The 1992 London Ambulance Service Computer Aided Dispatch System Failure

Erich Musick

Abstract— **This paper goes beyond a simple glance at the immediate results of the 1992 failure of the London Ambulance Service's computer aided dispatch system and explores the sequence of events leading up to the incident in attempt to determine professional responsibility and how the project might have benefited from a more formal specification of the software system.**

I. Introduction

WHILE failures in software are perhaps one of the more quickly excused types of defects (after all, it is *surely* much easier to forget to free memory than it is to miscalculate the amount of weight a bridge can hold), they are not immune to causing immense amount of damage. The integration of software into millions of modern electromechanical devices, combined with the fallibility of its developers allows room for a wide margin of error.

The London Ambulance Service's (LAS) 1992 computer aided dispatch (CAD) software system failure is one instance of the considerable negative effect that a small error in software can have on a large population of people. A careful examination of the events surrounding the incident, however, suggests that there was more to the issue than just an error in the software. Rather, the overall carelessness with which the application's development was approached from its conception set the stage for such a grand failure.

II. The London Ambulance Service

A. Overview

In 1992, the LAS provided ambulance service to 6.8 million people living in a 600 square mile area. Of its 318 emergency ambulances, an average of 212 were in service at any given time, in addition to 445 transport ambulances, one helicopter, and a motorcycle response unit. A total of 70 ambulance stations which employed 2746 staff members housed these vehicles. The entire system was managed from a central location at the LAS Headquarters in Waterloo. On average, between 2000 and 2500 calls were received daily, 60% of which were requests for emergency services. The LAS devoted 55% of its staff and 76% of its budget specifically to emergency response [2]. According to Wikipedia, the LAS remains the "largest ambulance service in the world that does not directly charge its patients for its services" [4].

B. Existing System

In the mid-1980s, the LAS emergency dispatch system was run completely manually and consisted of three main tasks:

1) Call Taking : When a call requesting emergency ambulance service was received, the call taker filled out a paper form, found the caller's location on a map, and then put the form on a conveyor belt [2].

2) Resource Identification : The conveyor belt took the forms to another LAS employee who analyzed the locations and statuses of ambulances in the call's region. He then assigned the call to an available unit and wrote the assignment on the form [2].

3) Resource Mobilization : Finally, a dispatcher made contact with the ambulance to which the call was assigned and provided the operator(s) with the details of the call [2].

Looking back on this system from the perspective of 2006, in which computers dominate everyday life, a manual system that relies on people's memories, paper, and human reasoning for optimal resource utilization seems silly. Indeed, even at the time, some of these inefficiencies were acknowledged and a new, computer-based system was proposed [2].

C. Proposed System

The government-imposed stipulation that calls be responded to within three minutes, in addition to the obvious deficiencies of the manual dispatch system led those in charge of the LAS seek out a computer-based alternative [2]. The proposed system's highlights consisted of an automatic vehicle locating system (AVLS) and mobile data terminals (MDTs) that would be placed inside emergency vehicles and would facilitate communication using a computer terminal [1].

The LAS looked at adopting an existing system but found problems with each of the available options. LAS management decided to create a new one and proceeded to gather requirements without receiving input from ambulance crews or dispatchers [2].

Management set big goals. Rather than simply assisting dispatchers, this completely computerized system would do nearly everything automatically. The simple sequence of steps performed by a person would be to answer the phone, enter incident data into a computer terminal, and then respond if the system displayed exception messages resulting from no ambulances being available for longer than 11 minutes. The locations of calls would be mapped by the software. The system would then use this map data and the location and status details provided by the AVLS to find and dispatch the available ambulance closest to the incident's location. To top it off, this system would be implemented not incrementally, but all at once [2].

III. The Failure

A. What Happened

After a whole slew of issues, including a project cancellation and re-design, a software system got developed and was deployed the morning of October 26, 1992 [2]. Just a few hours later, however, problems began to arise. The AVLS was unable to keep track of the ambulances and their statuses in the system. It began sending multiple units to some locations and no units to other locations. The efficiency with which it assigned vehicles to call locations was substandard. The system began to generate such a great quantity of exception messages on the dispatchers' terminals that calls got lost. The problem was compounded when people called back additional times because the ambulances they were expecting did not arrive. As more and more incidents were entered into the system, it became increasingly clogged. The next day, the LAS switched back to a part-manual system, and shut down the computer system completely when it quit working altogether eight days later [1].

