Why the Vasa Sank:
10 Problems and Some Antidotes for Software Projects

Richard E. Fairley, *Oregon Health and Sciences University*
Mary Jane Willshire, *University of Portland*

On 10 August 1628, the Royal Swedish Navy's newest ship set sail on its maiden voyage. The Vasa sailed about 1,300 meters and then, in a light gust of wind, capsized in Stockholm's harbor, losing 53 lives. The ship was Sweden's most expensive project ever, and a total loss. This was a major national disaster—Sweden was at war with Poland and needed the ship for the war effort. A formal hearing the following month did not determine why the ship sank, and no one was blamed.

In 1628, the Royal Swedish Navy launched the Vasa. After sailing only about 1,300 meters, it sank. The authors review the project's problems, including why the ship was unstable and why it was still launched. They interpret the case in terms of today's large, complex software projects and present some antidotes.

After initial salvage attempts, the ship was largely forgotten until Anders Franzen located it in 1956.[1] In 1961, 333 years after it sank, the Vasa was raised; it was so well preserved that it could float after the gun portals were sealed and water and mud were pumped from it. The sheltered harbor had protected the ship from storms, and the Baltic Sea's low salinity prevented worms from infesting and destroying the wooden vessel. Today it is housed in the Vasa Museum (www.vasamuseet.se), near the site where it foundered.[2] Figures 1 and 2 show the restored ship and a recreation of its sinking.

Researchers have extensively analyzed the Vasa and examined historical records concerning its construction. It sank, of course, because it was unstable. The reasons it was unstable, and launched when known to be unstable, are numerous and varied. Although we may never know the exact details surrounding the Vasa, this article depicts our "most probable scenario" based on the extensive and remarkably well-preserved documents of the time, evidence collected during visits to the Vasa Museum, information from the referenced Web sites, and publications by those who have investigated the circumstances of the Vasa's sinking. The problems encountered are as relevant to our modern-day attempts to build large, complex software systems as they were to the 17th-century art and craft of building warships.

The Vasa story

The story of the Vasa unfolds as follows.

Changing the shipbuilding orders

On 16 January 1625, Sweden's King Gus-

0740-7459/03/$17.00 © 2003 IEEE

tav II Adolf directed Admiral Fleming to sign a contract with the Stockholm ship builders Henrik and Arend Hybertsson to design and oversee construction of four ships. Henrik was the master shipwright, Arend the business manager. They subcontracted with shipbuilder Johan Isbrandsson to build the ships under their direction over a period of four years. Two smaller ships were to have about 108-foot keels; the two larger ones, about 135-foot keels.

Based on a series of ongoing and confusing changes the King ordered during the spring and summer of 1625, Henrik requested oak timbers be cut from the King's forest for two 108-foot ships and one 135-foot ship. On 20 September, the Swedish Navy lost 10 ships in a devastating storm. The king then ordered that the two smaller ships be built first on an accelerated schedule to replace two of the lost ships. Construction of the Vasa began in early 1626 as a small, traditional ship; it was completed two and a half years later as a large, innovative ship, after undergoing numerous changes in requirements.

On 30 November 1625, the King changed his order, requiring the two smaller ships to be 120 feet long so that they could carry more armament: 32 24-pound guns in a traditional enclosed-deck configuration.[1] ("24 pounds" refers to the weight of the shot fired by the cannon. A 24-pounder weighed approximately 3,000 pounds.) Henrik inventoried materials and found that he had enough timber to build one 111-foot ship and one 135-foot ship. Under the King's direction, as conveyed by Admiral Fleming, Henrik laid the keel for a 111-foot ship because it could be completed more quickly than the larger one. The records are unclear as to whether the keel was initially laid for this size or initially for a 108-foot ship and then extended to 111 feet.

No specifications for the modified keel

After the Vasa's 111-foot keel had been laid, King Gustav learned that Denmark was building a large ship with two gun decks. The King then ordered the Vasa to be enlarged to 135 feet and include two enclosed gun decks (see Figure 3). No one in Sweden, including Henrik Hybertsson, had ever built a ship with two enclosed gun decks. Because of schedule pressure, the shipbuilders thought that scaling up the 111-foot keel using materials planned for the bigger ship would be more expeditious than laying a new 135-foot keel.

