Lessons Learned Building Reusable OO Telecommunication Software Frameworks

M. Chandra Pal **A. Vinay Babu** **A. Sadanandam**

***Abstract*— Developing complex software systems is expensive and error-prone. Object-oriented (OO) programming languages are heavily touted technologies for reducing software cost and improving software quality. When stripped of their hype, the primary benefits of OO stem from the emphasis on modularity and extensibility, which encapsulate volatile implementation details behind stable interfaces and enhance software reuse.**

***Index Terms*— Software Reuse, Framework, Software Artifacts, Object Oriented Components.**

I. INTRODUCTION

Developers in certain well-traveled domains have successfully applied OO techniques and tools for years. For instance, the Microsoft MFC GUI framework and OCX components are in fact industry standards for creating graphical business applications on PC platforms. Although these tools have their limitations, they demonstrate the productivity benefits of reusing common frameworks and components. Software developers in more complex domains like telecom have traditionally lacked standard off-the-shelf middleware components. As a result, telecom developers largely build, validate, and maintain software systems from scratch. In an era of deregulation and stiff global competition, this in-house development process is becoming prohibitively costly and time consuming. Across the industry, this situation has produced a "distributed software crisis", where computing hardware and networks get smaller, faster, and cheaper; yet telecom software gets larger, slower, and more expensive to develop and maintain.

M.Chandra Pal is with Department of Computer Science, Kakatiya University, Warangal, AP, INDIA. 506009, e-mail: pauldec1@gmail.com
A.Vinay Babu is with Department of Computer Science & Engineering., JNTU ,Hyderabad.
A.Sadanadam is with Department of Computer Science, Kakatiya University, Warangal, AP, INDIA. 506009.

II. LESSONS LEARNED BUILDING REUSABLE OO COMMUNICATION SOFTWARE FRAME-WORKS

The challenges of building distributed software has inherent and accidental complexities associated with telecom systems:

- Inherent complexity is the fundamental challenge for developing telecom software. Chief Among these is detecting and recovering from network and host failures, minimizing the impact of communication travel time, and determining an optimal partitioning of service components and workload onto processing elements throughout a network.
- Accidental complexity has limitations with tools and techniques used to develop telecom software. A common source of accidental complexity is the Wide spread use of algorithmic decomposition, which results in non-extensible and non-reusable software designs and implementations.

The lack of extensibility and reuse in-the-large is particularly problematic for complex distributed telecom software. Extensibility is essential to ensure timely modification and enhancement of services and features. Reuse is essential to leverage the domain knowledge of expert developers to avoid re-developing and re-validating common solutions to recurring requirements and software challenges. While developing high quality reusable software is hard enough, developing high quality extensible and reusable telecom software is even harder. Not surprisingly, many companies attempting to build reusable middleware fail, often with enormous loss of money, time, and market share. Those companies that do succeed, however, reap the benefits resulting from their ability to develop and deploy complex applications rapidly, rather than wrestling endlessly with infrastructure problems. Unfortunately, the skills required to successfully produce telecom middleware remain

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

something of a "black art", often locked in the heads of expert developers.

III. SUCCESSFUL REUSE-IN-THE-LARGE REQUIRES NON-TECHNICAL PREREQUISITES

Many political, economical, organizational, and psychological factors can impede successful reuse in telecom companies. We have found that reuse-in-the-large works best when (1) the marketplace is competitive (i.e., time-to-market is crucial, so leveraging existing software substantially reduces development effort), (2) the application domain is non-trivial (i.e., repeatedly developing complete solutions from scratch is too costly), and (3) the corporate culture is supportive of an effective reuse process (e.g., developers are rewarded for taking the time to build robust reusable components). When these prerequisites don't apply, we found that developers often fall victim to the "not-invented-here" syndrome and rebuild everything from scratch.