Because of the large area serviced by the LAS, many people were directly affected by the computer system failure. There were as many as 46 deaths that would have been avoided had the requested ambulance arrived on time. One heart attack patient waited six hours for an ambulance before her son took her to the hospital. Four hours after that, the LAS called to see if the ambulance was still needed. Another woman called the LAS every 30 minutes for almost three hours before an ambulance arrived. It was too late, as her husband had already died. One ambulance crew arrived only to find that the patient had not only died, but his body had been taken away by a mortician [2]

B. Immediate Causes

At the time the system went live, there were 81 known issues with the software and no load-tests had been run. No provisions for a backup system had been made. While the gap of 10 months between the time dispatchers were first trained to use the software and when it was deployed played its role in the disaster, the software had three primary flaws that immediately caused the failure:

1) Imperfect Data : The software system did not function well when given invalid or incomplete data regarding the positions and statuses of ambulances [2].

2) Interface Issues : The deployed system had a wide variety of quirks in different parts of the user interface. For example, parts of the MDT terminal screens had black spots, thus preventing ambulance operators from getting all information needed. When ambulance crews attempted to remedy mistakes after pushing incorrect wrong buttons, the system did not accept the fix. Again, the software failed to compensate for the error conditions that occur in normal, day-to-day operation [2].

3) Memory Leak : The root cause of the main breakdown of the system, however, was a memory leak in a small portion of code. This defect retained memory that held incident information on the file server even after it was no longer needed. As with any memory leak, after enough time, the memory filled up and caused the system to fail [2].

IV. Deeper Causes

IEEE member and risk management expert Robert Charette suggests, "Bad decisions by project managers are probably the single greatest cause of software failure today" [3]. While the software controlling the LAS's CAD system had some key flaws, without which the system may not have failed to the extent it did, the events surrounding the failure, the state of the LAS as an organizational entity, and the process with which the LAS approached the development of the CAD played just as large, if not a larger role in the system's failure.

Getting the LAS CAD system to the point of deployment was challenging and opened the doors for failure to creep in at several points. The original design of the LAS's CAD system was first proposed in 1987. It was modified in 1989 and then, due to gross overspending by about 300%, the project was canceled in 1990. However, the national mandate to reduce emergency response times pushed the LAS to look into a computerized system once again. [2].

A. Hardware Reuse

When the project was re-opened, those LAS individuals in charge of it approached it with the primary goal to save as much time and money as possible. While this is a reasonable aim of any project, it held too much weight in the LAS's CAD project. As part of the attempt to save money, the LAS decided to reuse some of the hardware that had already been purchased when working on the failed project instead of purchasing hardware that was either more up-to-date or more suitable for the new system [2].

B. Vendor Selection

The two people who were primarily in charge of choosing a software vendor for creating the system were "a manager expecting to become redundant and a contractor who was a temporary addition to the organisation." These individuals' roles and their lack of stake in the project draw in to question their ability to select the best company for the job. Further, the selection committee weighed a bid's price as the most important factor in selecting a vendor. All companies that submitted bids greater than £1.5 million were immediately thrown out. This is an unreasonably low price, especially considering the fact that even after £7.5 million had been poured into the previous attempt at a CAD system, the project had failed [2].

Selection of a developing organization was further constrained by the requirement that the project be completed in 11 months. Once again, any bids not meeting this hard constraint were not considered. Several companies proposed modified deployment schedules in which some functionality would be delivered after the 11 month deadline and the rest a year later. These, too, were thrown out. [2].

The LAS accepted a bid of just under £1 million that was submitted by a conglomeration of companies. The software

portion of the system was "offered as a throw-in in a hardware deal" for a meager £35 thousand and was completed by a company called System Options. The fact that the majority of the cost for this package which relied heavily on software went instead toward hardware should have raised a flag that something was askew [2].

Further, System Options was declared the project lead. Though it had developed many smaller software packages, System Options had never worked on such a large project and had absolutely no past experience with "real-time, safety-critical, command and control systems." Its inexperience developing such software led the contractor selection committee to raise concerns over the company's ability to perform the task at hand. However, even though these concerns were further substantiated by an audit of the selection process, the LAS hired the company [2].

