasantries of denial, groundless rationalization, and anger, often with intensities directly related to how high in the organization's leadership hierarchy UML fever has stricken. The battle against UML fever can be won, but not until it is recognized as a genuine malady, and those who are afflicted with it get on the road to recovery. Q

REFERENCES

1. Adams, D. *The Hitchhiker's Guide to the Galaxy*. Crown Publishing Group, New York: NY, 1980.
2. Rumbaugh, J., Jacobson, I., and Booch, G. *The Unified Modeling Language Reference Manual*. Addison-Wesley, Boston: MA, 1998.
3. Larman, C. *Applying UML and Patterns*. Prentice Hall PTR, Upper Saddle River: NJ, 2001.
4. Ambler, S. The Practices of Agile Modeling; http://www.agilemodeling.com/practices.htm.
5. Ambler, S. The Principles of Agile Modeling; http://www.agilemodeling.com/principles.htm.
6. Bittner, K. Why Use Cases are not "Functions." The Rational Edge. (December 2000); http://www.therationaledge.com/content/dec_00/t_ucnotfunctions.html.

LOVE IT, HATE IT? LET US KNOW

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

ALEX E. BELL (alex.e.bell@boeing.com) is a software architect with The Boeing Company. He has more than 22 years of software experience in a variety of domains that include command and control, air traffic control, and telecommunications.

The Fever is Real

GRADY BOOCH, IBM FELLOW

The late '80s and early '90s saw the proliferation of many competing software design methodologies, each with unique concepts, notations, terminologies, processes, and cultures associated with them. While some of these methodologies introduced new and innovative ideas, most aspired to very similar objectives although used dissimilar geometrical figures, colors, and vocabularies to achieve them.

The emerging methodological stew brought with it consequences that included an increasing number of software engineers with nonportable skills, a bane of toolsets that were not interoperable, and an inability to characterize software design in a commonly understandable form. Because none of the methodologies was emerging as a frontrunner, the software engineering industry was desperate for standardization to drive badly needed unification.

The UML (Unified Modeling Language) was specifically created to serve as this unifying force and was unanimously adopted as a standard by the OMG (Object Management Group) in November 1997. The UML introduced a standard notation and underlying semantics with which its users could describe and communicate software design as never before. Its notation was designed to transcend programming languages, operating systems, application domains, and software life-cycle phases to meet the needs of an industry hungry for order. The dawn of a new age in software engineering was poised to emerge.

And emerge it did. Six years since its adoption by the OMG, the UML continues to be widely embraced by the software engineering community. Software development organizations continue to invest in UML training, UML-enabling toolsets, and integration of the UML into their development processes. Many organizations across the globe have successfully used the UML in innovative ways to improve how they design and develop software.

Many other organizations, however, have not enjoyed the successes they assumed to be implicit by merely using the UML. Success with the UML requires thought and planning accompanied by an understanding of its purpose, limitations, and strengths-much like the usage of any technology. It is only through such awareness that an organization is most capable of applying the UML to address its unique needs, in its own context, and in a value-added manner. Blind adoption and usage of technology for technology's sake is a recipe for disaster for any technology.

The maladies described in Alex's "Death by UML Fever" are often indicative of an organization's misunderstanding of the UML, but more so, a systemically troubled development process in whose context it is used. For example, the UML was not intended to model every single line of an organization's software to pristine detail. It was not intended to be a front-end syntax to define the context for comprehensive simulations. It was not intended for drawing diagrams that have no value or do not tie back to a software development process. And finally, the UML was certainly not intended to supplant the software development process itself. On the contrary, the UML was and is meant to be an important element of a software development process whose objectives include prescribing value-conscious usage of applicable technologies.

The next significant evolutionary milestone for the UML is the release of version 2.0, scheduled for 2004. Six years of industry experience with UML 1.x have exposed several areas worthy of upgrade and include improvement for behavioral specification, as well as user extensibility. The eagerly awaited improvements of UML 2.0, however, do not eradicate UML fever, nor do they minimize susceptibility to it. Intelligent usage of technology, observing a good software development process, and having experienced people with the proper skill mix are as critical to success now as in the days before the UML.

The entertaining tenor of "Death by UML Fever" should not be inferred to suggest that UML fever is an imaginary ailment. It is genuinely real, it is thriving, and its presence is causing cost and schedule trauma on many software programs right now. Furthermore, the root causes of this fever, in general, have nothing to do with the UML itself: Rather, this fever and its various manifestations are largely symptoms of deeper ills in an organization's software development practices. Software organizations should consider launching self-diagnosis campaigns to assess the presence or extent of UML fever on their programs and plan rehabilitation strategies as necessary. Developing good software is a difficult enough task without having to endure the preventable and often painful complications of the dreaded UML fever.

GRADY BOOCH is one of the original developers of UML and is recognized for his innovative work on software architecture, modeling, and software engineering processes.