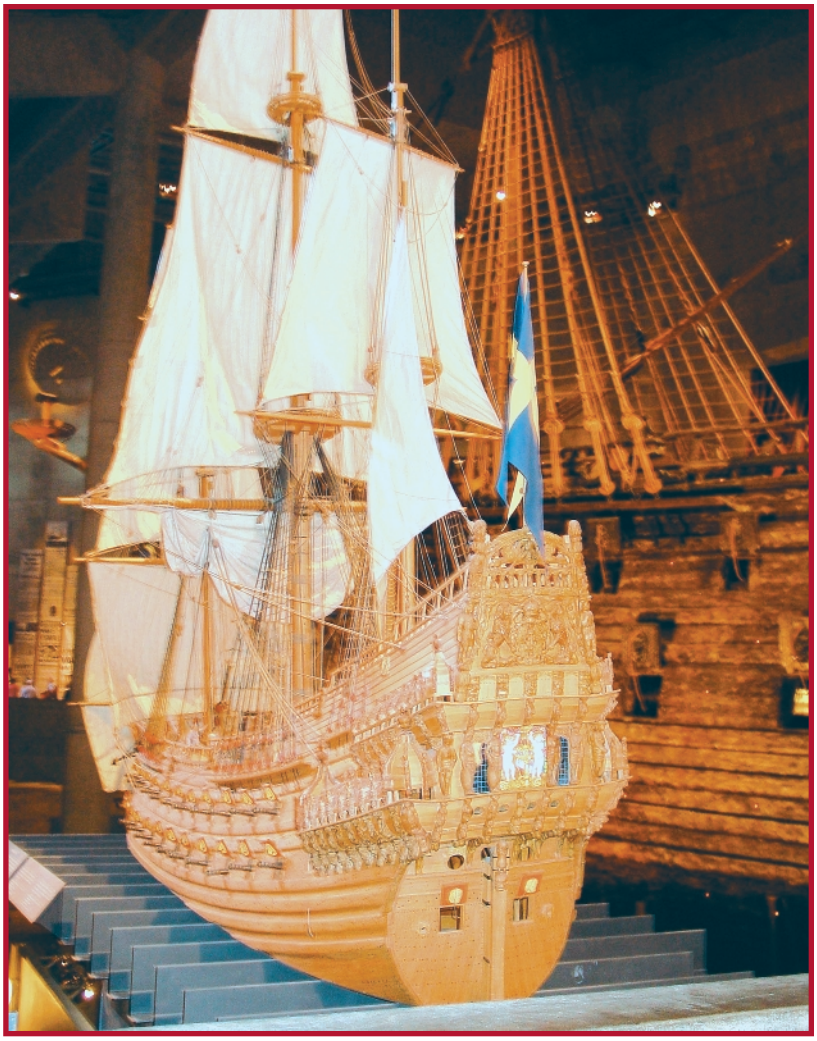


Figure 1. The restored Vasa, housed in Stockholm's Vasa Museum (model in foreground, Vasa in background).

The evolution of warship architecture from one to two enclosed gun decks in the early 1600s marked a change in warfare tactics that became commonplace in the late 1600s and 1700s. The objective became to fire broadside volleys and sink the opponent. Before that, warships fired initial cannon volleys to cripple their opponent's ship so that they could board and seize it. To this end, earlier warships carried large numbers of soldiers (as many as 300).

Although the contract with the Hybertssons was revised (and has been preserved), no one has ever found specifications or crude sketches for either the 111-foot or 135-foot Vasa, and none of the related (and well-preserved) documents mentions such drawings. It is unlikely that anyone spent time preparing specifications, given the circumstances and schedule pressure under which the Vasa was constructed. They probably would not have been prepared for