IV. ITERATION AND INCREMENTAL GROWTH IS ESSENTIAL

Expanding on the corporate culture theme, we observed that it's crucial for software managers to openly support the fact that good components, frameworks, and software architectures take time to craft and hone. For reuse to succeed in-the-large, management must have the vision and resolve to support the incremental evolution of reusable software. In general, an 80% solution that can be evolved is often preferable to trying to achieve a 100% solution that never ships. Truly useful components and frameworks are derived from solving real problems, e.g., telecommunications, medical imaging, avionics, OLTP, etc. Therefore, a time honored way of producing reusable components is to generalize from working systems and applications. In particular, resist the temptation to create ``component teams" that build reusable frameworks in isolation from application teams. We have learned the hard way that without intimate feedback from application developers, the software artifacts produced by a component team won't solve real problems and will not be reused. Apply simple solutions to complex problems that sound too good to be true typically are, For example, translating code entirely from high-level specifications or using trendy OO design methodologies and programming languages is no guarantee of success. In experience, there is simply no substitute for skilled software developers, which leads to the following final “lesson learned”.

Ultimately, reusable components are only as good as the people who build and use them. Developing robust, efficient, and reusable telecom middleware requires teams with a wide range of skills. We need expert analysts and designers who have mastered design patterns, software architectures, and communication protocols to alleviate the inherent and accidental complexities of telecom software. Moreover, we need expert programmers who can implement these patterns, architectures, and protocols in reusable frameworks and components. In experience, it is exceptionally hard to find high quality software developers. Ironically, many telecom companies treat their developers as interchangeable, "unskilled labor" who can be replaced easily.

VI. CONCLUSIONS

Developing reusable OO middleware components and frameworks is not a silver bullet. Software is inherently abstract, which makes it hard to engineer its quality and to manage its production. The good news, however, is that OO component and framework technologies are becoming main stream. Developers and users are increasingly adopting and succeeding with object-oriented design and programming.

On the other hand, the bad news is that (1) existing OO components and frameworks are largely focused on only a few areas (e.g., GUIs) and (2) existing industry standards still lack the semantics, features, and interoperability to be truly effective throughout the telecom software domain. Too often, vendors use industry standards to sell proprietary software under the guise of open systems. Therefore, it is essential for telecom companies to work with standard organizations and middleware vendors to ensure the emerging specifications, support true interoperability and define features that meet telecom software needs.

Finally, to support the standardization effort, it is crucial for us to capture and document the patterns that underlie the successful telecom software components and frameworks that do exist. Likewise, we need to rectify these patterns to guide the creation of standard frameworks and components for the telecom domain. We hope that the next generation of OO frameworks and components will be a substantial improvement for those who worked with in the past.

REFERENCES

[1] Jacobson, M. Griss, P. Jonsson, “Software Reuse: Architecture, Process and Organization for Business Success”, ACM Press, 1997.
[2] R.E.Johnson, B. Foote, "Designing Reusable Classes", Journal of Object-Oriented Programming, 1(3):26-49, July-Aug, 1998.
[3] K.Kang, S. Cohen, J. Hess, W. Novak, A. Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study (CMU/SEI-90-TR-21,ADA 235785). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1990
[4] Gib.T. The Principles of Software Engineering Management. Addison-Wesley, 1998.
[5] Graham L., Henderson-Sellers, B., Younessi, H.: The OPEN Process Specification. Addison-Wesley,1997.
[6] Griss M.L.: Software Reuse: From Library to Factory. IBM Systems Journal, November-December 1993, 32(4), pp. 548-566.

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[7] Cooper, D.R., Schindler, P.S.: Business Research Methods. McGraw-Hill International edition. 7th Edition, 2001.
[8] Creswell, J.W.: Research Design, Qualitative and Quantitative Approaches. Sage Publications, 1994.
[9] Creswell, J.W.: Research Design, Qualitative and Quantitative Approaches. Sage Publications, 1994.
[10] Gilb, T., Graham, D.: Software Inspection. Addison-Wesley, 1993.
[11] Jacobson, I., Christenson, M., Jonsson, P., Övergaard, G.: Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, revised printing, 1995.
[12] Generalized Perspective-Based Inspection to handle Object-Oriented Development Artifacts. Proc. ICSE'99, Aug. 1999, IEEE CS-Press, pp. 494-503.

ISBN: 978-988-19252-4-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)