



Whitemarsh
Information Systems Corporation

Strategy for
Successful Development of
Business Information Systems

Michael M. Gorman

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where Whitemarsh Press is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

©2007 by Whitemarsh Information Systems Corporation
All rights reserved.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If legal advice or other expert assistance is required, the services of a competent professional person should be sought. FROM A DECLARATION OF PRINCIPLES JOINTLY ADOPTED BY A COMMITTEE OF THE AMERICAN BAR ASSOCIATION AND A COMMITTEE OF PUBLISHERS.

Reproduction or translation of any part of this work beyond that permitted by section 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright holder is unlawful. Requests for permission or further information should be addressed to Whitemarsh Press.

ISBN 978-0-9789968-1-9

Printed in the United States of America

Table of Contents

Preface	xiii
1. The Problem	1
1.1 Organizational Environment for the Nine Step Approach	5
1.2 Problem Summary	7
1.3 Remainder of this Book	8
1.4 Questions and Exercises	11
2. Essentials for Business Information System Development Success	13
2.1 The Knowledge Worker Framework	16
2.2 Data-Driven Methodology	29
2.3 Database Object Classes	34
2.4 Data Architectures	40
2.4.1 Database Architecture Classes	41
2.4.2 Data Model Generalization Levels	43
2.5 Business Information System Generators	47
2.6 Metabase Environment	54
2.7 Discrete and Release Development Environments	66
2.8 Metrics and Work Environment Multipliers	70
2.9 Summary	72
2.10 Questions and Exercises	74
3. Nine Step Approach to Business Information Systems Development ...	77
3.1 Step 1: Develop Missions	78
3.2 Step 2: Design Database	85
3.3 Step 3: Generate Prototype	103
Step 3.1: Create Clarion Dictionary	108
Step 3.2: Generate Initial Business Information System	110
3.4 Step 4: Evolve the Specification Through Prototyping	131
Step 4.1: Demonstrate Prototype	131
Step 4.2: Evolve Prototype	133
Step 4.3: Achieve Benefits From Prototyping	149
3.5 Step 5: Create Request for Proposal	151
3.6 Step 6: Evaluate Proposal Responses	154
3.7 Step 7: Award Contract	154
3.8 Step 8: Manage Contractor	155
3.9 Step 9: Test Conformance to Prototype	155
3.10 Nine-Step Approach Summary and Conclusion	157

3.11 Questions and Exercises	160
4. Business Information Systems Planning	165
4.1 Rationale for a Business Information Systems Plan	165
4.2 Characteristics of a Quality Business Information Systems Plan ...	168
4.3 Business Information Systems Plan Within the Context of the Metadata Environment	169
4.4 The Business Information Systems Plan Steps	170
4.4.1 Step 1: Create Mission Model	173
4.4.2 Step 2: Build the High Level Data Model	173
4.4.3 Step 3: Create Resources and the Resource Life Cycles	173
Step 3.1: Determine the Resources	175
Step 3.2: Determine the Resource Life Cycles	176
4.4.4 Step 4: Allocate Precedence Vectors among Resource Life Cycle Nodes	177
4.4.5. Step 5: Allocate Business Information Systems and Databases to the Resource Life Cycle Nodes	181
Step 5.1: Allocate Existing (As-is) Databases or Files to Resource Life Cycle Nodes	182
Step 5.2: Allocate Existing (As-Is) Business Information System to Resource Life Cycle Node	183
Step 5.3: Allocate Future (To-Be) Databases to Resource Life Cycle Node	184
Step 5.4: Allocate Future (To-Be) Business Information System to Resource Life Cycle Node	185
Step 5.5: Configure Business Information Systems Plan Projects	186
4.4.6 Step 6: Allocate Standard Work Break down Structures to Each Business Information Systems and Database Project	189
4.4.7 Step 7: Load Resources into Each Project	192
4.4.8 Step 8: Schedule through a Project Management Package ...	192
4.4.9 Step 9: Produce and Review the Business Information Systems Plan	194
4.4.10 Step 10: Execute and Adjust the Business Information Systems Plan Through Time	197
4.5 Business Information System Plan Summary	198
4.6 Questions and Exercises	199

Table of Contents

5. Project Management	201
5.1 Why Project Management is Important	201
5.2 Whitemarsh Project Management Environment	204
5.3 Whitemarsh Project Management, a Difference in Kind	205
5.4 Architecture and Concept of Operations	208
5.5 Whitemarsh Project Management Summary	218
5.6 Questions and Exercises	219
6. Summary and Conclusions	221
6.1 The Preface	221
6.2 The Problem	222
6.3 Essentials for Information Technology Success	223
6.4 Nine-Step Approach	224
6.5 Business Information Systems Plan	227
6.6 Project Management	227
6.7 Way Ahead: BLUF	228
6.8 Questions and Exercises	229
Attachment 1. Knowledge Worker Framework. Rows and Columns	
Description	231
A1.1 Knowledge Worker Framework Rows	232
A1.1.1 Scope Row	234
A1.1.2 Business Row	235
A1.1.3 System Row	236
A1.1.4 Technology Row	238
A1.1.5 Deployment Row	239
A1.1.6 Operations Row	240
A1.2 Knowledge Worker Framework Columns	240
A1.2.1 Mission Column	240
A1.2.2 Database Object Class Column	242
A1.2.3 Business Information System Column	244
A1.2.4 Business Event Column	247
A1.2.5 Business Function Column	249
A1.2.6 Organization Column	252
Attachment 2. Database Architecture Class Characteristics	255
Index	261