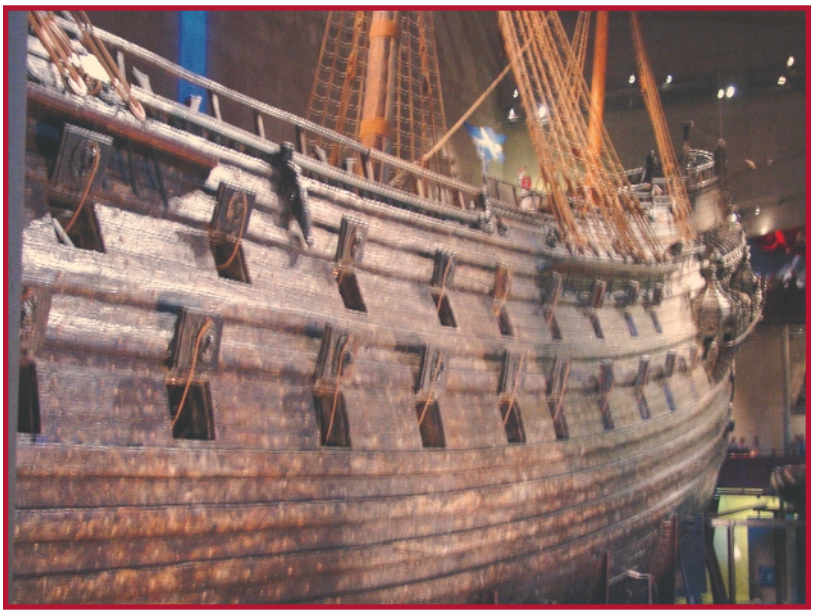
Figure 2. A recreation of the Vasa disaster. (photo by Hans Hammarskiöld)

the original 108-foot ship, because these types of ships had been routinely built for many years and Hybertsson was an experienced shipwright, working with an experienced shipbuilder. Henrik Hybertsson probably "scaled up" the dimensions of the original 108-foot ship to meet the length and breadth requirements of the 111-foot ship and then scaled those up for the 135-foot version of the Vasa.

How he scaled up the 111-foot Vasa to 135 feet was constrained by its existing keel, which contained the traditional three scarfs. He added a fourth scarf to lengthen the keel, but the resulting keel is thin in relation to its length, and its depth is quite shallow for a ship of this size.

Hybertsson's assistant, Hein Jacobsson, later said that the Vasa was built one foot, five inches wider than originally planned to accommodate the two gun decks. However, the keel was already laid, so they could make that change in width only in the upper parts of the ship. This raised the center of gravity and contributed to the Vasa's instability (sailing ships are extremely sensitive to the location of the center of gravity; a few centimeters can make a large difference). Also, those outfitting the ship for its first voyage found that the shallow keel did not provide enough space in the hold for the amount of ballast needed to stabilize a 135-foot ship. The keel's thinness required extra bracing timbers in the hold, further restricting the space available for ballast.

Figure 3. The Vasa's two gun decks.



Shifting armaments requirements

The numbers and types of armaments to be carried by the scaled-up Vasa went through a number of revisions. Initially, the 111-foot ship was to carry 32 24-pound guns. Then, the 135-foot version was to carry 36 24-pound guns, 24 12-pound guns, eight 48-pound mortars, and 10 smaller guns. After a series of further revisions, the Vasa was to carry 30 24-pounders on the lower deck and 30 12-pounders on the upper deck. Finally, the King ordered that the Vasa carry 64 24-pound guns—32 on each deck—plus several smaller guns (some documents state the required number as 60 24-pound guns).

Mounting only 24-pound guns had the advantage of providing more firepower and allowed standardization on one kind of ammunition, gun carriage, powder charge, and other fittings. However, the upper deck had to carry the added weight of the 24-pound guns in cramped space that had been built for 12-pound guns, further raising the ship's center of gravity. In the end, the Vasa was launched with 48 24-pound guns (half on each deck), because the gun supplier's manufacturing problems prevented delivery of more guns on schedule. Waiting for the additional guns would have interfered with the requirement

to launch the ship as soon as possible. Another indication of excessive schedule pressure is that the gun castings were of poor quality. They might well have malfunctioned (exploded) during a naval battle.

Artisan shipbuilders constructed the Vasa's rigging and outfitting without explicit specifications or plans, in the traditional manner that had evolved over many years. The King ordered that the ship be outfitted with hundreds of ornate, gilded, and painted carvings depicting Biblical, mythical, and historical themes (see Figure 4). The Vasa was meant to impress by outdoing the Danish ship being built; no cost was spared, making the Vasa the most expensive ship of its time. However, the heavy oak carvings raised the center of gravity and further increased instability.

Figure 4. The ship's ornate, gilded carvings.

