Architecting Complex Systems for Robustness

By

Jason C. Slagle

B.S. Aerospace Engineering
The Pennsylvania State University, 1997

Submitted to the System Design and Management Program
In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management
at the
Massachusetts Institute of Technology

February 2007

Signature of Author: _____

Jason C. Slagle
System Design and Management Program
February 2007

Certified by: _____

Daniel D. Frey
Associate Professor of Mechanical Engineering and Engineering Systems
Thesis Supervisor

Certified by: _____

Edward F. Crawley
Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Supervisor

Accepted by: _____

Patrick Hale
Director
System Design and Management Program

Abstract

Architecting Complex Systems for Robustness

by

Jason C. Slagle

Submitted to the System Design and Management Program
February 2007 in partial fulfillment of the
Requirement for the Degree of Master of Science in
Engineering and Management

ABSTRACT

Robust design methodologies are frequently utilized by organizations to develop robust and reliable complex systems. The intent of robust design is to create systems that are insensitive to variations from production, the environment, and time and use. While this process is effective, it can also be very time consuming and resource intensive for an engineering team. In addition, most robust design activity takes place at the detailed design phase, when the majority of the product life cycle cost has already been committed. Addressing robustness and the "ilities" at the architecture level may be more effective because it is the earliest and highest leverage point in the product development process. Furthermore, some system architectures are inherently more robust than others. In this thesis, a framework based on principles is proposed to architect complex systems for type I and II robustness. The principles are obtained by tracing the architectural evolution of the jet engine, which is an extremely complex system that has evolved to high reliability. This framework complements existing robust design methods, while simultaneously incorporating the robustness focus earlier in the product development process.

Thesis Supervisor: Daniel D. Frey
Title: Associate Professor of Mechanical Engineering and Engineering Systems

Thesis Supervisor: Edward F. Crawley
Title: Professor of Aeronautics and Astronautics and Engineering Systems

Acknowledgements

The last two years have passed unimaginably fast, and much has been accomplished. I owe many people thanks for the support they have given to make all of this possible. This passage is an attempt to show, in some small way, my sincere appreciation.

I would like to thank Dan Frey and Ed Crawley for their lectures in system engineering and system architecting respectively, which have significantly improved my understanding of complex systems. They also provided much guidance, insight and feedback for this thesis.

MIT has taught me a great deal about the management of innovation, disruptive technologies and technology strategy. I am grateful for the fascinating presentations in these areas by professors Ralph Katz, Eric von Hippel, Clayton Christensen, Olivier de Weck and James Utterback among others.

I would like to thank Pat Hale for the excellent leadership he has provided to the SDM program. Also, I would like to thank my team members Jeff Lloyd, Paul Braunwart, Robbie Allen, Ben Entezam and Kevin Baughey for making numerous challenging projects both enjoyable and achievable.

I owe a debt of gratitude to all of my mentors, many of whom do not even realize they have this role. Their guidance has contributed considerably to my personal and professional development, and has further fueled my passion for technology and innovation.

My parents and family have supported me throughout my life and helped me to pursue my dreams. I am sincerely thankful and love you all.

Most importantly, I am grateful for the never-ending support and encouragement from my wife Elana. Thank you, and I love you.

Table of Contents

Abstract 2

Acknowledgements 3

Table of Contents 4

List of Figures 6

List of Tables 8

Nomenclature 9

1. Introduction 13

1.1 Background and Motivation 13
1.2 Objectives 14
1.3 Research Approach 15
1.4 Sources 18
1.5 Structure of Thesis 19

2. Literature Review 21

2.1 The Quality Literature 21
2.2 The System Architecture Literature 25
2.3 The Complex Systems Literature 26

3. System Architecture and Complex Systems 28

3.1 System Architecture 28
3.3 Complexity 31
3.4 Complex Systems 32
3.5 Emergence 34
3.6 Principles and Heuristics 36
3.7 Summary 37

4. Robustness and Reliability 38

4.1 Robustness 38
4.2 Reliability 40

5. Architectural Evolution of the Jet Engine 42

5.1 Introduction 42
5.2 Technical Overview of the Jet Engine 42
5.3 Architectural Innovation 49
5.4 Dominant Designs 51
5.5 Architectural Evolution 54
5.5 Summary 68

6. Principles of Robust System Architecture 70

6.1 Introduction to Principles of Robust System Architecture 70
6.2 Stability 72

6.3 Modularity 76
6.4 Feedback 84
6.5 Standardization 87
6.6 Redundancy 90
6.7 Simplicity 93
6.8 Independence 96
6.9 Autonomy 98
6.10 Scalability 99
6.11 Summary and Framework 102

7. The Product Development Process 104

7.1 The Product Development Process 105
7.2 Robust System Architecting Methodology 107
7.3 Design for Six Sigma 111

8. Conclusions and Future Work 116

8.1 Conclusions 116
8.2 Future Work 118

Bibliography 121

Appendix A. Technology Readiness Levels 129

Appendix B. Invention Levels 131

Appendix C. Jet Engines List 132

Vita 139

List of Figures

Figure 1. Robust Design in the Product Development Process (Modified from Fabrycky and Blanchard, 1991) 14
Figure 2. Engine Reliability Trend 16
Figure 3. Engine Life on Wing Trend 17
Figure 4. P-Diagram 23
Figure 5. Form - Function Relationship (Crawley, 2005) 29
Figure 6. Two Types of Aircraft Architecture (Modified from Cessna, 2006) 30
Figure 7. Evolution of Actual and Perceived Complexity (Crawley, 2005) 34
Figure 8. Illustration of Emergent Use of a Product System 35
Figure 9. Cross-Section of a Turbojet Engine (Century of Flight, 2006) 43
Figure 10. GE90 Turbofan Engine (Modified from GE, 2006) 43
Figure 11. Boeing 777-200LR Aircraft (Boeing, 2006) 44
Figure 12. Thrust-to-Weight Ratio Trend for GE Engines 47
Figure 13. Specific Fuel Consumption Trend 47
Figure 14. IFSD Rate for B727 with JT8D Engines 48
Figure 15. Types of Innovation 49
Figure 16. Architectural Innovation in Lawn Tractors 50
Figure 17. Trends in Maximum Speed of Propeller-Driven Aircraft (Donlan, 1954) 53
Figure 18. Trend in Thrust Specific Fuel Consumption (Koff, 2004) 56
Figure 19. GEnx Engine with Counter-rotating Spools (GE, 2006) 57
Figure 20. Secondary Flow Systems for the GE90-94B Engine (GE, 2006) 59
Figure 21. Number of Engine Control Functions (Koff, 2004) 60
Figure 22. Evolution of Engine Controls (Jaw and Garg, 2005) 61
Figure 23. Jet Engine Total Part Count 63
Figure 24. Number of Fan Blades for Large Turbofan Engines 64
Figure 25. Number of Compressor Stages for Large Turbofan Engines 65
Figure 26. Unique Part Numbers per Engine 66
Figure 27. Architectural Evolution of Jet Engine 68
Figure 28. Visualizing Stability 73
Figure 29. Engine Cross-Section Showing Fan Blades and Bearing Support (Doerflein et al., 2004) 75
Figure 30. Close-up of Bearing Support 76
Figure 31. Types of Modularity (Ulrich, 1995) 78
Figure 32. Propulsor Concept (GE, 2006) 79
Figure 33. P&W J57 Turbojet Engine (Pratt & Whitney, 2006) 81
Figure 34. The Off-Design Dilemma (Blanton, 2004) 82
Figure 35. Two-Spool Vs. One-Spool Architecture 83
Figure 36. Compressor Stall Map / Margin (Prasad, 2005) 83
Figure 37. Trend toward Faster Feedback (Modified from Fey and Rivin, 2005) 85
Figure 38. Trends in the Number of Engine Sensors 87
Figure 39. Boeing Study on 122 Maintenance Errors Occurring over 3 Years (Boeing, 1993) 89
Figure 40. Redundancy in the FADEC III Control Unit (Hispano-Suiza, 2006) 92
Figure 41. Aircraft Engine Starting Techniques 93

Figure 42. GE T700 Turboshaft Engine (GE, 2006) 95
Figure 43. T700 Compressor Blisk (DRS, 2006) 96
Figure 44. Types of Design Matrices (Suh, 1990) 97
Figure 45. Line of Transition to Active Adaptive Systems 98
Figure 46. Types of Scaling for Elements 100
Figure 47. Types of Scaling for Connections 100
Figure 48. GE Commercial Engine Development (Joyce and Rosario, 2002) 101
Figure 49. The Product Acquisition Process (CIPD, 2004) 104
Figure 50. Generic Product Development Process 105
Figure 51. Spiral Development (Boehm, 1988) 106
Figure 52. Generic Product Development Process (Crawley, 2005) 107
Figure 53. Robust Architecting Methodology 108
Figure 54. System Architecture Framework for Upstream Influences (Crawley, 2005) 109
Figure 55. System Architecture Framework for Downstream Influences (Crawley, 2005) 110
Figure 56. Incentive to Architect for Robustness (modified from Soman, 2004) 112
Figure 57. Relative Cost of Correcting an Error (Frey, 2005) 113
Figure 58. GE Design for Six Sigma Process (Soman, 2004) 114
Figure 59. Updated DFSS Process Incorporating Robust Architecting Methodology ... 115

List of Tables

Table 1. Information Sources for the Research 19
Table 2. Four Architecting Methodologies (Modified from Rechtin and Maier, 2002)... 36
Table 3. Definitions of Robustness (Santa Fe, 2001) 38
Table 4. Types of Robustness (Chen, et al., 1996) 39
Table 5. Examples of Architectural Innovation 51
Table 6. Dominant Designs for Jet Engines 52
Table 7. Sample Architectures that Are Excluded 53
Table 8. Robust System Architecture Principles 71
Table 9. Evolution of the Number of Engine Sensors 86
Table 10. Principles in P-Diagram Framework 102
Table 11. Description of Technology Readiness Levels (DoD, 2006) 129

Nomenclature

AB	Afterburner
AC	Advisory Circular
ACC	Active Clearance Control
AD	Airworthiness Directive
AFRL	Air Force Research Laboratory
AIAA	American Institute for Aeronautics and Astronautics
ALL	Allison
APST	Autonomous Propulsion Systems Technology
APU	Auxiliary Power Unit
ARINC	Aeronautical Radio, Incorporated
ARIZ	Algoritm Reshenia Izobretatelskih Zadach (Algorithm for Inventive Problem Solving)
ASME	American Society of Mechanical Engineers
ATO	Aborted Takeoffs
BIT	Built-In Test
Blisk	Integral blade and disk
BPR	Bypass Ratio
CAAM	Continued Airworthiness Assessment Methodologies Committee
CDP	Compressor Discharge Pressure
CFD	Computational Fluid Dynamics
CFR	Code of Federal Regulations
CLM	Component Level Model
CMC	Ceramic Matrix Composite
COTS	Commercial Off-The-Shelf
CPM	Critical-Parameter Management
CPR	Compressor Pressure Ratio
CTQ	Critical to Quality (Critical "Y")
DARPA	Defense Advanced Research Projects Agency
DEEC	Digital Electronic Engine Control
DFC	Design for Changeability
DFSR	Design for System Reliability
DFSS	Design for Six Sigma
DMADOV	Define Measure Analyze Design Optimize Verify
DMAIC	Define Measure Analyze Improve Control
DMICS	Design Methods for Integrated Control Systems
DoD	Department of Defense
DoE	Department of Energy
DoE	Design of Experiments
DOF	Degrees of Freedom
DP	Design Parameter
DSM	Design Structure Matrix
EASA	European Aviation Safety Agency
EBU	Engine Build Unit
ECS	Environmental Control System