Figures

Figure 1. Database architecture classes.	42
Figure 2. Phases for the traditional information systems development life cycle.	48
Figure 3. Iterative systems development life cycle.	49
Figure 4. Cost structure of Systems Development Life Cycle	52
Figure 5. Interrelationship between good documentation and metadata repository meta entities.	55
Figure 6. Interrelationship between Metabase meta models and Knowledge Worker Framework columns.	56
Figure 7. Whitemarsh metabase domain.	58
Figure 8. Opening window of the Whitemarsh Metabase system.	64
Figure 9. Selecting metabase database instance screen.	65
Figure 10. Continuous flow environment.	68
Figure 11. Human resources mission hierarchy.	79
Figure 12. Metabase support for mission, organizations, functions, and positions.	82
Figure 13. Metabase mission hierarchy screen.	82
Figure 14. Metabase mission organization assignment.	84
Figure 15. Movie rental outlets database domain.	88
Figure 16. Entity-relationship diagram model for movie rental outlets.	89
Figure 17. Meta model for the specified data model.	91
Figure 18. Meta-model for the data elements through to DBMS columns. ...	93
Figure 19. Tagging window for creating attributes.	94
Figure 20. Changing an attribute.	96
Figure 21. Update screen for changing an attribute.	97
Figure 22. Adding semantics to an attribute.	98
Figure 23. Metadata models for data modeling.	99
Figure 24. DBMS schema, DBMS tables, DBMS columns, and relationships.	104
Figure 25. Movies data warehouse table relationship diagram	105
Figure 26. Reuse of metadata across data element through operational data models.	107
Figure 27. Object oriented application generation paradigm.	107
Figure 28. Movies data warehouse of tables, columns and relationships. .	109
Figure 29. Creating a new application generation.	112
Figure 30. Start process for detailing options for application generation. ...	113

Figure 31. Application tree generated from database table structure.	114
Figure 32. Menu of browses of auto-generated movies data warehouse.	115
Figure 33. Menu of reports of auto-generated movies data warehouse.	116
Figure 34. Generated movie rental record browse and update screen.	117
Figure 35. Trimmed application tree.	119
Figure 36. Revised movie rental record browse and update screen.	120
Figure 37. Movie rental records browse with column header sorting.	121
Figure 38. Movie Rental Record update screen with referential integrity select.	122
Figure 39. Customer select screen with referential action.	124
Figure 40. Establishing the reports library.	126
Figure 41. Synchronization between dictionary and report library.	127
Figure 42. Sort and break fields for a specific report.	128
Figure 43. Ready to run report.	129
Figure 44. Example of an executed report.	130
Figure 45. Modified movie sales data model.	134
Figure 46. Screen for importing a single entity from the specified data model into the implemented data model.	135
Figure 47. Foreign key creation screen.	136
Figure 48. Schema import screen from Implemented to Operational Data Models.	137
Figure 49. Movies application menu modification process.	138
Figure 50. Modified application tree for the new distributor module.	139
Figure 51. Selecting the browse wizard generation template.	141
Figure 52. Starting the distributor code-generation process.	142
Figure 53. Ready to modify distributor generated window.	143
Figure 54. Newly generated, unmodified distributor browse.	144
Figure 55. Initially generated movies browse.	145
Figure 56. Modified movies browse.	146
Figure 57. Initial Movies update screen.	147
Figure 58. Entry of referential actions specifications on movies update screen.	148
Figure 59. Critical components of the systems development life cycle.	150
Figure 60. High-level view of metabase meta entities involved in information systems planning.	169
Figure 61. Case.	177
Figure 62. Courts Personnel.	177
Figure 63. Document.	178

Table of Contents

Figure 64. Precedence vectors among resource life cycles. 179

Figure 65. Alternatives and assessment areas necessary to determine preferred project for database or file transformations. 187

Figure 66. Alternatives and assessment areas necessary to determine preferred projects for business information systems. 188

Figure 67. Projects allocated to resource life cycle nodes with deliverables and project templates. 189

Figure 68. Resource life cycle network. 193

Figure 69. RLC network seen as a PERT chart. 194

Figure 70. Overall database architecture for Whitemarsh project management 209

Figure 71. Project template list of the Whitemarsh project management system. 210

Figure 72. Deliverable template for a specific deliverable template type. . . 211