The shipwright's death

Henrik Hybertsson became seriously ill in 1626 and died in 1627, one year before the Vasa was completed. During the year of his illness, he shared project supervision with Jacobsson and Isbrandsson, but according to historical records, project management was weak. Division of responsibility was not clear and communication was poor; because there were no detailed specifications, schedule milestones, or work plans, it was difficult for Jacobsson to understand and implement Hybertsson's undocumented plans. Communication among Hybertsson, Jacobsson, and Isbrandsson was poor. This resulted in further delays in completing the ship.

Admiral Fleming made Jacobsson (Hybertsson's assistant) responsible for completing the project after Hybertsson's death. At this time and during the subsequent year, 400 people in five different groups worked on the hull, carvings, rigging, armaments, and ballasting—apparently with little, if any, communication or coordination among them. This was the largest work force ever engaged in a single project in Sweden up to that time. There is no evidence that Jacobsson prepared any documented plans after becoming responsible for completing the ship.

No way to calculate stability, stiffness, or sailing characteristics

Methods for calculating the center of gravity, heeling characteristics, and stability factors for sailing ships were unknown, so ships' captains had to learn their vessels' operational characteristics by trial-and-error testing. The Vasa was the most spectacular, but certainly not the only, ship to capsize during the 17th and 18th centuries.

Measurements taken and calculations performed since 1961 indicate that the Vasa was so unstable that it would have capsized at a heeling over of 10 degrees; it could not have withstood the estimated wind gust of 8 knots (9 miles per hour) that caused the ship to capsize.[3] Recent calculations indicate the ship would have capsized in a breeze of 4 knots.

That the wind was so light during the Vasa's initial (and final) cruise is verified by the fact that the crew had to extend the sails by hand upon launch. Lieutenant Petter Gierdsson testified at the formal inquiry held in September 1628: "The weather was not strong enough to pull out the sheets, although the blocks were well lubricated. Therefore, they had to push the sheets out, and one man was enough to hold a sheet."[4]

During the formal inquiry, several witnesses commented that the Vasa was "heavier above than below," but no one pursued the questions of how and why the Vasa had become top-heavy. No one mentioned the weight of the second deck, the guns, the carvings, or other equipment. In those days, most people (including the experts) thought that the higher and more impressive a warship, and the more and bigger the guns it carried, the more indestructible it would be.

Table 1

Ten software project problems and some antidotes

Problem area	Antidotes
1. Excessive schedule pressure	Objective estimates
	More resources
	Better resources
	Prioritized requirements
	Descoped requirements
	Phased releases
2. Changing needs	Iterative development
	Change control/baseline management
3. Lack of technical specifications	Development of initial specifications
	Event-driven updating of specifications
	Baseline management of specifications
	A designated software architect
4. Lack of a documented project plan	Development of an initial plan
	Periodic and event-driven updating
	Baseline management of the project plan
	A designated project manager
5. & 6. Excessive and secondary innovations	Baseline control
	Impact analysis
	Continuous risk management
	A designated software architect
7. Requirements creep	Initial requirements baseline
	Baseline management
	Risk management
	A designated software architect
8. Lack of scientific methods	Prototyping
	Incremental development
	Technical performance measurement
9. Ignoring the obvious	Back-of-the-envelope calculations
	Assimilation of lessons learned
10. Unethical behavior	Ethical work environments and work cultures
	Personal adherence to a code of ethics

The failed prelaunch stability test

Captain Hannson (the ship's captain) and a skeleton crew conducted a stability test in the presence of Admiral Fleming during outfitting of the Vasa. The "lurch" test consisted of having 30 men run from side to side amidship. After three traversals by the men, the test was halted because the ship was rocking so violently it was obvious it would capsize if the test were not halted. The ship could not be stabilized because there was no room to add ballast under the hold's floorboards (see Figure 5). In any case, the additional weight would have placed the lower-deck gun portals near or below the ship's waterline. The Vasa was estimated to be carrying about 120 tons of ballast; it would have needed more than twice that amount to stabilize.

That the Vasa was launched with known stability problems was the result of poor communication, pressure from King Gustav to launch the ship as soon as possible, the fact that the King was in Poland conducting a war campaign (and thus unavailable for consultation), and the fact that no one had any suggestions for making the ship more stable.