ECU	Electronic Control Unit
EDU	Engine Diagnostic Unit
EEC	Electronic Engine Control
EGT	Exhaust Gas Temperature
EMS	Engine Monitoring System
EPP	Engine Program Plug
ELOS	Equivalent Levels of Safety
EPR	Engine Pressure Ratio
ETOPS	Extended Twin-engine Operations
FAA	Federal Aviation Administration
FADEC	Full Authority Digital Engine Control
FAR	Federal Aviation Regulation
FBD	Functional Block Diagram
FBO	Fan Bladeout
FEA	Finite Element Analysis
FDI	Fault Detection and Isolation
FM	Failure Mode
FMEA	Failure Modes and Effects Analysis
FOD	Foreign Object Debris
FR	Functional Requirement
GE	General Electric
HCF	High Cycle Fatigue
HER	Event History Recorder
HIDEC	Highly Integrated Digital Electronic Control
HISTEC	High Stability Engine Control
HMFC	Hydro-Mechanical Fuel Control
HMU	Hydro-Mechanical Unit
HOT	Highly Optimized Tolerance
HP	High-Pressure
HP	Horsepower
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
IAE	International Aero Engines
ICAO	International Civil Aviation Organization
ICE	Internal Combustion Engine
ICR	Inappropriate Crew Response
IDG	Integrated Drive Generator
IFSD	In-Flight Shutdown
IFPC	Integrated Flight and Propulsion Control
IGV	Inlet Guide Vane
IHPTET	Integrated High Performance Turbine Engine Technology
INA	Inverse Nyquist Array
INCOSE	International Council on Systems Engineering
IPCS	Integrated Propulsion Control System
ISA	International Standard Atmosphere
JAA	Joint Aviation Authorities

JAR	Joint Airworthiness Regulation
JSF	Joint Strike Fighter
LCC	Life Cycle Cost
LCF	Low Cycle Fatigue
LEC	Life-Extending Control
LOTC	Loss of Thrust Control
LP	Low-Pressure
LPC	Low-Pressure Compressor
LPT	Low-Pressure Turbine
LQR	Linear Quadratic Regulator
LRD	Load Reduction Device
LRU	Line-Replaceable Unit
MAPSS	Modular Aerospace Propulsion System Simulation
MBCD	Model-Based Controls and Diagnostics
MDO	Multidisciplinary Design and Optimization
MIL-HDBK	Military Handbook
MIL-SPEC	Military Specification
MIL-STD	Military Standard
MPH	Miles per Hour
MRO	Maintenance, Repair and Overhaul
MTBF	Mean Time Between Failures
MTBUR	Mean Time Between Unscheduled Removal
MVCS	Multivariable Control Synthesis
NACA	National Advisory Committee for Aeronautics
NASA	National Aeronautics and Space Administration
NGV	Nozzle Guide Vane
NPI	New Product Introduction
NPRM	Notice of Proposed Rule Making
NPSS	Numerical Propulsion System Simulation
NTSB	National Transportation Safety Board
OEM	Original Equipment Manufacturer
OPD	Object-Process Diagram
OPL	Object-Process Language
OPM	Object-Process Methodology
OPN	Object-Process Network
OPR	Overall Pressure Ratio
OW	Operating Window
P&W	Pratt & Whitney
PCA	Propulsion-Controlled Aircraft
PCC	Passive Clearance Control
PDE	Pulse Detonation Engine
PDP	Product Development Process
PHM	Prognostic and Health Management
PMA	Parts Manufacturer Approval
PR	Progress
PSC	Performance Seeking Control

PSM	Propulsion System Malfunction
PV	Process Variable
QFD	Quality Function Deployment
RLV	Reusable Launch Vehicle
RPM	Revolutions per Minute
RR	Rolls-Royce Plc
SA	System Architecture
Scramjet	Supersonic Combustion Ramjet
SDM	System Design and Management
SFC	Specific Fuel Consumption
S/N	Signal-to-Noise Ratio
TBC	Thermal Barrier Coating
TBO	Time Between Overhaul
TCF	Turbine Center Frame
TOW	Time on Wing
TQM	Total Quality Management
TRF	Turbine Rear Frame
TRIZ	Teoriya Resheniya Izobretatelskikh Zadach (Theory of Inventive Problem Solving)
TRL	Technology Readiness Level
TSFC	Thrust Specific Fuel Consumption
TSO	Technical Standard Orders
UAV	Unmanned Aerial Vehicle
UCAV	Unmanned Combat Aerial Vehicle
UDF	Unducted Fan
UER	Unscheduled Engine Removal
UHB	Ultra-high bypass
USAF	United States Air Force
UTRC	United Technologies Research Center
VAATE	Versatile, Affordable, Advanced Turbine Engine Program
VBV	Variable Bleed Valve
VIGV	Variable Inlet Guide Vane
VLJ	Very Light Jet
VSV	Variable Stator Vane
VTOL	Vertical Takeoff and Landing
WI	Williams International
W.U.	Whittle Unit

1. Introduction

"It is very expensive to achieve high degrees of unreliability. It is not uncommon to increase the cost of an item by a factor of ten for each factor of ten degradation accomplished."
- Norman Augustine, Augustine's Laws, Law Number XI

1.1 Background and Motivation

Complex systems permeate our lives and form the infrastructure for our society. Their ubiquity is evident as we use transportation systems for our daily commute, plug into the electrical grid for electricity and look up information on the Internet. Our dependency on these systems necessitates their robustness. The systems must be able to perform their intended functions despite changing environmental conditions, deterioration over time and changing user needs.

Robust design is the predominant methodology currently in use to engineer complex systems for robustness. It has been particularly effective, with Clausing and Frey (2005) noting "these methods seem to have accounted for a significant part of the quality differential that made Japanese manufacturing so dominant during the 1970s." Robust design has significantly improved quality, lowered production costs, lowered warranty costs and provided a competitive advantage for the manufacturer.

To improve the efficacy of robust design, progressive systems engineering innovations have occurred, pushing the entry point of robust design further upstream in the product development process. As illustrated in Figure 1, the earliest points in the product development process (PDP) offer the highest leverage because the cost committed and the cost incurred are very low. Over 66% of the product life cycle cost is committed at the conceptual design phase (Fabrycky and Blanchard, 1990). Hari and Weiss (1999) claim the cost committed at conceptual design is even higher, exceeding 75%. Starting with tolerance design in the production phase, robust design has migrated forward to include parameter design at the detailed design stage. More recently, researchers have shifted their attention to the conceptual design stage.

This research aims to continue the progression by determining if the robustness focus can be moved full forward to the system architecting phase. It is a complement to conceptual design in that conceptual design is a subset of the overall architecting process. By tracing the evolution of the jet engine, I studied whether complex systems could be *architected*, instead of *designed*, for robustness.

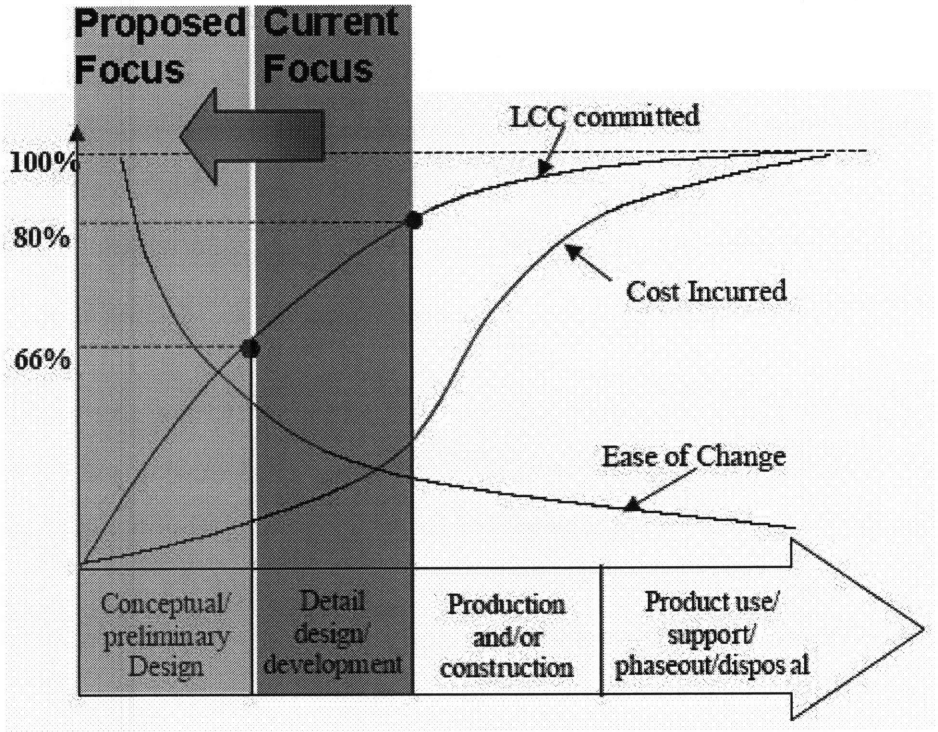


Figure 1. Robust Design in the Product Development Process (Modified from Fabrycky and Blanchard, 1991)

1.2 Objectives

Sahal (1985) argued that a "technology can properly function only for a particular combination of size and structure". As technological systems are scaled to improve performance metrics, he indicated that constraints will eventually be reached that can only be overcome with system, structural, and material innovations. Systems innovations are those that integrate multiple symbiotic technologies as a means to simplify the overall structure. Structural innovations address the non-uniform evolution of subsystems, or as

Sahal (1985) explains, "differential growth whereby the parts and the whole of a system do not grow at the same rate." These systems and structural facets of innovation speak directly to the system architecture of the product system. Along this line, the system architecture, with its composition of technologies, may define an upper bound on the performance of the system. Likewise, it is conceivable that one of these performance attributes is reliability, and its potential is established based on the architecture of the system.

Currently, robust complex systems are developed using robust design methodology. As mentioned in section 1.1, this thesis examines whether complex systems can be architected for robustness. Principles have been proposed as a way to guide the architecting process and manage complexity (Crawley, 2005). This thesis uses this approach to determine if a framework of "robustness principles" can be identified from the architectural evolution of the jet engine. In summary, the key research objectives are to:

- Determine if complex systems can be architected for robustness.
- Determine if principles can be identified that enhance robustness in complex systems.
- Determine if a "robustness" framework can be generated to guide the architect in creating robust systems.
- Determine if the robust design process can be moved forward in the product development process.
- Determine how the proposed principles and framework would integrate into the overall product development process.

1.3 Research Approach

Now that the objectives have been established for the thesis, we move on to the research approach. To identify principles of robust system architecture, the architectural evolution of the jet engine was researched. Why study the history of jet engines? Jet engines were examined because they are extremely complex engineering systems, with tens of thousands of parts, and they integrate mechanical, electrical and informational domains. Koff (2004) claims that the "gas turbine has evolved into the world's most complex

product which has made an astoundingly positive impact on mankind." They operate all around the world in tough and changing environments and as a result must be very robust. Robustness in jet engine systems is also critical to passenger safety. In spite of the complexity and challenging environments, jet engines have proven to be extraordinarily reliable. In order to be reliable, you must be robust and avoid mistakes (Clausing and Frey, 2005). One critical metric for engine reliability is the number of in-flight shutdowns (IFSDs). Another metric is the engine life on wing before a major overhaul or engine replacement is required. In both cases, the reliability improves exponentially over the past 50 to 60 years as illustrated in Figures 2 and 3 (Ballal and Zelina, 2004). While some of the improvement in robustness can be attributed to better tools, manufacturing and methodologies, it does not describe the whole picture and this study proposes that some of the improvement is attributable to architectural changes.