Figure 73. Information captured about deliverable templates. 211

Figure 74. Task template list. 212

Figure 75. Information captured about task templates. 214

Figure 76. Project list with associated tasks and deliverables. 215

Figure 77. Project update screen. 216

Figure 78. Task update screen. 217

Strategy for Successful Development of Business Information Systems

Tables

Table 1. Essentials for Business information system Success.	15
Table 2. Knowledge worker framework.	19
Table 3. Knowledge worker framework: 24 information technology environment cells.	23
Table 4. Knowledge worker framework: 12 information technology cells. . .	24
Table 5. Allocation of GAO information technology failure causes to the cells of the knowledge worker framework.	25
Table 6. Percent Allocation of GAO information technology failure causes to the cells of the Knowledge Worker Framework.	26
Table 7. Summary of U.S. General Accountability Office Reasons for Failure	27
Table 8. Critical Counts of artifacts from process-first versus data-first approaches.	30
Table 9. Data model generalization levels.	45
Table 10. Benefits from highly organized, standardized data.	45
Table 11. Costs involved in top-down data administration activities	46
Table 12. Metadata domains set against knowledge worker areas.	61
Table 13. Distribution of metadata across the knowledge worker framework.	62
Table 14. Example of a set of work environment factors for a specific project task.	72
Table 15. Critical difference between missions and functions.	80
Table 16. Names and brief descriptions of the data model classes supported by the Metabase.	100
Table 17. Benefits from the nine step approach to business information systems development.	160
Table 18. Characteristics of a quality Business Information Systems plan.	168
Table 19. Business Information Systems Plan development steps.	172
Table 20. Resources and Resource Life Cycles.	175
Table 21. Resource characteristics.	176
Table 22. Resource life cycle precedence vector enablement stories.	181
Table 23. Benefits from the nine step approach to business information systems development.	227
Table 24. Cross reference between Resource Life Cycle Nodes and Business Information Systems.	246

Table 25. Resource life cycle cross relationship with business information systems and business events. 248
Table 26. Data architecture class descriptions. 260

Preface

A government agency needed a project management system: Not Microsoft Project, but a business information system¹ to manage their functional projects. The overall scope included projects, staff, deliverables, assignments, status reports, travel and expenses, client organizations and their staff, and the like. So the agency hired a contractor to come in and do a requirements analysis, and build a business information system. The requirements analysis was standard fare: meet with management, meet with functional experts, and meet with technical staff. Everybody was interviewed and the requirements document filled several volumes. Everyone was happy.

The contractor took the requirements document away and implemented the business information system. After about a year of detailed design, coding, unit testing, and system testing, the business information system was delivered to the government agency. The government agency shrieked in horror. What had the contractor done? That is not what the agency had said. Not what they had wanted. How could this happen? So the contractor was promptly fired.

Since this clearly had to be an aberration, a new contractor was hired. The new contractor met with management, met with functional experts, and met with technical staff. Everybody was interviewed and the requirements document filled several volumes. Everyone was happy.

The contractor took the requirements document away and implemented the business information system. After about a year of detailed design, coding, unit testing, and system testing, the business information system was delivered to the government agency. The government agency shrieked in horror. What had the contractor done? That is not what the agency had said. Not what they had wanted. How could this happen? So the contractor was promptly fired.

Since this clearly had to be an aberration, a new contractor was hired. Now to save the readers time, just go to back to the start of this cycle and read it again: twice more.

¹. In this book, a business information system is a set of application-specific software that created, manipulates, evolves or deletes data—most commonly—from a database through a database management system in support of some mission area of the enterprise. A business information system is class of information system, and it is distinguished from other classes such as a computer's operating system, or "systems software" information system such as telecommunications management, database management systems, and end-user security management.

An epiphany then happened to the government agency. Since all the contractors were using the same data processing and implementation technology infrastructure to build the business information system, it had to be the fault of the underlying technology infrastructure. The next step was to try to terminate and/or replace the underlying technology infrastructure. At that point, the technology vendor brought in a consultant to completely evaluate the situation. The cycles of requirements through failure were examined to determine what went wrong. A classic methodology was used, and so it was presumed that there would be a classic result. Actually, there was.

After the study, a meeting was called by the consultant with the government agency's heads. The agency was eager to know not only what went wrong but who to blame. The answer was a shock. It was the agency that was at fault, not the contractors. Of course the agency angrily protested the findings. The consultant carefully went through the requirements-failure cycles and showed that during every cycle the perception of what the problem was and therefore what the solution should be had changed.

It was not a case of whether the contractor got the requirements right. Rather it was a case of the government agency not knowing what requirements it actually wanted. When the contractor talked to different government staff, the answers were different not just from one staff member to another, but from previous answers provided by the same government staff member over the different cycles. The objective, a successfully developed business information system, was impossible to achieve.

What to do? It was suggested to the government agency that they create their own detailed specification of what needed to be accomplished. The agency countered, "that's the contractor's responsibility!" The consultant indicated that if four different contractors couldn't create the "just right" specification, it was not the contractor's fault. The agency asked what it should do. The consultant suggested that a week-long workshop be conducted with agency staff to derive the requirements. The agency agreed. The consultant asked the agency head for the names of the staff that absolutely were too important to be on the team. A list was immediately produced. The consultant said, "these are the names of the individuals that must participate." The agency head's response was not "printable."