C. Design, Requirements, and Specifications

The process through which requirements were gathered and specifications and designs were written had several major flaws. First, it seems that a LAS team - not a software development team - went through this process without attempting to consult ambulance operators, dispatchers, and other key users of the system. A basic understanding of the software requirements process suggests that leaving out key stakeholders is detrimental to the project and will result in an incomplete set of requirements. Further, the requirements document was "highly detailed and extremely prescriptive," very much a direct contradiction of the requirements process' focus on the *what* as opposed to the *how*. Once the companies completing the project were selected, they were required to provide a final system design specification. The LAS incorrectly assumed that if the contractor could do this, they understood the system they had to create. However, there was no signoff on this design specification and it was updated after development began [2].

D. Flawed Software Process

Several other parts of the software process played a large role in the eventual failure of the deployed product. No quality assurance was performed, configuration management was absent, agreed-upon changes were not tracked, and test plans were not written, just to name a few. At one point in the CAD system's development, the LAS considered hiring one of the other bidders to perform some quality assurance tasks. Because this was seen as the job of the developing organization, this idea was disregarded [2]. Though any developing organization should be able to ensure its products' quality, testing is impossible when getting the project coded in the allotted 11 months is an already an unachievable goal.

V. Responsibility

While it is true that the CAD system's software errors demonstrate "carelessness and lack of quality assurance of program code changes," had the LAS paid more attention to selection process of the developing organization and imposed more reasonable and realistic expectations, the CAD system likely would not have failed as it did [5]. Additional time to develop the software would have allowed the developers to more meticulously follow the software process and provide an opportunity for adequate testing of the entire system. Had the LAS selected a different company with more experience with real-time applications to develop the CAD system, the project could have benefited from a firmer foundation for the software's conception.

Although the LAS erred in its decision to choose System Options to develop the CAD software, System Options is by no means absolved of blame. Consider a surgeon who specializes in orthopedic surgery but is completely inexperienced in heart surgery. He volunteers to perform an important and difficult heart surgery in one-eighth the time it would take an experienced heart surgeon to do it. Although this doctor believes he is capable of performing the surgery, his inexperience and stringent time restraints make this a high risk surgery better suited for a surgeon trained in the specific task. Similarly, System Options' lack of background in real time systems and insufficient allotment of time to complete the project should have provided sufficient reason for company decision makers to refrain from bidding on the project.

One might argue that System Options is not responsible for the small window of development time, as it was a restriction imposed by the LAS. However, had no company suggested that successful delivery was possible in the time frame the LAS required, the LAS would have realized that it was unreasonable and either expanded the time frame or abandoned the project. System Options' acceptance of the bid implied that it was, indeed, possible to create a successful CAD solution in the time frame given.

While the LAS pushed for expedient delivery, System Options, as "professionals" in the area of software system development, had an obligation to protect the public. Knowing that the system was incomplete, untested and buggy, System Options took an enormous risk in deploying it. It failed to do its duty to sufficiently evaluate this risk and refuse to release, just as those involved with the Challenger space shuttle incident minimized a known risk and thus failed to keep the spacecraft from launching. Even if System Options had a lot riding on the timely release of the software, the loss the failure caused was far greater than any monetary investment that could ever be made.

VI. Conclusion

Though a small software error often is the straw that breaks the camel's back, the responsibility for the LAS's CAD system failure does not lie solely on the single developer who made the error or even the developing organization to which he belonged. Rather, the attitudes of key LAS members toward the project and the unreasonable restraints they placed on the project allowed the failure to occur.

References

[1] A. Finkelstein, J. Dowell, "A comedy of errors: the London Ambulance Service case study," *iwssd*, p. 2, 8th International Workshop on Software Specification and Design (IWSSD'96), 1996.

[2] D. Dalcher, "Disaster in London: The LAS Case study," *ecbs*, p. 41, IEEE Conference and Workshop on Engineering of Computer-Based Systems, 1999.
[3] R.N. Charette, "Why Software Fails," *IEEE Spectrum*, Vol. 42, Issue 9, Sept. 2005, p. 42-49.
[4] "London Ambulance Service," *Wikipedia*, Mar. 20, 2006, Available http://en.wikipedia.org/wiki/London_Ambulance_Service, Apr. 10, 2006.
[5] D. Page, P. Williams, and D. Boyd, *Report of the Inquiry into the London Ambulance Service*, Communications Directorate, South West Thames Regional Health Authority, London, Feb. 1993.