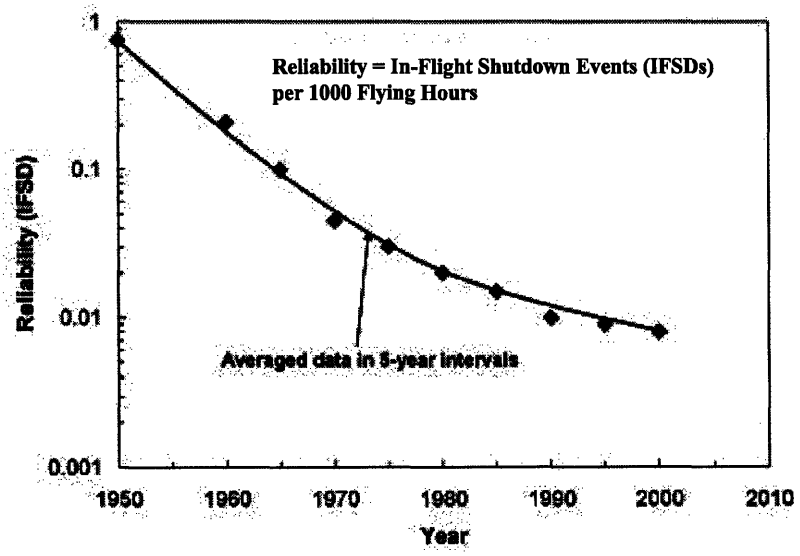


Figure 2. Engine Reliability Trend

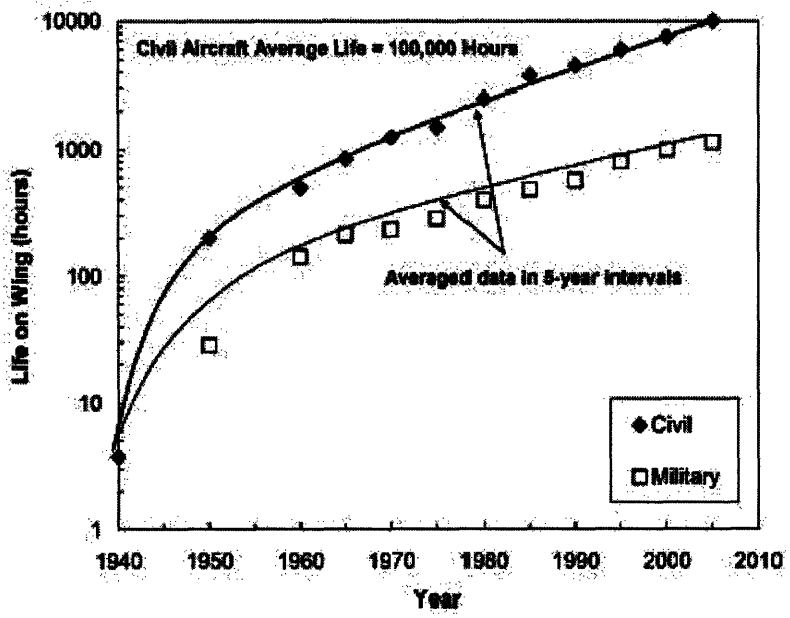


Figure 3. Engine Life on Wing Trend

The approach taken in this research leverages some aspects of work completed recently by Frey and Jugulum (2004) and Gomez (2005). These authors reviewed large bodies of patents to identify strategies for conceptual robustness. After strategies or principles were successfully identified, the authors developed a useful classification framework. The specific steps of the research approach taken here are as follows:

1. **Collect data on the history and evolution of jet engines with a specific focus on architectural changes and their effect on reliability and robustness.** Patents, as mentioned above, are one good information source for robustness oriented innovation and will be used here whenever possible. However, patents typically cover technology advances and may not capture architectural changes. An excellent source for reliability and architectural data is the engine manufacturers, but proprietary restrictions frequently preclude this option. As a result of these limitations, I employ a number of different sources to aggregate both qualitative and quantitative data about jet engines. The wide spectrum of sources used for this research is covered in section 1.4.

2. **Analyze the data to identify principles that have improved robustness from a system architecture perspective.** In this step, I mine the information to extract principles, and as they are identified, assess whether some principles are subsets of others. I strive to identify top-level principles so that they will have the maximum impact for the architect. For each principle, I identify at least one key case study to demonstrate its effectiveness.
3. **Determine which aspects of robustness are improved by the principles.** The intent of this research is to identify principles that facilitate type I robustness, type II robustness, or both during the architecting process. It is also an objective to understand which specific aspects of robustness are enhanced by the given principle.
4. **Create a useful framework from the principles.** Principles will help the architect, but to be really usable an integrative framework must be created. This framework should organize the principles in a useful manner and answer the questions of how, when and where to apply the information.
5. **Incorporate the framework into the product development process to create an enhanced robust design process.** I propose to add synergistically to the existing robust design process to further improve the effectiveness of the methodology. For this to occur, it is necessary to understand how the framework merges with robust design as well as the overall product development process.

1.4 Sources

A wide array of sources was leveraged to obtain the data for this study. This cross-section was chosen to ensure as much of the jet engine's history as possible was covered. In addition, it offered multiple viewpoints that in combination may have provided further insights. As noted in section 1.3, some limitations existed, because architectural innovations are not always codified in patents and the engine manufacturers frequently consider reliability and robustness data to be proprietary. However, there is a rich reservoir of information and data on jet engine evolution in the open literature. Numerous books and journal articles have been written on this revolutionary technology as well as its disruptive effect on the aircraft industry. AIAA and ASME journal articles

and conference proceedings are particularly helpful in this respect. In addition, the government agencies such as NASA, FAA, JAA and NTSB are excellent resources. As a final note, the primary author of this thesis has been designing and developing jet engines at GE for nearly a decade, which opened up several additional channels. In this capacity, it was possible to interview chief engineers, review GE engine data and spend considerable time walking through the GE Aviation Museum which houses engines from all periods. The ability to physically inspect the engines significantly facilitated understanding of the architecture and architectural changes. A partial listing of the resources utilized in this study is summarized in Table 1.

Table 1. Information Sources for the Research

AIAA, ASME and other journals
Patents
GE Data
Product literature
FAA, FARs
NTSB
NASA and NASA Technical Reports
GE Chief Engineers
GE Consulting Engineers
Six Sigma Training
U.S. Air Force Museum
GE Aviation Museum
Lean Six Sigma Experts
An AIAA historian
Open literature (books, papers)
Flight Safety
Conference Proceedings
JAA
EASA
The Internet
Airframers
Airlines

1.5 Structure of Thesis

In the present chapter, an introduction to the thesis topic of robust system architecture was provided and its relationship with robust design was briefly described. Section 1.2

summarized the pertinent research objectives and section 1.3 gave an overview of the research approach. Section 1.4 covered the diversity of resources used in the data collection process. In the following chapters, I will cover the material listed below.

- Chapter two reviews the literature and delves into the various manifestations of robust design. Alternate perspectives of robustness and architecture are captured through the lenses of the system architecture and complex systems literature.
- Chapter three describes system architecture for complex product systems and gives a background on the properties and attributes of complex systems.
- Chapter four reviews the different definitions of robustness and reliability from various domains and pinpoints the definitions used in the present research.
- Chapter five steps through the history of the jet engine and reviews the predominant architectural changes and innovations that have taken place since its inception. The dominant designs established in the marketplace are also reviewed.
- Chapter six proposes a set of principles that can be incorporated into the system architecture of complex technological systems to generate robust products. Each principle is accompanied with a detailed example from the aircraft engine industry.
- Chapter seven takes a step back and evaluates the framework in the context of the entire product acquisition process to establish higher level synergies. At this point, a robust system architecting methodology is introduced that integrates the framework and principles into the product development process for maximum benefit to the organization. To further increase the positive impact of the framework, robust system architecting methodology is also integrated into the design for six sigma process, which is a common robust design process in many business organizations.
- Chapter eight summarizes the key findings of this study and offers recommendations on next steps for the organization to make best use of the material. In addition, research items are identified for future work.

2. Literature Review

"Everything has been said before, but since nobody listens we have to keep going back and beginning all over again."
- Andre Gide

2.1 The Quality Literature

The quality literature has a multitude of information in the areas of process improvement and robust design. In this literature search, I concentrate on the robust design aspect, as this paper proposes to move the robustness-oriented activity earlier in the product development process to the system architecting phase. Robust design aims at creating systems that are insensitive to environmental noise factors, production variations and deterioration.

Dr. Genichi Taguchi (1988, 1993) was a pioneer in the field of robust design and developed a quality process comprised of the following three phases:

1. Concept / System design.
2. Parameter design.
3. Tolerance design.

We will take these in reverse order, starting with tolerance design, and working up to the beginning of the product development process. This approach is analogous to the development of the field of robust design.

Tolerance design looks at understanding the possible variation that may occur during production. At this stage, the geometric tolerances are reviewed and critical component dimensions are identified, prioritized in a pareto-optimal fashion and controlled. Authors such as Ostwald and Huang (1977) and Michael and Siddall (1981) have generated detailed methodologies for optimal tolerance assignment and selection.

Parameter design occurs after the concept and system level design have been established, and signals the onset of the detailed design phase. The objective of

parameter design is to develop a product system that will function properly in the presence of noise factors and variation through the careful manipulation of control factors in the design. More precisely, Taguchi indicates that the signal-to-noise ratio should be maximized. In some instances, the level of a design parameter / control factor affects the sensitivity or variation in the system. Therefore, these parameters must be selected to minimize variation while meeting program cost and timing constraints. As described earlier, Phadke (1989) prescribed the P-diagram (see Figure 4) as a visual aid in understanding the various categories of factors affecting the system response. Comprehensive methodologies for achieving robustness through parameter design have been compiled by Fowlkes and Creveling (1995) and also Wu and Hamada (2000).

Concept design occurs early in the product development process when the required functionality is mapped to a conceptual form. Actions at this phase offer higher leverage on the development process, and as a result, researchers are actively engaged in bringing these benefits to fruition. Whereas the prior two phases were conducive to the use of analytical tools from probability theory, concept design requires an alternate approach as it is both art and science. It requires rigorous processes but also creativity and judgment. Robust design at the conceptual design phase is the most closely related methodology to the principle-based approach that is proposed in this work. To discern the similarities and differences, we will review these authors' research more closely.

Jugulum and Frey (2004) have developed concept design strategies by studying an abundance of patents. They then generated a taxonomy of concept designs, based on the P-diagram (Phadke, 1989), that promote robustness. Because the P-diagram is widely used and understood by the engineering community, it is an excellent means to introduce new concept design approaches. I use this framework in the present study, albeit in a slightly different manner that will be expounded in chapter six.

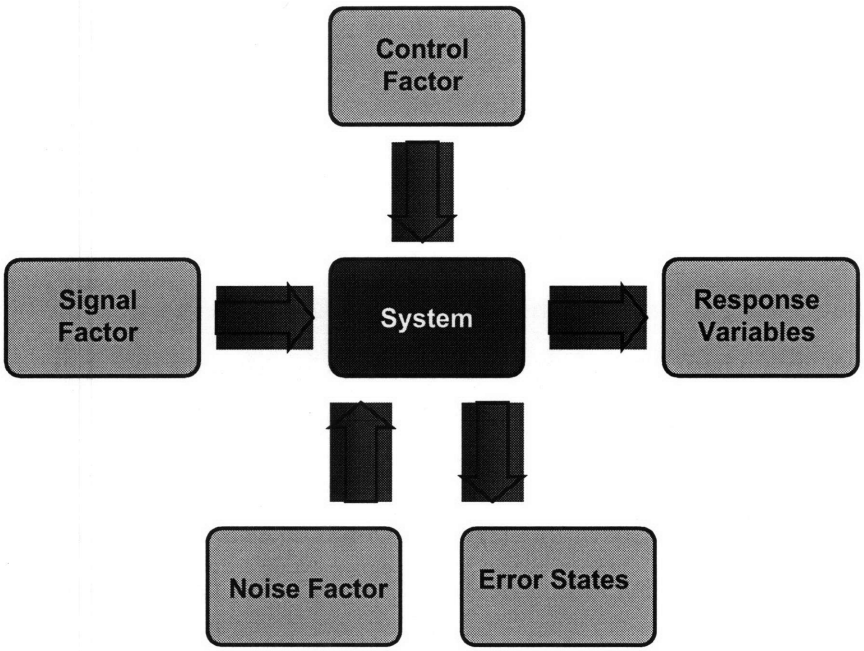


Figure 4. P-Diagram

Gomez (2005) also studied the patent literature to obtain concepts for "robustness inventions". These concepts were divided into two categories – intents to reduce system sensitivity to noise, and intents to reduce component sensitivity to noise. This work adds to the body of knowledge and framework initiated by Frey and Jugulum (2004).