The final agreement was this. A workshop would be conducted that would cause the creation not only of the specifications of the business information system to be produced, but also a prototype as the specification's

proof. Further, the specifications were to be stored in an early version of the Whitemarsh Metabase² so that the repository could be used in the subsequent full implementation contract as the key reference point for the specification of what needed to be implemented. Finally, the process was required to iterate the requirements via the prototype demonstrations until complete.

The workshop was conducted over five full days. There were four agency teams of four individuals each along with one information technology person per team. Each team lead had to be a functional expert. Additionally, the team lead was instructed that the information technology person was to act solely as the team's scribe in creating the metadata. If the information technology person got out of hand, it was suggested to the team lead that they could use duct-tape to silence the information technology person. A stick was also provided.

At the end of the week, the specification, the metadata, and a high-level prototype were complete. The agency's information technology department took the result and increased it with one or two more levels of detail, including evolving the prototype. The full implementation contract was let and the system was developed, tested, and accepted by the agency.

This book is all about how to create this kind of environment. That is, functional experts employing a CASE/Repository tool to create valid requirements, the use of business information system generators by these same individuals to create a prototype of the requirements, and the iteration of the prototype until all the requirements have been teased out of the functional experts and have been properly reflected in the overall specification that is demonstrated via the prototype.

The CASE/Repository tool employed to illustrate the overall process in this book is the Whitemarsh Metabase. Hereafter, the terms CASE and metadata repository will be referred to by the single term, Metabase, because

² Metadata is a generic term that identifies all classes of information technology specifications across the enterprise. Hence all data and process specifications are metadata. All requirements could also be considered metadata. A metadata repository is a database within which all metadata is stored. A metadata database is a metabase. Whitemarsh has employed the term, Metabase, in this context since 1982. A metadata repository system is a software system that captures, stores, reports, and manages all metadata. The system that manages the Metabase is the Metabase system. Sophisticated metadata repositories are multi-user and support the capture and reporting of metadata in a non-redundant, integrated manner across the enterprise.

it was designed to be both a CASE tool for database and systems engineering, and a metadata repository for all in the enterprise to employ. Readers can go to the Whitemarsh website (www.wiscorp.com) and download the Metabase. Once installed all the computer windows in this book can be seen via the Metabase system.

Whitemarsh started using the term, Metabase, in 1982 as a way to signify a metadata database³. An early production-class version of the Whitemarsh Metabase was implemented by Hartford Insurance in 1979 with great success and with an enterprise⁴ scope. Another Metabase was implemented by Whitemarsh for the U.S. Army in 1984. The Army Metabase enabled the manufacturing of both specifications and software to such an extent that the per-system cost was reduced from about \$400K to less than \$40K.

Today, the term, metabase, has become very popular. A recent "Google search" identified almost one million hits. However, whenever this term, Metabase, is used in this book it refers to the Whitemarsh Metabase. Metabase has been used on almost every Whitemarsh project. The result has always been the same: More for less in a shorter time, at higher quality and lower risk. Over the years, the Metabase has been implemented through different database management systems such as CSC's Manage, Computer Associate's IDMS, Cincom's Supra, Information Busilder's Focus, and SoftVelocity's Clarion. The Clarion version is both the most recent and the

³ In this book and in all Whitemarsh materials, database and DBMS are very different terms. A database is meant to imply an organized collection of business data that conforms to rigorous semantics and high levels of standardization and integrity. Databases come in different classes such as original data capture, data warehouses, and subject areas. DBMS, meaning database management system is a software system created by vendors to define and manage databases. Oracle is a DBMS vendor and its DBMS is called Oracle. Oracle is thus not a database vendor because it does not create, sell, and maintain databases. Rather, it creates, sells and maintains DBMSs. Confusing database with DBMS is like confusing passengers with vehicle.

⁴ As defined within this book, an enterprise is merely a term to relate to a collection of organization units that have common collections of data, processes, activities within a business or a company and sometimes beyond corporate affiliations as in the case of data interchanges. An enterprise is therefore not just a synonym for business or a company. Rather, it is intended to convey a common data, process, and activity view across the organizational units sharing that view.

most sophisticated. Finally, a CSC Manage version of the Metabase was also used in the Workshop described in this Preface.

Additionally, all the software examples in this document were implemented through Clarion (www.SoftVelocity.com). Clarion was chosen because it, based on Whitemarsh's experience, alone fulfills the minimum essential requirements for the high quality, cost effective business information system generator that is essential for prototyping and iterating business information systems requirements specifications. The metadata et al for the case study in this book is available through the Metabase's demo database which can be obtained from the Whitemarsh website. The example data is from the Movies example Metabase database instance. The Movie Rental Corporation is a "nom de plume" for the largest movie rental business in the United States.

Clarion is also well suited for production class database applications on Microsoft operating system environments. All of Whitemarsh's production class software products are implemented through Clarion. The underlying databases currently operate either through Clarion access methods or through ODBC to standard SQL server-based engines.

Recently, Whitemarsh had the opportunity to create a large scale membership management business information system for an international association. Upon completion, the database and business information system was sized to be 6600 function points. By industry standards, a function point costs \$400. Thus, under traditional methods, the database and application should have cost \$2.64 million and taken more than 12 staff years. But through the use of this book's principles, Metabase, and Clarion, the business information system cost the Whitemarsh client only about \$350 thousand, and took less than two staff years. That's an 87% reduction in cost and more than six times improvement in staff productivity. The implemented application has been running now for about six years. No changes have been needed.

