Successful Architecture for Short Message Service Center

Eltjo R. Poort[1], Hans Adriaanse[1], Arie Kuijt[2], Peter H.N. de With[1,3]

[1] LogicaCMG, P.O. Box 159, 1180 AD Amstelveen, The Netherlands
[2] LogicaCMG Telecoms, Merweplein 5, 3432 GN Nieuwegein, The Netherlands
[3] Eindhoven Univ. of Technol., P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{eltjo.poort,hans.adriaanse,arie.kuijt,peter.de.with}@logicacmg.com

Abstract

This paper presents and analyzes the key architectural decisions in the design of a successful Short Message Service Center as part of a GSM network.

1. Introduction

In the early nineties, a Short Message Service Center[1] was developed according to the specifications for text messaging embedded in the GSM standard [3]. This paper looks back at the conceptual design phase of the realization project. The paper is a practitioner's report, analyzing the key architectural decisions and distinguishing factors that contributed to the system's success.

2. System requirements

The SMSC's key requirements are listed according to the categorization presented in [4]: first the primary (functional) requirements, and then the secondary requirements, divided in secondary functional requirements and quality requirements.

2.1. Primary requirements

Figure 1 shows the SMSC system in its primary context. The main purpose of the system is [PF1:] to pass messages between mobile telephones in a GSM network , and from and to other systems [PF2:] outside of the GSM network. Messages that cannot be immediately delivered are [PF3:] temporarily stored in the system.

1 This system was developed and commercially deployed by CMG, currently LogicaCMG Telecoms. In order to protect the commercial interests of the manufacturer, the descriptions have been left at a reasonably high level of abstraction, and data are mostly not quantified.

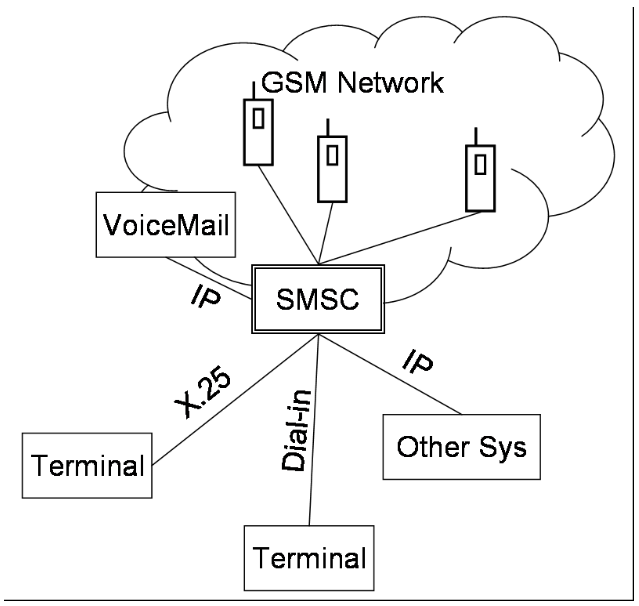

Figure 1. SMSC context.

2.2. Secondary requirements

The major secondary functional requirements were that [SF1:] a record of every message that has passed through the system is kept for billing purposes, and [SF2:] there is an interface to monitor and operate the system.

The major quality requirements set by the customers centered around [QR1:] *performance* of message throughput, [QR2:] *availability* of the messaging service and [QR3:] *reliability* of message storage. [QR4:] *Timeliness* in responses to external systems was critical. With a view to productizing of the solution, the manufacturer added requirements for [QR5:] *extensibility* and [QR6:] *scalability* of the solution.

3. Key architectural design decisions

In order to fulfill the requirements set out above, the architects made some design choices that distinguished the system from other similar systems in three major aspects: *platform choice*, *storage strategy* and *interprocess communication*.

3.1. Platform choice

The main choice to be made with respect to the platform for the SMSC was between a traditional "telecom switch" platform and an IT platform. Even though the telecom switch platforms were better rated in terms of performance [QR1], availability [QR2] and reliability [QR3], IT platforms were deemed superior in terms of extensibility at a reasonable cost [QR5].

At the time of the design of the SMSC, the most popular platforms for these kinds of medium-high performance requirements were Unix environments. The development team, however, also had ample experience with OpenVMS platforms. It was felt that the OpenVMS platform would better be able to fulfill the timing requirements [QR4].

3.2. Storage strategy

The performance of the system [QR1] was important and was perceived to become more important later on [QR6]. For this reason, it was decided to use a system where messages were stored in memory and on disk in parallel. The permanent message store mechanism is based on proprietary OpenVMS file I/O. If a more conventional storage strategy would have been used, such as an RDBMS, the added resource usage needed to perform the more complex file operations would have made it harder to fulfill the performance requirements [QR1]. Thus, the chosen storage strategy provided a better fit with the non-functional requirements.

3.3. Interprocess communication

A process architecture over multiple nodes was necessary to fulfill the performance and scalability requirements [QR1,QR6], resulting in a need for transparent communication between processes (IPC) running on different hardware units. It was felt that using the commercial-off-the-shelf IPC products available at the time would cause problems fulfilling the performance and flexibility [QR5] requirements. The team decided to develop a mean-and-lean transparent IPC itself. The resulting utility was christened VIQ (Virtual Interprocess Queue).

4. Conclusions and discussion

In the years following delivery of the system to the first customers, demand for short message services grew spectacularly. In the race to keep up with this growing demand, performance and reliability turned out to be the main deciding factors. The product quickly became the world's leading SMS product in terms of number of subscribers being serviced.

The major lesson we learned from this success story is to ***beware of fashion in system design.*** In the SMSC case, key architectural choices deviated from the prevailing "fashion" at that time, because analysis indicated that the more popular practices were not the best choices to fulfill the key requirements of performance, timeliness and reliability. The deviations turned out to be the key distinguishing factors in the architecture, that led to a success story.

Practicing architects in our experience are often under pressure from managers and customers to follow trends and fashions in system design. This phenomenon can partly be attributed to personal risk management behavior: it is hard to blame a manager for making a wrong decision if many others made the same wrong decision. We frequently encounter the term "best practice" to rationalize decisions that follow trends and fashions, often without a clear trade-off analysis as to why these practices are best for that particular situation. For this reason, we prefer the term "best fit practice" to the ubiquitous "best practice".

Methods like the Cost Benefit Analysis Method [1] can help architects to present the benefits of their choices in an objective way. This can be especially helpful when arguing choices that go against prevailing "fashion ". It should, however, be kept in mind that the previously mentioned "career risk management" argument for following trends and fashions is not necessarily invalid, and risk management related quality attributes can rightfully show up in architecture evaluations [2].

References

[1] J. Asundi, R. Kazman, and M. Klein. Using economic considerations to choose among architecture design alternatives. Technical Report CMU/SEI-2001-TR-035, SEI, 2001.

[2] P. Clements, R. Kazman, and M. Klein. *Evaluating Software Architectures*. Addison Wesley, 2002.

[3] ETSI. European digital cellular telecommunications system (phase 1);technical realization of the short message service point-to-point (gsm 03.40). http://www.etsi.org, 1995.

[4] E. R. Poort and P. H. N. de With. Resolving requirements conflicts through non-functional decomposition. In *Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, June 2004.