Clausing and Frey (2005) developed strategies for improving robustness in the form of failure mode avoidance, and subsequently demonstrated the strategies using printer and jet engine examples. Their approach is to avoid one-sided failure modes in the concept phase by making design parameter or even functional changes to the system.

Six sigma. Highly structured quality processes and frameworks have appeared in business settings over the years, with the most prevalent being six sigma. Many large corporations have become practitioners of six sigma including General Electric, Raytheon, Motorola and Honeywell. Six sigma is a quality initiative started in the mid 1980s at Motorola to reduce defects, process variation and product development time (Motorola, 2006). While the integrated methodology was new in the business context, it

leverages all of its tools and techniques from the existing body of quality engineering knowledge.

Within the six sigma paradigm, two predominant processes exist, namely DMAIC and DMADOV. DMAIC is an acronym for "Define, Measure, Analyze, Improve and Control" and its goal is to improve an existing product or process within the business. An application area where DMAIC has been particularly effective is manufacturing processes. DMADOV, which stands for "Define, Measure, Analyze, Design, Optimize, Verify", is a methodology to develop new products or processes. DMADOV falls under the larger veil of Design for Six Sigma (DFSS), which takes a systems engineering approach to designing and developing new products, services and processes. DFSS and its use of robust design has been codified in great detail by Slutsky and Creveling (2002) among others.

As stated above, six sigma is a framework comprised of many quality and robust design methodologies, including some of those proposed by Taguchi. One critical difference between the Taguchi methodology and the six sigma framework is that Taguchi's intent is to maximize signal-to-noise (S/N) ratio, whereas six sigma's objective is to maximize a Z-score.

The methods covered in this section address robustness in various parts of the product development process, with most emphasis placed on detailed design. New research has begun to prioritize the conceptual design phase for robustness. However, none effectively address robustness at the system architecture level. As evidenced by the patent focus that several have taken, many of the concepts involve the introduction of a new technology. Robustness at the system architecture level involves the integration of technologies and the establishment of form in addition to concept generation. The technologies may all be pre-existing and simply integrated or linked in a unique fashion.

2.2 The System Architecture Literature

Design for Changeability (DfC) focuses on the characteristics of robustness, flexibility, agility and adaptability. This framework, first proposed by Fricke (1999), was further developed by Fricke and Schulz (2005) to include three basic principles and six extending principles that facilitate the various aspects of changeability. While DfC focuses primarily on changeability, it does touch on robustness and as such offers a good starting point for this research. The DfC framework does not discuss the different facets of robustness and how the framework influences them. It also does not describe the relationship with robust design and whether or not the approaches can be implemented in a complementary fashion. Additionally, the possibility of other robustness principles needs to be further explored. These are key pieces needed for a framework centered on architecting robust complex systems.

Crawley et al. (2004) discuss architecture as a way to design and manage complex systems. They note that properties of a system, such as robustness and adaptability, can be affected by architecture, but indicate that no consensus has been reached as to whether robustness can be achieved by architectural form alone.

Maier and Rechtin (2002) describe the overall system architecting process in great detail and explain how heuristics can be used to guide the process. They also pull together a myriad of sample heuristics from different domains, and categorize them by their approximate application point in the architecture process. While Maier and Rechtin provide an excellent overview on heuristics, they do not address heuristics in the context of robustness.

TRIZ (pronounced *treez*) has recently made inroads into system architecture, robust design and product development research due to its systematic approach to invention and problem solving. It is included in this section because the technical creativity emphasis meshes well with the architecting process. Genrikh Altshuller developed TRIZ by analyzing a portfolio of patents from around the world and identifying approximately 1500 technical contradictions that could be solved with a set of fundamental principles

(Altshuller, 1984, 2000). In addition, he developed laws of technological evolution that have been described in greater depth by Fey and Rivin (2005). Clausing and Fey (2004) have worked to integrate TRIZ and robust design into the early stages of product development. Some of the TRIZ tools and techniques will be further elaborated upon in chapter five. TRIZ attempts to resolve technical contradictions, and this can be an especially useful tool for the architect. However, the architecting process is much more encompassing than solely technical contradictions, and it requires a more holistic and integrative approach.

2.3 The Complex Systems Literature

The scientific literature contains an extensive body of research on robustness of complex systems. Much of this information centers on the analysis of biological, ecological and social systems. Some researchers have taken an analytical approach by applying random graph theory, percolation theory, and multiple models to understand network topology (Albert and Barabasi, 2002). They note that dissimilar topologies have different reactions under failure conditions or attack.

Carlson and Doyle (1999, 2000) argue that highly optimized tolerance (HOT) systems can be robust to variations for which they were designed, but very fragile to unknown or emergent variations. Willinger and Doyle (2002) investigate the history of the Internet to gain an understanding of robustness and complex systems. They identify different connotations of robustness including flexibility, survivability and simplicity and generate corresponding models.

Simon (2000) investigated principles of complex system design, all of which were culled from natural systems. He identifies homeostasis, membranes, specialization and near-decomposability as principles, but notes that this is merely a starting point and others will be discovered. His research does not delve deeply into robustness, but instead focuses on general principles observed in nature.

Many of these studies are scientific explorations that provide useful insight for the scientist, but provide little guidance for the architect and engineer. Another complication

is that the complex systems literature is based primarily on analyses of natural systems. While this contributes to an understanding of the science of complex systems, it does not treat robust design or more importantly fill the knowledge gap for architecting robust engineering systems.

3. System Architecture and Complex Systems

"Architecture is the learned game, correct and magnificent, of forms assembled in the light."
-Le Corbusier

In this chapter, I endeavor to answer questions such as "what is system architecture?" and "what are some of the attributes of architecture?" I discuss the salient terminology for the architect and then explore some of the features of complex engineering systems. Many people envision buildings, pristine churches and novel homes designed by Frank Lloyd Wright when they think of architecture. Civil architecture is a long-standing and well-known facet of this field, but it is a small subset of the total realm. Rechtin (1991) observes that "Architecting, the planning and building of structures, is as old as human societies – and as modern as the exploration of the solar system." All systems have architecture, and this will be discussed in more detail below.

3.1 System Architecture

A number of system architecture definitions exist in the literature of varying hues. To give the reader a better overall understanding of the topic, a handful of these definitions are covered here.

"The embodiment of concept, and the allocation of physical/informational function to elements of form, and definition of interfaces among the elements and with the surrounding context." (Crawley, 2006)

"Architecture: The structure (in terms of components, connections, and constraints) of a product, process, or element." (Rechtin and Maier, 2002)

"Architecture: The organizational structure of a system...identifying its components, their interfaces, and a concept of execution among them." (MIL-STD-498)

"Systems architecture: The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors." (INCOSE, 2006)

Reviewing the definitions, there are subtle differences, but we find more commonality than dissonance. System architecture is the form or structure of a system that defines the interfaces, the components, the relationships between the components, and the relationship between the system and its context as a means to fulfill its required functions.

As Crawley (2005) indicates, the architecture also embodies the concept, which is the vision or mental image for mapping function to form as illustrated in Figure 5. Dori (2002) describes concept as "the system architect's strategy for a system's architecture."

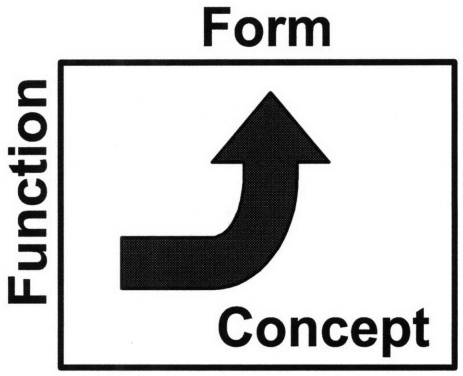


Figure 5. Form - Function Relationship (Crawley, 2005)

The functions are the operational requirements of the complex system and are defined early in the program. They answer the question "what do we need the system to do or accomplish?" The architect determines needs from the customer or stakeholders, translates these needs into goals and then establishes functional requirements. The functional requirements are stated in so-called solution neutral form to obviate preconceived solutions.

Form is an attribute of the system that constitutes the actual physical or informational embodiment (Crawley, 2005). It communicates the structure and arrangement of the

elements. The architect can decompose form into smaller units, all the way down to the individual elements if needed. The word "element" is used here to mean an atomic unit of the system.

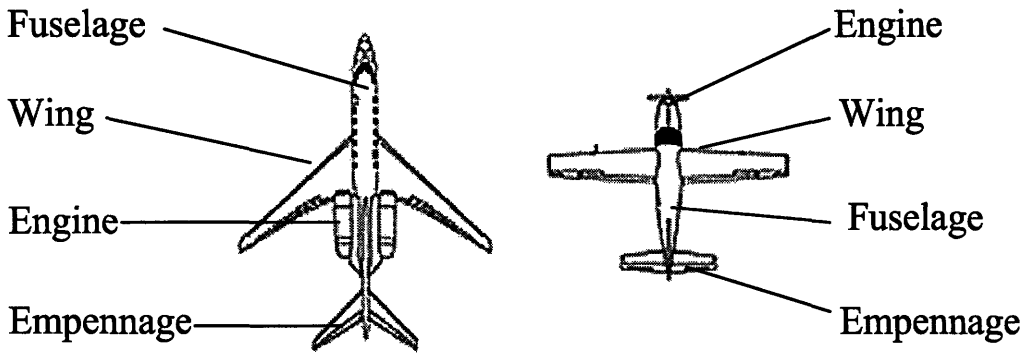


Figure 6. Two Types of Aircraft Architecture (Modified from Cessna, 2006)

As mentioned above, all systems have architecture. Figure 6 shows two sample architectures of small, fixed-wing aircraft, with each aircraft system configured for different uses and flying characteristics. The business jet on the left is architected to transport a small number of businessmen quickly from point-to-point. It utilizes a long, slender fuselage to accommodate up to a dozen passengers in addition to a pilot and copilot. The wings are swept and optimized for high-speed flight, with the aircraft powered by two small jet engines located aft of and above the wing. In contrast, the picture on the right is representative of a private, sport aircraft. This architecture provides a low-cost aircraft for a pilot and perhaps a few passengers for occasional flights, many of these being in the leisure category. The aircraft has a high-wing with a rectangular planform and thick camber for slow and stable flight. The propulsion system is a single piston engine with a propeller to keep costs down while providing adequate thrust. While the business jet locates the two smaller diameter engines aft of the wing, the sport aircraft places the larger diameter propeller and piston engine at the front of the fuselage.

Some other examples of systems having an architecture include computers, ecosystems, financial systems, social networks and the electrical grid. While some of these systems

have evolved over time, others were engineered deliberately. Some engineered systems also evolve over time, one category being infrastructure systems. Systems are architected to meet their functional requirements, but the architecture also greatly influences the "ilities", which are especially important for long-lived systems such as infrastructure (Ulrich and Eppinger, 2004; Crawley et al., 2004). The "ilities" refers to various aspects of the life cycle of a complex system such as reliability, robustness, repairability, maintainability, operability and durability. This point is key to this thesis as I am proposing a framework to architect systems for robustness.

The system architecture plays a critical role in establishing the functional behavior, while strongly affecting the "ilities", complexity level and emergent behavior for a system (Crawley et al., 2004). We have covered the functionality aspect of a complex system in this section and also touched on the "ilities". In sections 3.3 and 3.4, an overview of complexity and complex systems respectively is presented. Section 3.5 covers emergent behavior and the benefits and challenges it poses for the architect.