Testimony at the formal hearing indicated that Jacobsson, the shipwright, and Isbrandsson, the shipbuilder, were not present during the stability test and were unaware of the outcome. The boatswain, Matsson, testified that Admiral Fleming had accused him of carrying too much ballast, noting that "the gunports are too close to the water!" Mattson then claimed to have answered, "God grant that the ship will stand upright on her keel." To which the Admiral replied, "The shipbuilder has built ships before and you should not be worried."[4]

Whether Admiral Fleming and Captain Hannson intentionally withheld the stability test results is a matter for speculation. The King had ordered that the Vasa be ready by 25 July and "if not, those responsible would be subject to His Majesty's disgrace." The Vasa's maiden voyage on 10 August was more than two weeks later. It was reported that after the failed stability test, Admiral Fleming lamented, "If only the King were here."[5]

Ten problem areas and their antidotes

Many of the Vasa project's problems sound familiar to those who have grappled with large software projects. Table 1 presents 10 problems from the Vasa project that are also common to software projects, along with some antidotes for software projects.

1. Excessive schedule pressure

Many software projects, like the Vasa project, are under excessive schedule pressure to meet a real or imagined need. According to Fred Brooks, more software projects have failed ("gone awry") for lack of adequate calendar time than for all other reasons combined.[6] Excessive schedule pressure in software projects is caused by

- Truly pressing needs
- Perceived needs that are not truly pressing
- Changing of requirements without adjusting the schedule or the resources (see Problem 2)

- Unrealistic schedules imposed on projects by outside forces
- Lack of realistic estimates based on objective data

When schedule estimates are not based on objective data, software engineers have no basis for defending their estimates or for resisting imposed schedules. You can sometimes meet schedule requirements by increasing project resources, applying superior resources, descoping the (prioritized) requirements, phasing releases, or some combination of these antidotes.

2. Changing needs

Some common reasons for changes to software requirements include competitive forces (as in the case of the Vasa), user needs that evolve over time, changes in platform and target technologies, and new insights gained during a project. There are two techniques for coping with changing software requirements:

- Pursuing an iterative development strategy (for example, incremental, evolutionary, agile, or spiral)
- Practicing baseline management

You can use these techniques alone or together. In the case of iterative development, you can accommodate requirements changes based on priority within the constraints of time, resources, and technology—any of which you can adjust as the project evolves. When using baseline management, you initially place the requirements under version control. Approved changes result in a new version of the requirements (a new baseline), which is accompanied by adjustments to schedule, resources, technology, and other factors as appropriate. The goal of both approaches is to maintain, at all times, a balance among requirements, schedule, resources, technology, and other factors that might pose risks to successful delivery of an acceptable product within the project constraints.

3. Lack of technical specifications

Software projects, like the Vasa project, sometimes start as small, familiar activities for which technical specifications are deemed unnecessary. These projects often grow to become large, innovative ones. In the Vasa's case, verbal communication might have compensated somewhat for the lack of written specifications and a written project plan. In the case of software projects, initially developing and baselining requirements, even for a small, simple project, is much easier (and less risky) than trying to "reverse-engineer" requirements later, when the project has crossed a complexity threshold that warrants developing technical specifications.