Notwithstanding this ringing endorsement of Clarion as a client-side development environment, Whitemarsh has engineered its software and the underlying databases such that they can be implemented through other client-side development environments. The key "take-away" here is not that Clarion should be purchased and employed (although that's a highly recommended idea), it's that the process of iterative design-prototype development is the right strategy to employ so as to:

- Avoid wasting the time and resources of information technology.

- Accelerate the creation, evolution, and deployment of information technology solutions
- Maximize the precious time of functional experts

If these are key goals in your organization, this book is for you and your organization. Act on these goals. Increasing production, lowering cost, increasing quality, and reducing risk will be your result.

This book is not however just about one-off business information systems development. It also addresses the need for business information systems planning, at a high level, across the enterprise. It does not serve the enterprise well if competing organizations implement conflicting solutions characterized by different definitions for the same data, different processes for the same functions, and redundant business information systems. Such redundant efforts are a profound waste of time and resources. Not only is the time to create the redundant and conflicting solution a waste, so also is the effort to intersect these ill-conceived solutions, to ultimately dismantle them, and to then replace them. Added to this waste is the opportunity lost to accomplish the “real” missions of the enterprise.

This book addresses the need for having business information system projects (regardless of sponsor or author) integrated within one project management environment so that maximum efficiency can be achieved.

There needs to be an integrated approach to data within the enterprise so that some organizations can use the data created by others. This is important. There is more than enough data within enterprises. What is scarce are integration, non-redundancy, and semantic harmonization.

A significant quantity of today’s key infrastructure data is already captured and managed by “ERP” (Enterprise Resource Package) systems. Consequently, most new business information systems are ancillary or supplementary to key functionality that is already automated.

Currently, most organizations have a mixed technology environment. That is, some legacy batch processing, some client-server, and other environments that are Internet based. The reason why this book and the approach illustrated herein is independent of implementation strategy is because this book is not about implementation. Rather, it is about the creation of an implementable set of business information system specifications that are inherently valid. Then comes implementation.

Preface

This book has been years in the making because it has evolved through many cycles of actual use within projects starting back in the late 1970s. During this time, the Metabase has been implemented a number of times under a myriad of DBMSs for clients in many different industry and government sectors. This implementation of the Metabase is within Clarion (www.SoftVelocity.com), which is by far the most sophisticated business information system generator that exists today.

Along the way, there have been key contributors to this effort. Thanks therefore goes to Herman Koester of St. Louis who challenged me to put the entire approach into one of those infamous “12-step” plans. Sorry, there are only 9-steps. Thanks goes to Hank Lavender, who, for almost 10 years, has been a constant and relentless reviewer and inquisitor. Gratitude also goes to Bruce Haberkamp who read a mid 1990s manuscript of this book and encouraged me to incorporate its key points into the process that is now part of Army policies, AR 25-1 and the DA PAM 25-1-1. Thanks also goes to Peter Rush who, despite work, home, and personal pressures found the time to make very cogent and insightful comments. Finally, thanks goes to my wife Maxine and my son, Matt who listened attentively, and asked critical questions that sent me back into my rewrite cave.

1

The Problem

Many, if not most, information technology projects exhibit these characteristics: over budget, under specified, delivered late, and unable to meet organizational expectations^{5 6 7}. In one of the studies by The Standish Corporation, 31% of a class of client/server systems efforts failed outright; another 53% were challenged (late, greater than budgeted, and fewer features than promised), and only 16% were delivered on-time, within budget, and with features as promised.

In the Chaos study by Standish, the top three reasons uncovered for successful business information systems were:

- User involvement.
- Executive management support.
- Clear statement of requirements.

And, the three top reasons cited by executives for business information systems failures were:

- Incomplete requirements.
- Lack of user involvement.
- Lack of resources.

⁵ The Standish Group. *CHAOS: 1998: A Summary Review: 1999*. The Standish Group International, Dennis, MA 02638.

⁶ Matson, Eric. Speed Kills (the Competition). *Fast Company* (August 1996). Page 1. Web: www.fastcompany.com/online/04/speed.html.

⁷ Strassmann, Paul A. *40 Years of IT History: 1997*. Page 6. Web: www.strassmann.com/pubs/datamation1097/index.html

All the Standish studies have been updated almost every year. While there has been some improvement, there has not been a dramatic change. The failure or challenge percentages are relatively the same and the reasons for success and failure remain almost constant.

The United States Government's General Accountability Office (GAO) has been studying information technology projects for a number of years. A review of United States General Accountability Office (www.gao.gov) studies of why business information systems fail shows that new requirements so commonly crop up during the business information systems development phase that this must be considered intrinsic to the software development life cycle as currently practiced. This phenomenon is the root cause of a preponderance of these GAO identified reasons for failure.

The new problems that arise when new requirements are uncovered during development fall into two categories: Database design changes and software changes.