3.3 Complexity

As we are examining complex systems, it is necessary to clarify the meaning of complexity and review its elusive attributes. Definitions from the system architecture literature are as follows:

"Complexity: A measure of the numbers and types of interrelationships among system elements. Generally speaking, the more complex a system, the more difficult it is to design, build, and use." (Rechtin and Maier, 2002)

"Complexity: Having many interrelated, interconnected or interwoven elements and interfaces...an absolute and quantifiable system property." (Crawley, 2005)

"A set of structure-based metrics that measure the attribute of the degree to which a system or component has a design implementation that is difficult to understand and verify." (IEEE, 96)

The above definitions in combination with those from the complex systems literature indicate that complexity is a metric characterizing the amount of information needed to completely describe the state of a system. As the information content increases, the complexity increases. Several authors (Lankford, 2003; McCabe, 1989; Suh, 1999) have created highly developed methodologies for computing complexity, although they differ on how the information content is quantified. Most of these treatments are based on algorithmic complexity theory.

Boothroyd and Dewhurst (1987) developed a simple part counting method to quantify complexity. Given the number of parts (N_p), the number of types of parts (N_t) and the number of interfaces (N_i), the complexity factor is:

$C = (N_p * N_t * N_i)^{1/3}$

Crawley (2006) also offers a simple metric for determining complexity. Given the number of things (N_{things}), the number of connections among things ($N_{connections}$) and the number of types of connections ($N_{types\_of\_connections}$), the complexity is:

$C = N_{things} + N_{types\_of\_things} + N_{connections} + N_{types\_of\_connections}$

In this section, I provided a brief overview of complexity and presented a few simple metrics for quantifying the complexity level. I next discuss complex systems and introduce additional descriptors of complexity including apparent and essential.

3.4 Complex Systems

With complexity introduced above, we now move on to complex systems. We start by discussing the definition of a system.

"A collection of things or elements which, working together, produce a result not achievable by the things alone." (Rechtin and Maier, 2002)

"A set of interrelated elements that perform a function, whose functionality is greater than the sum of the parts" (Crawley, 2005)

These definitions converge nicely on precisely what constitutes a system. We see three main pieces: A set of one or more elements, that are interacting together and providing functionality greater than the sum of the constituents. Next, we identify the characteristics of a *complex* system, or more specifically, a complex engineering system.

Ulrich and Eppinger (2004) remark that a complex system "must be decomposed into several subsystems and many components." They continue saying these "subsystems and components are developed by many teams working in parallel, followed by system integration and validation."

Crawley (2005) states that a complex system "requires a great deal of information to specify." He then proposes a rule of thumb for system classification based on the architectural form:

- Simple systems: (7 ± 2) elements
- Medium systems: $(7 \pm 2)^2$ elements
- More Complex systems: $(7 \pm 2)^3$ elements

An example of a simple system is a pencil, while a skateboard is representative of a medium system. As mentioned in the introduction, complex systems surround us and are frequently readily identifiable. Satellite constellations, the electrical grid and jet engines are all examples that easily surpass the $(7\text{-}2)^3$ or 125 element criteria for a complex system.

It is important to note that Crawley (2005) distinguishes between three types of complexity: essential, actual and perceived. Essential complexity is the minimum complexity level necessary for a system to perform its intended functions. The definition of actual complexity is straightforward; it is the amount of complexity in the system. Perceived complexity is the complexity level one perceives when examining the system

in an abstracted format. Figure 7 illustrates the different complexity types and their trends in the product development process. Crawley (2005) recommends that systems are architected so the perceived complexity does not exceed the limit of understanding, otherwise a system becomes both complex and *complicated.* Secondly, he suggests that the actual complexity be kept as close as possible to the essential complexity. To aid the architect in managing complexity, he offers three approaches: abstraction, decomposition and hierarchy. I explore these approaches in chapter six and discuss how complexity can be traded for robustness and the "ilities".

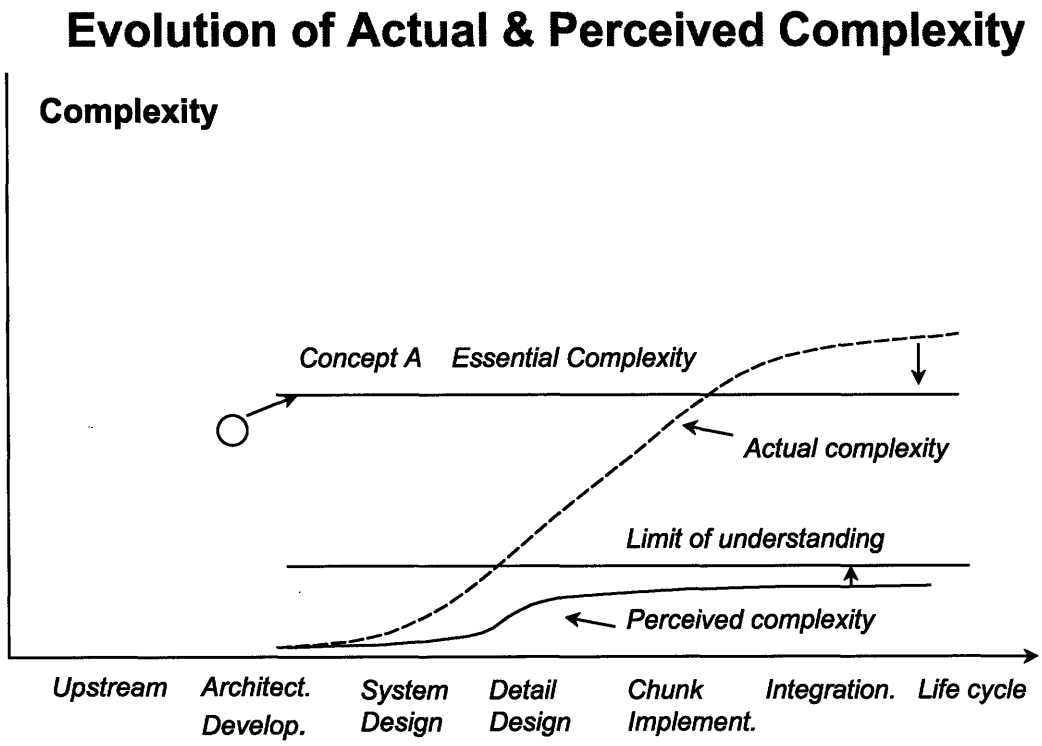


Figure 7. Evolution of Actual and Perceived Complexity (Crawley, 2005)

3.5 Emergence

Complex systems can be designed for specific purposes, but with sufficient complexity it becomes extremely difficult to predict all of the behaviors that the system will exhibit over time. Gestalt theory tells us that the whole is greater than the sum of the parts. Similarly, a complex system comprised of multiple components will have attributes and

functionality different than the combination of attributes of the components taken individually. Additionally, the system may be utilized in novel ways and combined with other components to form new systems. These previously unknown behaviors that appear are known as emergent properties or functionalities.

In a System Architecture lecture at MIT, Steve Imrich conveyed the essence of emergence by showing the postcard in Figure 8 (Crawley, 2005). An airplane's wing was not designed to support a tennis match, and certainly not during flight. As stated earlier, a product system will be used in ways during its life cycle that the designers and architects could not have foreseen.

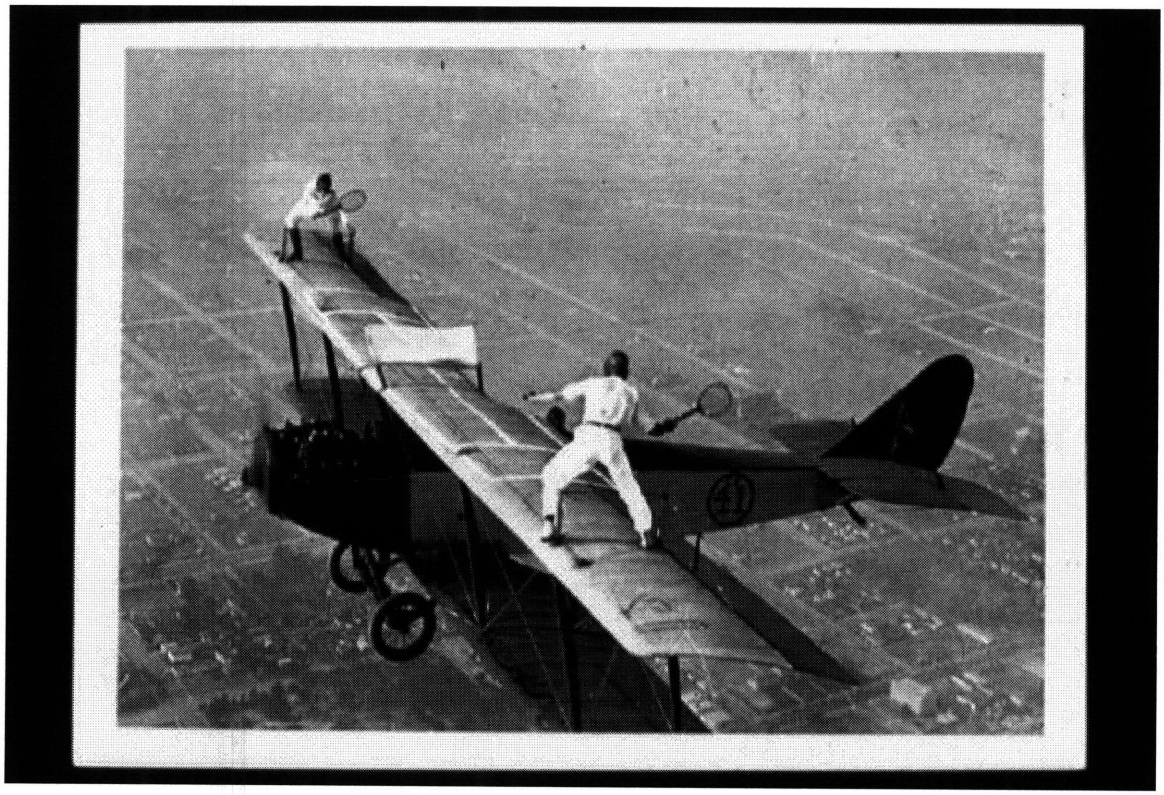


Figure 8. Illustration of Emergent Use of a Product System

Emergence can bring forth both desirable and undesirable functions. The prior example was undesirable because a design team would be ill-prepared to generate an architecture that could exhibit robustness to a near infinite number of unknowns. However, in the course of developing a common engineering system, one takes advantage of the desirable

traits of emergence. The components are brought together to create a system with a new emergent process. Fan blades are assembled around a rotating shaft in a jet engine system to aid in thrust production. The architect can *zoom* the desired emergent process to find its constituent sub-processes. Zooming is analogous to decomposition of function (Crawley, 2005).

Emergence presents a challenge for engineers attempting to "build-in" robustness. Functions do not add linearly, so it is difficult to predict emergent functionality *a priori* (Crawley, 2005). With unknown-unknowns both internal and external to the system, how can we confidently architect and design robust systems? This brings us to the topic of section 3.6 – principles.

3.6 Principles and Heuristics

As one begins to architect a system, he will want to follow a process to guide his work and ensure the greatest expected value of utility for the stakeholders. There are four key categories of architecting methodologies that exist, and these are summarized in Table 2 (Rechtin and Maier, 2002; Lang, 1987; Rowe, 1987).

Table 2. Four Architecting Methodologies (Modified from Rechtin and Maier, 2002)

Type	*Description*	*Examples*
Normative	Solution-based	Building Codes, Communications Standards
Rational	Method-based	Systems Analysis, Engineering
Participative	Stakeholder-based	Concurrent Engineering, Brainstorming
Heuristic	Lessons learned	Keep it simple.

In this paper, the heuristic, or principle-based, approach is used to achieve robustness through systems architecting. Heuristics emerge as the lessons learned and the wisdom from many years of experience, and are qualitative in nature. They are time-tested principles that guide the system architect towards a successful solution for a given problem domain. Crawley (2005) defines principles as "the underlying and long enduring fundamentals that are always (or almost always) valid." As principles are qualitative in

nature, they cannot be "proven". However, some principles have lasted thousands of years, which gives us some level of confidence that they approach "truth".

One sample principle relevant to the topic of robustness is "robust functionality drives essential complexity" (Crawley, 2005). As mentioned in section 3.4, essential complexity is the minimum complexity or information level necessary in a system architecture for the system to perform its intended operations. Robust functionality, the ability of the system to function in the face of variability, acts to increase the required complexity because additional functions are needed to compensate for the variability.