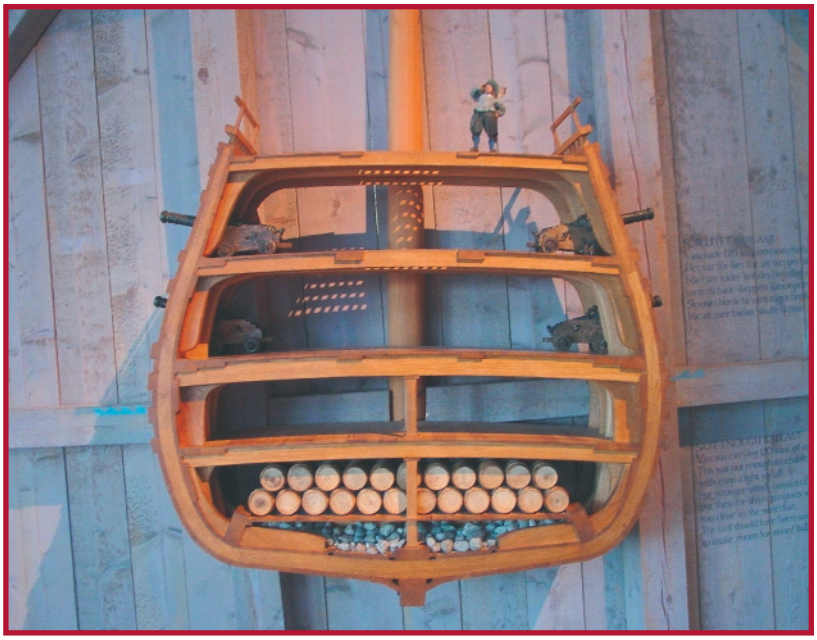


Figure 5. A cross section of the Vasa showing its shallow keel and its multiple decks, including the two gun decks.

4. Lack of a documented project plan

Because its designers saw the Vasa as a small, familiar ship and because the project team was experienced in building these types of ships, they might have thought systematic planning was an unnecessary use of time and resources. Software projects often start under similar circumstances. As in the case of requirements, evolving a baselined project plan for an initially small project is much easier than trying to construct a project plan later (when there is no time to do so). A software project plan must include

- Decomposition of the work to be done (that is, a *work breakdown structure*)
- Allocation of requirements to the work elements of the WBS
- An allocated schedule (for example, a milestone chart)
- Allocation of resources to each schedule increment
- Deliverable work products for each schedule increment

In modern society, the role of engineering is to provide systems and products that enhance the material aspects of human life, thus making life easier, safer, more secure, and more enjoyable.

- Plans for acquiring software from vendors and open sources
- Plans for managing subcontractors
- A risk management plan
- A clear statement of authorities and responsibilities[7]

You can document the project plan for a small software project in a few pages and then expand it as the project grows.

5. Excessive innovation

In *To Engineer Is Human*, Henry Petroski observes that engineering projects often fail when people attempt innovations beyond the state of the art.[8] Software projects are always innovative to some extent because replicating existing software, unlike replicating physical artifacts, is a trivial process. Software projects are conducted to develop new systems "from scratch" and to produce new versions of existing systems, but not to replicate software. To control excessive innovation in software projects, you can apply the following techniques:

- Baseline control of working documents (for instance, requirements, project plans, design documents, test plans, and source code)
- Impact analysis of proposed changes
- Continuous risk management

6. Secondary innovations

Reasons for secondary innovations in software projects include the need to accommodate the constraints of the technologies used, the addition of derived requirements to support primary requirements and changes to them, and innovations based on the developers' creative impulses. As in the Vasa's case, these secondary requirements can overwhelm a software project. In addition to the techniques mentioned for controlling Problem 5, you should assign responsibility and give authority to a designated software architect to maintain the software product's vision and integrity, which is an additional antidote for many other problems.[7]

7. Requirements creep

It seems that no one was aware of the overall impact of the changes made to the Vasa during the two and a half years of construction. This can (and does) easily happen to software projects. Fundamental techniques for maintaining control of software projects are developing initial documentation and consistently updating requirements and plans to maintain an acceptable balance among requirements, schedule, and resources as the project evolves. Often-heard excuses for failure to establish initial project documentation are lack of sufficient knowledge to do so and the belief that "everything will change anyway, so why plan?" You should develop initial requirements and plans to the extent possible in the face of imperfect knowledge, with the expectation that they will change and with procedures in place to evolve them systematically. Failure to update initial requirements and plans periodically, and as events require, is often the result of a perception that there is not enough time to do so ("never enough time to do it right, but always time to redo it"). This perception, in turn, comes from insufficient procedures for managing requirements and plans on a continuing basis and a lack of appreciation for the risks thus created.

8. Lack of scientific methods

Because software has no physical properties, you cannot calculate many traditional engineering parameters for software (at least, at this time). Unlike physical artifacts such as the Vasa, however, you can build a software system in small, incremental steps and monitor the evolution of technical parameters such as memory usage, performance, safety, security, and reliability as the system evolves.

9. Ignoring the obvious

In the case of the Vasa, the lurch test demonstrated that the ship was dangerously unstable. There was no room for additional ballast. If there had been room, the added weight would have placed the lower gun portals below the waterline. Although we lack scientific methods in many software engineering areas, we can often avoid foreseeable disasters by performing a few, often quite simple, "back of the envelope" calculations. If, for example, a certain transaction processing system must process 1,000 transactions per second and if each transaction requires four complex queries, the system must process 4,000 complex queries per second, including the time required for con-

text switching among transactions (that is, 250 microseconds per context switch and transaction). Accounting for system latencies might show the envisioned system to be infeasible.