Significant database design changes often result from an insufficiently data-driven methodology through which the database is initially designed. Experience has shown that very high quality database designs created through a data-driven methodology commonly return many times their design cost in reduced software development and evolution costs.

Software changes result from database design changes and also from process logic changes. Database design changes can be largely eliminated through the use of a quality database design methodology. The onerous effects of process logic changes, can be dramatically affected through the use of object-oriented analysis, design and programming techniques employed within the environment of business information system generators.

A key strategy to minimize the negative impact of software changes is "code-generation," that is, through a business information system generator. It is a software system that takes in metadata specifications that have been created through the Metabase or other tools such as data modeling tools, and outputs the actual production-ready business information system. Business information system generators have evolved greatly in the last 20 years and should be used in almost all situations. Clarion is a business information system generator.

A review of eight GAO studies mentioned above clearly shows that the main reasons why business information systems fail has nothing to do

The Problem

with information technology^{8 9}. Rather, the failures reside outside the sphere of information technology control. Nevertheless, information technology must work with the enterprise in the creation of a knowledge worker environment that enables information technology to be successful. Such an environment would be Win-Win all the way around. In support of that goal, this book presents the following:

- A description of the essential prerequisites for information technology success.
- The nine-step approach to business information systems development.
- A strategy for developing, executing and maintaining enterprise-wide business information systems and to identify the right accomplishment sequence for these business information systems.
- An overview of enterprise-wide project management that enables the nine-step projects to be completed on-time, within budget, and delivering what is promised.

Before presenting this 9-step approach in any detail, a fundamental objection to this entire approach must be addressed. It has become painfully clear that some organizations try to avoid the problem of underdeveloped or immature requirements by buying COTS. COTS, that is Commercial Off the Shelf Software, if improperly procured, may exacerbate this problem, not solve it.

When COTS is purchased, what is actually being procured is software based on somebody else's requirements analysis. Was that analysis sufficient? Was it comprehensive? Did it match the organization's real needs? If any of the answers is no, then buying COTS could produce a bad result for four reasons.

⁸ Gorman, Michael M. Knowledge Worker Framework: 1999. Web: www.wiscorp.com

⁹ United States Government Accounting Office. Managing Technology: Best Practices Can Improve Performance and Produce Results.: 1997 (GAO/T-AIMD-97-38). Washington, D.C. (web: www.gao.gov)

- Any business information system built on top of inadequate requirements causes an unsuitable COTS system to be selected and installed. The software will have been purchased. Staff will have been trained. Hardware will have been procured. Data will have been converted. Only after production use has begun will it be known that it was the wrong solution. Fixing that problem requires either abandoning the purchase, training, hardware, and data conversion, and beginning the process all over again, or convincing the users that their real requirements, which are only starting to be discovered through system use, aren't all that important or necessary. The first alternative is very expensive, and the second is quite unacceptable.
- Changing COTS installed software ranges from difficult and expensive, to impossible. Once changed, some COTS becomes custom software, which subsequently, is both difficult and expensive to change to the next COTS version.
- How would you know if the COTS system is based on inadequate requirements if a thorough requirements analysis hasn't been performed? Without doing this book's 9-step approach, the probability of having the right requirements will be very low. The risk will be very high.
- Because the acquired software is COTS, the procuring organization now lacks both the capabilities and the tools to make its own software modifications. Rather, the organization's mission is accomplishable only when and if some outside vendor changes the COTS system. If the initial requirements analysis either wasn't done or was done in a cursory manner, the organization's mission accomplishment may be fatally impacted.

If an organization wishes to purchase COTS, it is because of the four reasons cited above that the 9-step approach, albeit modified, is even more essential. During Step 8, instead of actually "building" the software, the organization takes the five critical elements that result from this approach, that is, Missions, Organizations, Functions, Database Design, and validated Prototype and wraps all five into a Request for Proposal and issues it to a potential set of vendors. What will be given to the potential vendors is a highly refined and

validated set of requirements, and what will come back from the vendors will be a COTS proposal and software system that matches the “real” requirements.

1.1 Organizational Environment for the Nine Step Approach

This book presents a nine-step process that addresses many of the Standish and GAO information technology findings. The nine steps are:

- Develop missions.
- Design the database.
- Generate the prototype.
- Evolve the specification through prototyping.
- Create the request for proposal.
- Evaluate the vendor responses.
- Award the contract.
- Manage the contractor.
- Test conformance to prototype.

This book presumes that many requirements development organizations¹⁰ do not contain sophisticated information technology development organizations and have no interest in establishing them. Rather, this book presumes that requirements development organizations prefer to specify information technology needs through some sort of central design authority, and, through the business information systems development organizations, procure, install,

¹⁰ In the context of this paper, a requirements development organization is one that possesses significant functional knowledge of a business area that is to undergo automation or an automation upgrade. In contrast, an information systems development organization is one that possesses the skill necessary to transform a requirements document into a highly efficient computer system.