The proposed framework in this study is based on the heuristic approach, and principles are codified that help the architect achieve system robustness. Both the principles and framework are summarized in chapter six.

3.7 Summary

This chapter began with a description of system architecture, and it was followed with a discussion of the mapping of function to form through concept. I next defined complexity and gave an overview of complex systems. After categorizing the different types of system complexity, the concept of emergence was explored. Now that we have established the essential background for system architecture and complex systems, we proceed to evaluate its interrelationship with robustness in chapter four.

4. Robustness and Reliability

"When a company's products are robust – highly functional, elegant in their design, fairly priced, and a pleasure to use – the corporation itself will be equally robust."
-Meyer and Lehnerd (1997)

4.1 Robustness

As the focus of this work lies in architecting *robust* complex systems, I now seek to provide the reader with a better understanding of what robustness entails, and how it may be imparted into a system. Many different definitions exist for robustness in the literature, and in this chapter I establish the characteristics incorporated into the present study. The Santa Fe Institute compiled the following seventeen definitions from diverse domains such as robotics, software, ecosystems and other complex systems:

Table 3. Definitions of Robustness (Santa Fe, 2001)

1	Robustness is the persistence of specified system features in the face of a specified assembly of insults.
2	Robustness is the ability of a system to maintain function even with changes in internal structure or external environment.
3	Robustness is the ability of a system with a fixed structure to perform multiple functional tasks as needed in a changing environment.
4	Robustness is the degree to which a system or component can function correctly in the presence of invalid or conflicting inputs.
5	A model is robust if it is true under assumptions different from those used in construction of the model.
6	Robustness is the degree to which a system is insensitive to effects that are not considered in the design.
7	Robustness signifies insensitivity against small deviations in the assumptions.
8	Robust methods of estimation are methods that work well not only under ideal conditions, but also under conditions representing a departure from an assumed distribution or model.
9	Robust statistical procedures are designed to reduce the sensitivity of the parameter estimates to failures in the assumption of the model.
10	Robustness is the ability of software to react appropriately to abnormal circumstances.
11	Robustness of an analytical procedure is a measure of its ability to remain unaffected by small, but deliberate variations in method parameters, and provides

	an indication of its reliability during normal usage.
12	Robustness is a design principle of natural, engineering, or social systems that have been designed or selected for stability.
13	The robustness of an initial step is determined by the fraction of acceptable options with which it is compatible out of total number of options.
14	A robust solution in an optimization problem is one that has the best performance under its worst case (max-min rule).
15	"..instead of a nominal system, we study a family of systems and we say that a certain property (e.g., performance or stability) is robustly satisfied if it is satisfied for all members of the family."
16	Robustness is a characteristic of systems with the ability to heal, self-repair, self-regulate, self-assemble, and/or self-replicate.
17	The robustness of language is a measure of the ability of human speakers to communicate despite incomplete information, ambiguity, and the constant element of surprise.

The definition of robustness adhered to in this paper is two-part. The first part is: Robustness is the ability of a system to perform its intended functions in the presence of environmental noise, internal noise, variations in production and variations resulting from time and use. This definition falls under the category of Type I robustness as illustrated in Table 4 (Chen, et al., 1996).

The second part covers variation in design variables or control factors, sometimes referred to as the adaptability of the system. This part is congruent with Type II robustness (see Table 4). I do not, however, include Type III robustness in this research as the qualitative aspect of the architecture principle methodology is not readily amenable to quantification.

Table 4. Types of Robustness (Chen, et al., 1996)

Three Types of Robust Design	
Type I	Uncertainty in noise or environmental and other noise factors
Type II	Uncertainty in design variables or control factors
Type III	Uncertainty introduced by modeling methods

Type II robustness is important because complex systems can be long-lived, experiencing change to the system itself as well as changes to the system's context. Common examples of long-standing systems include highway systems, satellite constellations and offshore oil platforms. Technologies evolve along their respective "S-curves", creating

new utility as well as new integration challenges for the system. Customers' use profile of the system may also change creating emergent processes and exposing the system to new environments. Type II robustness gives the system flexibility and adaptability so that it can meet these environmental changes. Type II robustness is also important because it is the primary form of robustness that addresses the "ilities". The reader will note that the use of type II robustness here refers to control factor changes that happen before or after the product development process. This differs from Chen's intent to only be robust to control factor changes prior to release of the product.

In summary, the robustness framework and principles proposed in this thesis aim to improve both type I and II robustness.

4.2 Reliability

We extend our discussion of robustness into the area of reliability not only because it is a critical aspect of robust design, but also because reliability is what the market "sees". Reliability is commonly measured in industries, whereas robustness data can be obscure and difficult to obtain. In this respect, reliability enhances our ability to understand robustness changes in architecture. We now define reliability and discuss its relationship with robustness.

"Reliability is the proper functioning of the system under the full range of conditions experienced in the field." (Clausing and Frey, 2005)

"Reliability is the extent to which an experiment, test, or measuring procedure yields the same results on repeated trials." (Merriam-Webster, 2006)

We will adopt Clausing and Frey's definition for this research, along with their two requirements for reliability. These are:

1. Mistake avoidance
2. Robustness

The goal of the first requirement, mistake avoidance, is to eliminate errors that occur during the design process, during production or by the end-user. During the design process, decisions can be in error or based on the wrong data. In production, holes can be misdrilled and bolts can be installed backwards. The second requirement implies that robustness is necessary, but not sufficient, to achieve reliability. This is very useful as it enables us to exploit reliability data to discern the effects of architectural changes and innovations on robustness, as long as we also review the mistake avoidance component.

Recent jet engine models exhibit outstanding reliability, making them an excellent subject for further study. The GE90-115B turbofan engine, for example, has an engine departure reliability of 99.97% and has not experienced any in-flight shutdowns (GE, 2006). In the next chapter, we explore jet engine history more deeply to identify architecture, reliability and robustness trends.

5. Architectural Evolution of the Jet Engine

"Know how to solve every problem that has been solved."
- Richard Feynman

5.1 Introduction

Up to this point, we have worked to establish the foundation of the thesis, describing robust design and the attributes of system architecture, complex systems and robustness. We now switch gears to review the jet engine, which is the complex system from which our architecture principles are drawn or verified. We will start off by discussing the high-level technical features of the jet engine, and then walk through the primary modules that comprise the engine. After quickly reviewing the history of the jet engine, we will examine the dominant architectures that have been established. We will also clarify which specific types of engines are included and which are excluded from our analysis. This is followed by a discussion of the key engine reliability metrics. Finally, we will review the historical trends and summarize our observations. In chapter six, we will delve into the principles and framework drawn from this analysis.

5.2 Technical Overview of the Jet Engine

The present section provides an overview of basic jet engine technology. Jet, or gas turbine, engines provide propulsion and power for aircraft. In the most common form, the jet engine generates the thrust that an airplane needs for sustained flight. Other architectural arrangements use the gas turbine engine predominantly for shaft power. Typical applications for these turboshaft engines include helicopters and turboprop aircraft. The present research is limited to gas turbine jet engines, which omits a few forms of jet engine such as the rocket engine and scramjet.

A jet engine, as the name implies, creates a thrust force to propel an aircraft by generating a high-speed jet of air. The jet propulsion principle is based on Newton's third law of motion; for every action there is an equal and opposite reaction. Thermodynamically, jet engines operate using the Brayton cycle. As shown in Figure 9, the basic turbojet engine

is comprised of an inlet, a compressor, a combustor, a turbine and a nozzle. The thrust developed by this engine all comes from the high velocity air exhausted out of the aft side of the engine.

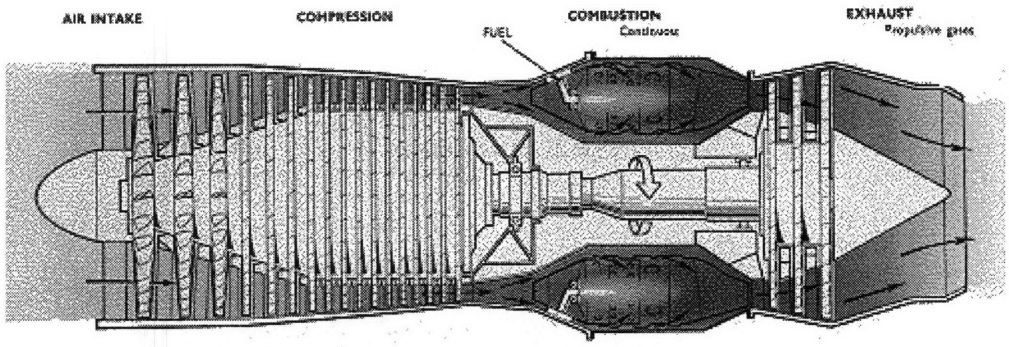


Figure 9. Cross-Section of a Turbojet Engine (Century of Flight, 2006)

Figure 10 below shows a modern, high bypass turbofan engine. This commercial engine powers the Boeing 777 aircraft (see Figure 11), delivering an incredible 94,000 lbs and 115,000 lbs of takeoff thrust for the GE90-94B and GE90-115B engines respectively.

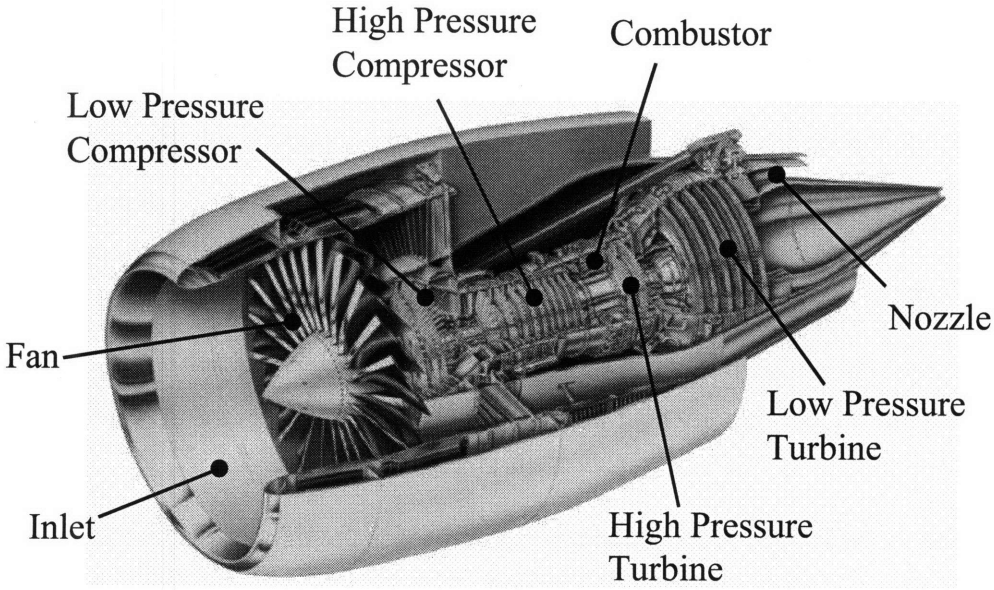


Figure 10. GE90 Turbofan Engine (Modified from GE, 2006)

The turbofan engine produces thrust in a slightly different manner as compared to the turbojet. In addition to producing thrust at the nozzle, the turbofan engine also exhausts a large amount of air from the fan to generate thrust. It is common for the fan to produce approximately 80% of the thrust while the nozzle only supplies roughly 20% (GE, 2006).

Figure 11. Boeing 777-200LR Aircraft (Boeing, 2006)

Each of the primary components of the turbofan jet engine is clearly illustrated in the cut-away picture in Figure 10. In the next few sections, I describe each component and its functionality in further detail.

Inlet. The inlet is the forward-most component on a jet engine and its function is to channel air from the ambient environment into the engine. The inlet also acts as a diffuser, slowing the air velocity as it travels aft to meet the fan or compressor.