10. Unethical behavior

In modern society, the role of engineering is to provide systems and products that enhance the material aspects of human life, thus making life easier, safer, more secure, and more enjoyable. Technological innovation often involves ethical considerations. In the Vasa's case, the goal was to make Sweden more secure for its citizens and to bring glory to the country. At the last moment, when it became obvious the ship was not seaworthy, those with authority to stop the launch did not do so.

In our time, software engineers should first serve the public; second, be advocates for the customers and users of their products; and third, act in the interest of their employers, in a manner that is consistent with the public interest. Software engineers should "accept full responsibility for their work" and "approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy, or harm the environment. The ultimate effect of the work should be to the public good."[9] Every software engineer should be familiar with and adhere to a code of ethics.

Table 1 presents only some of the many possible antidotes for software projects. Some antidotes listed in one problem area apply equally to other problem areas. For the sake of brevity, we leave elaboration of Table 1 to readers.

According to the formal Vasa hearing's transcript, no one asked how or why the ship had become unstable or why it was launched with known stability problems. The failure of this line of inquiry is perhaps the most compelling problem from the Vasa case study, as is our often-observed, present-day failure to learn from our mistakes in software engineering.

On a positive note, large, two-deck warships were subsequently built and sailed during the latter 17th and the 18th and 19th centuries. In the same way, we can now, with reasonable confidence of success, build larger and more complex software-intensive systems than in the past.

Acknowledgment

The reviewers and editor offered many valuable recommendations for improvements to this article's initial draft. Photos by permission of the Vasa Museum.

References

1. A. Franzen, *The Warship Vasa: Deep Diving and Marine Archaeology*, 6th ed., Norstedt & Soners Forlag, Stockholm, 1974.
2. *The Royal Ship Vasa*, http://home.swipnet.se/~w-70853/WASAe.htm.
3. C. Borgenstam and A. Sandstrom, *Why Vasa Capsized*, AB Grafisk Press, Stockholm, Sweden, 1995.
4. *The Story of the Royal Swedish Man-of-War Vasa*, multimedia documentary, VyKett AB, Stockholm, Sweden, 1999; available (in English) on floppy disk by email inquiry to the Vasa Museum at vasamuseet@sshm.se.
5. A. Wahlgren, *The Warship Vasa*, a documentary film, Vasa Museum, Stockholm, Sweden, 1996; available (in English) on a VCR tape by email inquiry to the Vasa Museum at vasamuseet@sshm.se.
6. F. Brooks, *The Mythical Man-Month*, Addison-Wesley, Boston, 1995.
7. *IEEE Std. 1058-1998, Standard for Software Project Management Plans*, IEEE, Piscataway, N.J., 1998.
8. Henry Petroski, *To Engineer Is Human: The Role of Failure in Successful Design*, Vintage Books, New York, 71992.
9. *Software Engineering Code of Ethics and Professional Practice*, IEEE Computer Society and ACM Joint Task Force on Software Engineering Ethics and Professional Practices, 1999, http://computer.org/tab/code11.htm.

About the Authors

Richard E. Fairley is a professor of computer science and associate dean of the OGI School of Science & Engineering of the Oregon Health & Science University. He also participates in the Oregon Master of Software Engineering program, which is offered collaboratively by four Oregon universities. His research interests include requirements engineering, software architecture, software process engineering, estimation, measurement and control of software projects, risk management, and software engineering education. He is a member of the IEEE Computer Society and ACM. Contact him at the OGI School of Science and Eng., 20000 NW Walker Rd., Beaverton, OR 97006; dfairley@cse.ogi.edu.

Mary Jane Willshire is an associate professor of computer science in the Electrical Engineering and Computer Science Department at the University of Portland, where she teaches courses and conducts research in software engineering, human-computer interfaces, and database technology. She is a member of the IEEE, ACM, Association for Women in Science, and Society of Women Engineers. Contact her at the School of Engineering, Univ. of Portland, 5000 N. Willamette Blvd., Portland, OR 97203-5798; willshir@up.edu.