While the knowledge areas of these two groups may overlap, they are largely non-intersecting. Finally, both organizations may be in the same or different enterprises. They deal with each other through formal memorandums of understanding, or even legally binding contracts. In the former case, they would be two different agencies within some government organization, large industry, or university setting, and in the later case they would be separate corporate entities.

employ information technology solutions that are both functionally acceptable and conformance tested to ensure common functionality across requirements development organizations.¹¹

Whenever requirements development organizations take on full development including complete business information systems development, operation, evolution and maintenance, that is, all nine steps, they commonly fail to achieve optimum results. While there can be many reasons for this, the GAO studies show the most common to be:

- Failure to meet initial end-user expectations.
- Inability to continuously infuse advances in technology in the deployed information technology environments.
- Inability to break from the long-standing tradition of individual autonomy, that is, stove-pipe development.

This last reason, autonomy, prevents the effective deployment of business information systems based on a unified business information system's design and implementation strategy across a group of requirements development organizations because:

- One size does not fit all.
- Requirements development organizations often require slightly different functionality one from the other.
- The cost of evolving business information systems through old technologies and approaches is neither cost effective nor viable.

A solution that does work is one that capitalizes on the strengths of both the requirements development organizations and business information systems development organizations, while avoiding their weaknesses. The proposed approach consists of a three-part paradigm:

¹¹ Strassmann, Paul A. *Outsourcing IT: Miracle cure or emetic*: 1998. Web: www.strassmann.com/pubs/outsourcing.shtml

The Problem

- A requirements development organization to define, validate through prototyping, and maintain functional requirements.
- A business information systems development organization that creates production class business information systems from the prototype.
- Conformance tests and testing by the requirements development organization that ensures that the developed business information system conforms to the essential functionality contained in the prototype and the Metabase.

Even though this nine-step approach is optimum for large, heterogeneous hardware environments across multi-site requirements development organizations, this approach can be simplified to accommodate homogenous and/or single-site requirements development organization environments.

While all nine steps are the responsibility of requirements development organizations, the bulk of the actual work to actually implement the business information system, that is, Step 8 would be accomplished by a business information system organization, with oversight, of course, from the requirements development organization.

All of these “Whitemarsh” tool-based examples are fundamentally based on just good information technology common sense. Hence every tool-presented window should be able to be accomplished by any number of tool suites. The reason why the Whitemarsh tool suite was employed was to eliminate any objection to the use of the book’s strategies. That is, that the book is just a collection of theories. Clearly these are not just theories. Rather, this book represents the best practice of tens of thousands of users around the world.

1.2 Problem Summary

Inadequate responses to the problem addressed by this book, that is, how to successfully create “requirements” in advance of business information system development are endemic. Information technology’s traditional solution to this problem has however been quite unique. It has been to demand something that is not only impossible but is never done in almost any other discipline.

Information technology's demand is that the requirements document be complete and comprehensive, unchanging, and near absolutely correct before any prototyping, coding, or other development activities are begun. If all the requirements are either not present or imperfect, there will be cost overruns, and excessive time spent correcting the "mistakes" of the functional users.

In virtually every other complex discipline, models and prototypes are developed and iterated a number of times so as to "tease out" the true nature, design, and desired behavior. Over the years, the need for prototypes has been trumpeted loud and clear. Everyone gets excited, but then nothing seems to happen. That's because there's been no information technology methodology specifically engineered to create the prototypes; no Metabase environment to capture, store, and iterate the prototypes; and no integrated code-generation environment that can quickly create the prototyped business information systems in a form that is sufficiently real to enable the real requirements to surface. Finally, no strategy exists to iterate a given requirement from initial inclusion within the Metabase through prototype environment and recycle.

This book presents a methodology for prototyping. Further, it points to the Metabase tool that can be downloaded and employed for a sufficient amount of time to fully demonstrate the prototyping solution. Finally, this book points to an integrated code-generation environment, Clarion, that fully supports prototyping.

1.3 Remainder of this Book

Chapter 2 of this book presents a description of the essential prerequisites for business information system development success, and states why having this environment returns many times its cost. It shows that the use of the tools described in this book can reduce business information systems development costs by about 66%. In this chapter, the discussion of the Knowledge Worker Framework identifies the different major classes of activities and products that have to be created in support of enterprise database, and business information systems development.

This chapter also shows the allocation of the U.S. General Accountability Office's "Reasons for failures" that occur when the activities and products identified in the Knowledge Worker Framework are not

accomplished. The most surprising part of the allocation of failure is that 95% of all business information systems fail for reasons outside the domain of information technology.

Addressed in this chapter are the steps of a methodology specially designed to advance the quality of the very requirements that must be present for successful databases and business information systems. The chapter identifies the characteristics of business information system generators that need to be present to accomplish prototyping, the key characteristics of the Metabase, and CASE tools.

This chapter addresses the need to reorient from a stove-pipe project-based mentality to a "release" mentality to enable capabilities across multiple business information systems to advance in a coordinated fashion.

Finally, the chapter identifies key metrics and work environment factors that dramatically affect both the quality and the timeliness of database and business information system accomplishment.

Chapter 3 presents the nine-step approach through screen-shots from the Metabase and the Clarion environment that illustrate how to accomplish each of the key steps. The key surprise of this 9-step approach is that the first seven steps require only business users and functional experts, and that only the last two steps require the involvement of an information technology organization. Step 8 is the step within which the business information systems development organization creates the business information system. Step 9 is the conformance testing step wherein the requirements development organization certifies that the created business information system meets the requirements.