Fan. The fan is the large rotating structure that provides the majority of the thrust generated by the engine. It compresses the air from the inlet, distributes a small amount to the compressor and then exhausts the bulk of the air out of the engine through a bypass duct. The ratio of airflow through the bypass duct versus the core is known as the bypass

ratio. The bypass ratio for the turbofan engine in Figure 10, for example, is 8.7 at takeoff. The fan has a larger diameter relative to other components of the engine so that it can increase the velocity of a large quantity of air by a small amount. This is more efficient than giving a small quantity of air a large increase in velocity as seen at the exhaust nozzle of the engine.

Compressor. The compressor increases the pressure of the air entering the engine and forces the air to flow through the rest of the engine. Large commercial engines today have compressors that increase the air pressure by a factor of 40 or more over the ambient pressure. While the compressor significantly increases the pressure and temperature of the air, the velocity stays relatively constant.

Combustor. In the combustor, fuel is injected as a fine mist, which mixes with some of the airflow. This mixture is then burned to increase the temperature and energy of the air. The amount of fuel that is burned determines the power that can be generated by the engine.

Turbine. The turbine turns the thermal energy and pressure of the air leaving the combustor into mechanical energy that can be used to power the fan or perform shaft work.

Nozzle. The nozzle receives the air from the turbine and channels it out the aft end of the engine. Additionally, as the air travels through the nozzle, its velocity increases.

In the case of the turbojet, the compressor and turbine are connected together and rotate on a single shaft. This is known as a single-spool engine. The turbofan engine is modularized differently with the compressor and turbine each being subdivided into two separate components. The fan, low-pressure compressor and low-pressure turbine rotate on a single shaft, while the high-pressure compressor and high-pressure turbine rotate on a second shaft. Therefore, the GE90 turbofan engine is said to have a two-spool engine architecture.

At this point, I have briefly described the primary modules in a jet engine. Now, some of the technical advantages and disadvantages of the jet engine, as compared to the piston engine, are mentioned to further ground the reader.

Advantages of Jet Engines over Piston Engines. As compared to piston and propeller aircraft, jet engines can fly at faster speeds and higher altitudes. They also have a higher power-to-weight ratio, which is critical in the weight sensitive aircraft business. For a given power rating, jet engines will take up less physical space. This reduced physical footprint improves aircraft performance because a decreased frontal area is associated with a drag reduction. St. Peter (2000) writes that gas turbine engines are easier to start and do not require long warm up periods like piston engines. Most importantly for this analysis, jet engines are much more reliable (Loftin, 2006). Tens of thousands of hours can be flown before a major overhaul is required.

Disadvantages of Jet Engines. Jet engines also have a few inherent disadvantages, most of which are anchored in cost. The largest one for the customer is that they are very expensive, and in some cases use more fuel at idle throttle points. In addition, they require a more constant speed setting and loading.

Metrics. Jet engines are governed by metrics that describe the performance, cost and reliability among other attributes. Performance metrics include the thrust to weight ratio of the engine, the maximum thrust produced and the fuel efficiency of the engine known as specific fuel consumption (SFC). Figure 12 (Maclin, 2003) shows that the thrust-to-weight ratio has steadily increased since the I-A, the first U.S. jet engine. In Figure 13, we see that the specific fuel consumption is decreased as new engines are developed. Both cases indicate higher levels of performance have been continuously achieved over time.

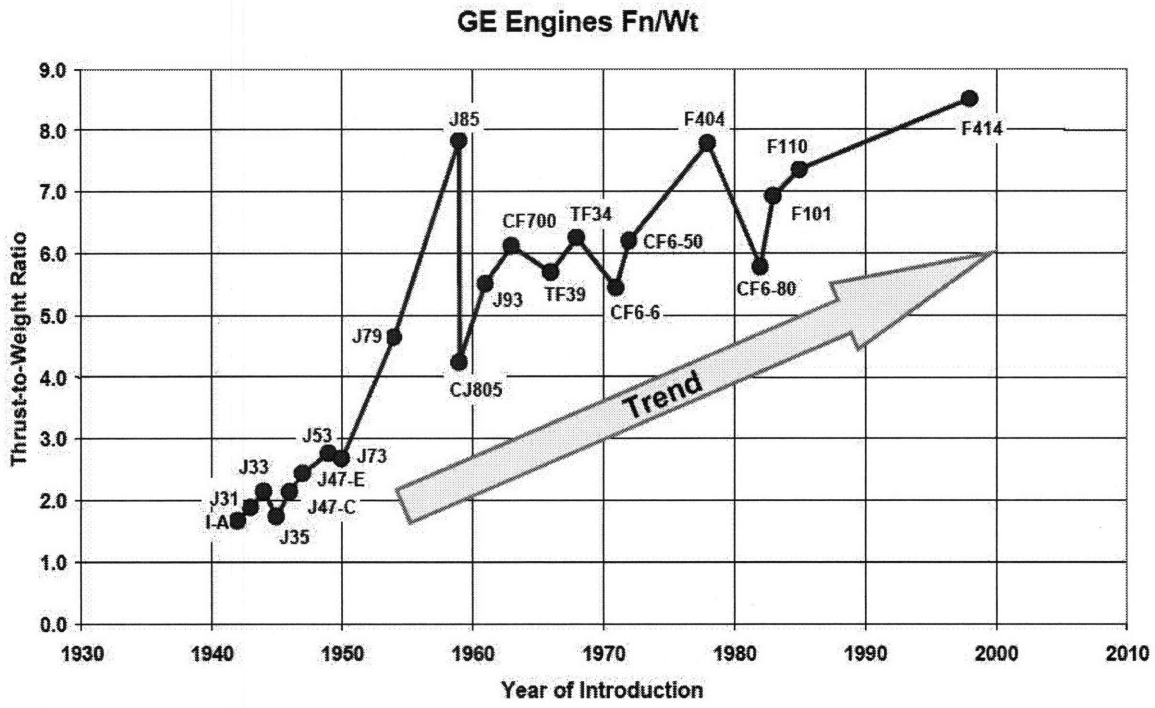


Figure 12. Thrust-to-Weight Ratio Trend for GE Engines

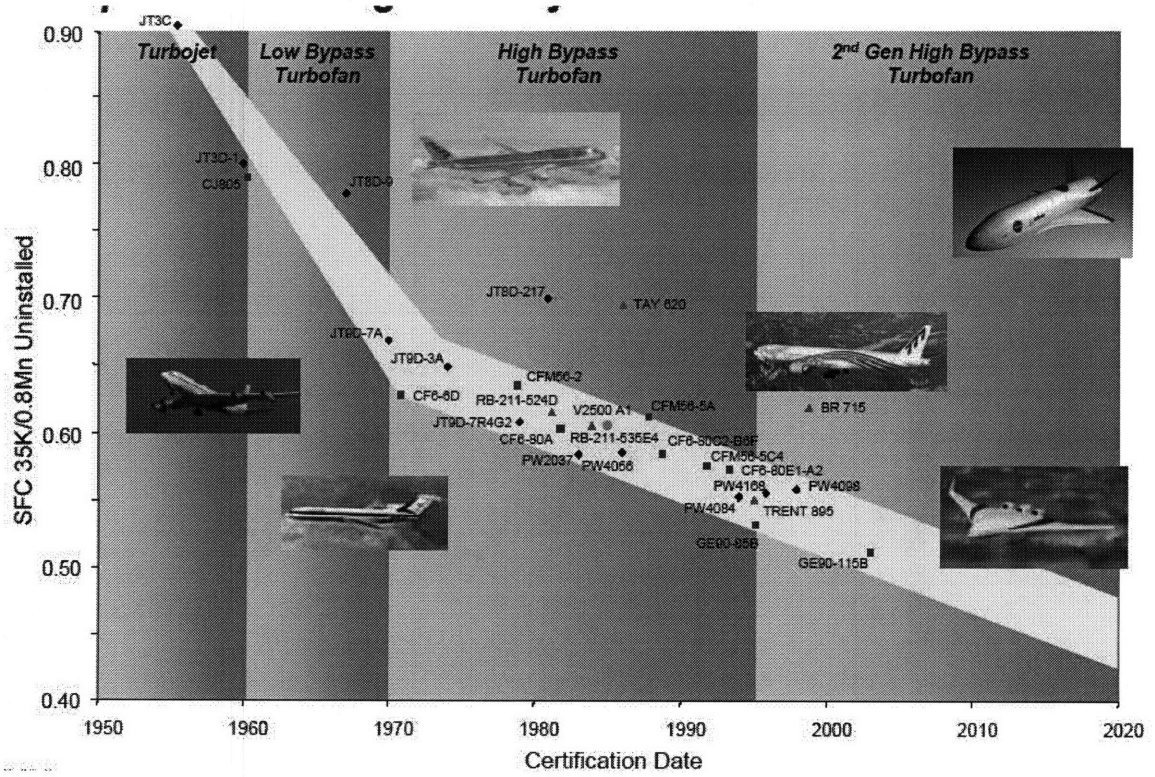


Figure 13. Specific Fuel Consumption Trend

Due to the criticality of safety in the aircraft industry, a number of jet engine metrics have also been created for reliability and robustness. Listed below are some of these metrics:

- IFSD (In-Flight Shutdowns)
- ATO (Aborted Takeoffs)
- LOTC (Loss of Thrust Control)
- Dispatch reliability
- Unscheduled Engine Removals (UER)
- Time-on-wing
- Maintenance cost

Figure 2 in the first chapter presented the IFSD trend for jet engines over the past 50 years. This data demonstrated exponentially increasing reliability. The same trend is observed in data taken from the Office of Technology Assessment (1988) for JT8D engines. As illustrated in Figure 3 at the beginning of this work, the time or life on wing is also exponentially improving.

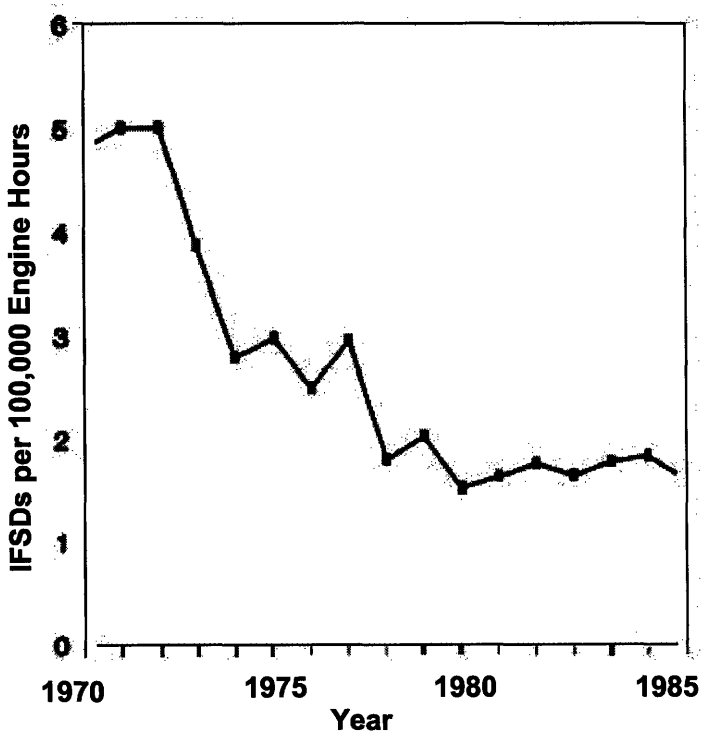


Figure 14. IFSD Rate for B727 with JT8D Engines

It can be seen from this list of metrics that the performance, reliability, robustness and safety of jet engines has consistently improved since the 1940s. In section 5.5, we will review jet engine history to identify architecture changes that have contributed to these improvements. As a first step, a framework for architectural innovation is covered in section 5.3.

5.3 Architectural Innovation

Before we assess the evolution of the jet engine system, we must clarify the attributes of change for which we are looking. Four types of innovation have been observed in technological systems as depicted in Figure 15 (Henderson and Clark, 1990).