The main objective of Steps 1-7 is to enable functional users to advance their awareness of their information technology requirement and to evolve this awareness through prototyping until it is then ready for information technology to implement: one time and correctly.

Chapter 4 presents an overview of the Whitemarsh Business Information Systems Plan process. It does no good to know how to implement business information systems well if you do not know which ones to implement and in which sequence.

Over the past 10 years, starting in the late 1990s, there has been a dramatic increase in the use of Enterprise Resource Packages (ERP). These are comprehensive database and business information system implementations across broad functional areas. Consequently, many of the newly created database and business information systems are derivative business

information systems, not one-off business information systems that start from blank slates. Because of this very significant change, enterprise-wide database and business information systems plans are more important than ever, and the very metadata created during the development of these plans needs to be stored in the Metabase so that impact analyses can be quickly developed and accomplished.

As a direct consequence of ERPs, there have been fewer large scale business information system implementations, and many more moderate to small business information system efforts. This increases the need for enterprise-wide information systems plans to integrate and interrelate the data from the more numerous decentralized and distributed database and business information system development. Many of these are building discordant stove-pipes of semantics, data models, process models, and the like. An enterprise-wide information systems plan can help identify and manage efforts to deconflict and eliminate redundant databases and business information systems.

Chapter 5, Whitemarsh Project Management presents an overview of the Whitemarsh approach to project management. This too is illustrated through screen-shots from the Whitemarsh project management system.

The Whitemarsh project management approach is different from traditional time-management approaches because it manages deliverables rather than expended time through work plans, and it enables enterprise-wide project management through the use of project, deliverable, and task templates coupled with person-based skill inventories and work environment factors. Thus, while every project is different, each is built from commonly found (define once, use many times) building blocks. The entire Whitemarsh environment meets one of the Software Engineering Institute's critical success factors: Self-correction. The Whitemarsh approach to project management is especially important because it is set within the context of other enterprise metadata and all the projects that are identified, in development, in production, or in maintenance across the enterprise.

Chapter 6, Summary, brings forth the features, advantages, benefits, conclusions and future actions that flow from adopting this approach to database and business information systems development. There is no downside to adoption. The overall information technology organization, and the functional organizations that are supported by information technology, are more productive, less costly, of higher quality, and lower risk because more work is done in a non-redundant, integrated manner. More work

products are able to be reused. Data semantics can be harmonized, which then eliminates whole classes of data transformation and reloading business information systems and logic. Again, there is no downside to doing more, faster at a lower cost and risk.

1.4 Questions and Exercises

1. Have you been involved in situations like the one in the Preface? What was the most common set of reactions? Was the root causes similar to those cited in the Preface? If yes, how did you resolve the situation. If no, what were the root causes and what did you do?
2. Have you used a CASE tool? What were its good and bad features? If it's no longer used, why did it fail? Did it have an explicit ODBC accessible database? Was or would that have been good?
3. Have you used a data and/or process modeling tools? What were its good and bad features? If it's no longer used, why did it fail? Did it have a explicit ODBC accessible database? Was or would that have been good? Were you able to integrate the data and process metadata? If not, what did you do? Was data a slave to process or vice versa? What's good or bad about that? Did these tools have an explicit ODBC accessible database? Was or would that have been good?
4. Have you used a Metadata Repository such as Rochade? What were its good and bad features? If it's no longer used, why did it fail? Did it have an explicit ODBC accessible database? Was or would that have been good?
5. Would a metadata management tool that combines, CASE, data and process engineering, and a metadata repository be of value? What should its features be? Should it have an open database schema architecture such that it could be accessed through ODBC via commercial report writers and 3GL programming languages?

6. Do you agree or disagree with the Standish reasons for business information system success or failure? Explain. What other reasons come to mind? Which ones are “app killers?”
7. Do you fundamentally agree or disagree with the 9-steps as a way to engineer and deploy business information system correctly and for the first time? Explain. What steps are missing and what’s the effect of their being missing?
8. Given that you have a practical way to do prototypes, how key are they in the overall business information system development success especially as a way to evolve and validate requirements?
9. Can you successfully avoid the “requirements problem” by just buying a commercial off the shelf (COTS) package?
10. How should the 9-step approach be modified to work with package purchasing? What steps are not really needed or should be modified?
11. If the 9-step approach is followed and a COTS procurement produces no-bidders, why might that be a good result? What’s been the cost? What’s possibly been saved? What then are your options?
12. What’s your definition of a stove-pipe? How can COTS be or not be a stove-pipe? How can stove-pipe development be avoided? How would a Metabase tool help avoid stove-pipes?
13. Are requirements changes a sign of failure or a sign of unfolding reality? Is it possible to discover all requirements up front? How will prototyping help discover new requirements?
14. How has Enterprise Resource Planning (ERP) packages affected the ability to have enterprise-wide data and process semantics and integration? Given that you have an ERP package, how do you integrate it with other database and business information systems and with other ERP packages?