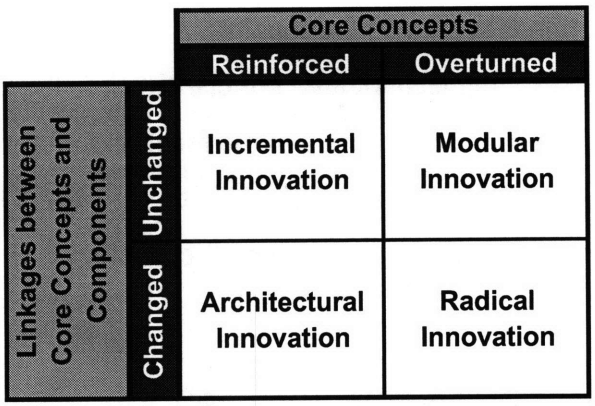


Figure 15. Types of Innovation

The jet engine is a radical innovation as compared to the piston engine, because the core concepts and linkages were both changed – rotary motion replaced reciprocal motion, and thrust was provided by a fast moving jet of air instead of the turning of a propeller.

Modular innovation involves concept changes but not necessarily linkage or integration changes. Henderson uses the example of analog phones being replaced with digital phones, because the product architecture does not change, just the concept for the dialing device. In the jet engine industry, a modular innovation example is the use of a nuclear powered combustor on GE's X211 engine. Instead of fuel being combusted, the air was heated by a nuclear reaction and the other components were kept primarily the same.

Incremental innovation involves gradual improvement in the components of a system, with no concept or linkage changes. Incremental improvements in the efficiency of the compressor and turbine modules have continuously improved the performance of jet engines without any changes to the overall architecture.

Architectural innovation is the type of change in which we are interested as it is the result of integration changes amongst the components. This type of innovation was originally coined "generational innovation" by Henderson (1988), and then later re-termed "architectural innovation" by Henderson and Clark (1990) as a better descriptor.

An architectural innovation may be a new technology at the system level, but each of the subcomponents uses pre-existing technology. For clarification, the top-level concept can change with architectural innovation in terms of the function to form mapping, but the basic component technology is primarily prior art. In this respect, a large component of architectural innovation deals with the integration of technology.

Figure 16. Architectural Innovation in Lawn Tractors

The pictures in Figure 16 are taken from Toro (2006), and they illustrate architectural innovation in the riding mower or lawn tractor industry. The front wheels swivel so the mower has a "zero-turn radius", which increases its utility and robustness in lawn cutting. The technology for both riding lawn mowers and swiveling wheels has been around for quite some time, however, this represents a unique integration of the technologies to

create a new, more effective system. A few other examples of architectural innovation are shown in Table 5.

Table 5. Examples of Architectural Innovation

Sliding extension ladder	Articulating ladder
Business jets	Very light jets
5.25" disk drive	3.5" disk drive
Low bypass turbofan	High bypass turbofan
Hinged car door	Sliding car door
Riding lawn mower	Zero-turn radius mower
Swinging garage door	Hinged garage door
Internal combustion engine (ICE) powered automobile	Hybrid (ICE and electric motor) powered automobile
Contact photolithography	Proximity photolithography

By studying the photolithographic industry, Henderson determined that companies frequently do not understand the significant effect architectural innovation has on competition in industries. She also presents proximity aligner data that shows tradeoffs can be made with architectures, one example being a slight reduction in minimum feature size capability for greater yield or robustness in the production process. In the jet engine domain, we seek architectural changes or innovations that increase the robustness of the system, ideally with no decrease in performance.

5.4 Dominant Designs

Throughout the evolution of the jet engine, certain system architectures have established themselves as predominant in the marketplace. Utterback (1996) has referred to these as dominant designs, although they frequently represent dominant architectures. A summary of the jet engine dominant designs is located below in Table 6. The table lists the dominant design in the left-most column accompanied by a brief description and a

category of aircraft in which it is typically utilized. The last column identifies an example aircraft application followed by the engine. As mentioned in the introduction, this research focuses on air-breathing, gas turbine, aircraft engines. Air-breathing implies that the engine uses atmospheric air during the combustion process to oxidize the fuel. The oxidizer does not need to be carried on board the engine or aircraft. Gas turbine indicates that energy is extracted from the air after the combustor by turbine blades.

Table 6. Dominant Designs for Jet Engines

Dominant Design	*Description*	*Typical Use*	*Example Application*
Turbojet	Single Spool Jet	Military Aircraft	North American F-86H with GE J73
Low bypass Turbofan	Low BPR, Two-spool Jet	Military Aircraft	General Dynamics F-16 with GE F110
Medium bypass Turbofan	Medium BPR, Two-spool Jet	Commercial Aircraft, Military Aircraft	Cessna CJ1 with WI FJ44-1
High bypass Turbofan	High BPR, Two-spool Jet	Commercial Aircraft, Military Aircraft	Boeing 777 with GE GE90
Turboshaft	Shaft Power	Helicopters	Sikorsky S-92 with GE CT7
Turboprop	Shaft Power to Propeller	Commercial Regional Aircraft	Lockheed Martin C-130 with All T56
Propfan	Unshrouded fan	Transport, Cargo Aircraft	Antonov An-70 with Pr D-27

The research focus excludes rocket engines, for example, because although they are jet engines, they are not air-breathing (see Table 7). Scramjets are air-breathing, but would also be excluded because they do not have turbomachinery and have not yet established themselves in the marketplace.

Table 7. Sample Architectures that Are Excluded

Type	Description	Typical Use	Example Application
Pulse jet	Pulsed Combustion	Military Missiles	V-1 "Buzz bomb"
Pulse Detonation Engine	Pulsed Detonation	Emerging	N/A
Ramjet	Jet with no turbomachinery	Missiles, High-Speed Military Aircraft	Lockheed Martin SR-71 with P&W J58*
Scramjet	Supersonic Combustion Ramjet	Emerging	N/A
Rocket Engine	Jet that is not air-breathing	Spacecraft, Missiles	Space Shuttle
Tip Jet	Jets at tip of rotor blades	Helicopter	Hiller Hornet

Much experimentation and competition occurs over time in an industry, and the survivors represent the dominant designs. The dominant designs can change periodically during the evolution and additionally new designs and system architectures can emerge in parallel. The Wright brothers' first powered flight utilized a 12 hp piston engine. The piston engine became the dominant architecture for many years for nearly all aircraft. The speed of piston-powered aircraft increased over time until reaching an upper limit of approximately 450 mph (Gunston, 1997). In the 1940s, the jet engine began to supplant the piston engine due to its superior capability and reliability (NASA, 2006).

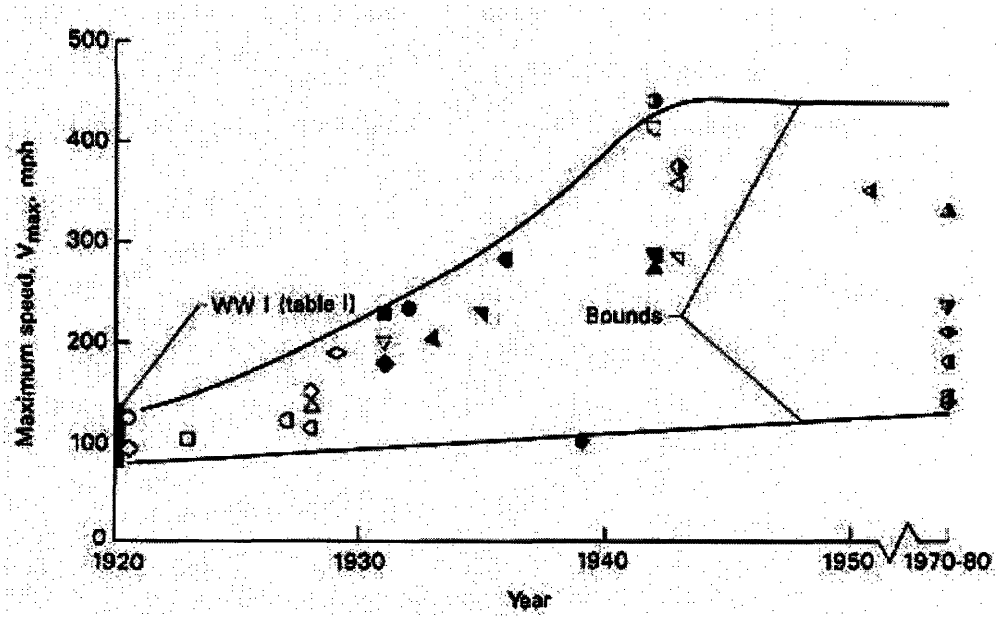


Figure 17. Trends in Maximum Speed of Propeller-Driven Aircraft (Donlan, 1954)

5.5 Architectural Evolution

Up to this point, the dominant engine architectures that have developed in jet engine history have been reviewed. We now look at the architectural evolution of the engine and discuss some of the major changes that have occurred in the years since the jet engine's invention. The intent here is to describe architecture changes to the engine system, although it is recognized that some of these innovations might also be considered technology changes. As mentioned above in Section 5.3, architecture changes reconfigure existing technologies. In some cases, this reconfiguration creates a new system level technology. Over 250 jet engines were investigated in the course of this research to identify architecture changes that improved robustness. The engines cover the period from 1937 to 2007. These engines are listed by engine certification date in Appendix C for the reader's reference.

Engine Configuration

The first jet engines, co-invented independently by Frank Whittle and Hans von Ohain, consisted of a single-spool architecture with the compressor and turbine connected to one rotating shaft (see Figure 9). These turbojet engines were fairly integral structures with fixed geometry and centrifugal compressors. In von Ohain's case, a single axial inducer was added to the centrifugal compressor for his He S 3B engine. The He S 3B was used in the Heinkel 178 aircraft and in 1939 it became the first plane to fly solely under jet power. Between test engines and test flights, it quickly became evident that the early turbojet engines were not very robust or efficient, and this was the impetus for many design and architectural changes in the following years.

Many of the initial changes were made to enhance the performance and stability of the compressor. As one example, axial compressors began to supplant centrifugal compressors due to their higher efficiency. Because the early jet engines were optimized for a single design point, they were subject to compressor stalls at low power, at startup and during transients. The compressor stall phenomena will be described in more detail in section 6.2. To improve the robustness of the engine, two disparate solutions were offered by industry in the 1950s. The first involved adding variable geometry to the

engine. Inlet guide vanes and stator vanes were made to rotate so that the incoming airflow could be tuned to the compressor rotor. The second solution worked in a nearly opposite manner by allowing the compressor rotor to adjust to the incoming airflow. This was achieved with a two-spool architecture that split both the compressor and turbine into separate low and high-pressure modules. The low-pressure compressor and low-pressure turbine were connected to one shaft, while the high-pressure compressor and high-pressure turbine were connected to another independent shaft. In this way, the coupling between front and aft portions of the compressor was substantially reduced. This modularization improved the robustness and stability of the compressor to changing inlet conditions and improved engine performance at off-design conditions. The benefits of two-spool engines are further described in chapter six. In the end, both variable geometry and two-spool architecture were committed to the turbojet dominant design.

An important consequence of the two-spool architecture is that it opened the door for the next dominant design - the turbofan engine. Recall that turbofan engines send the majority of the air through a fan bypass duct and a smaller amount of air through the core of the engine (see Figure 10). This configuration increases the thrust capability of the engine or lowers the fuel burn for a given thrust rating. Engineers were well aware of the performance benefits of increasing the bypass ratio, and in the following years worked to increase the length of the fan blades. While this was primarily an architectural change, it required numerous innovations to be realized. Extending the fan blades increased their mass, which at high rotational speeds could not be adequately supported by the fan shaft. In addition, the blades were not initially robust and flutter was encountered due to varying inlet conditions of the air. These strength and stability problems were solved with wide chord, hollow and composite fan blades, but a more detailed discussion of the solution is deferred until section 6.2. In the ensuing years after the 1950s, the bypass ratio steadily increased through low, medium and high bypass dominant designs. These architectural changes contributed considerably to the performance increase of the engine as shown in the thrust specific fuel consumption plot (TSFC) in Figure 18.

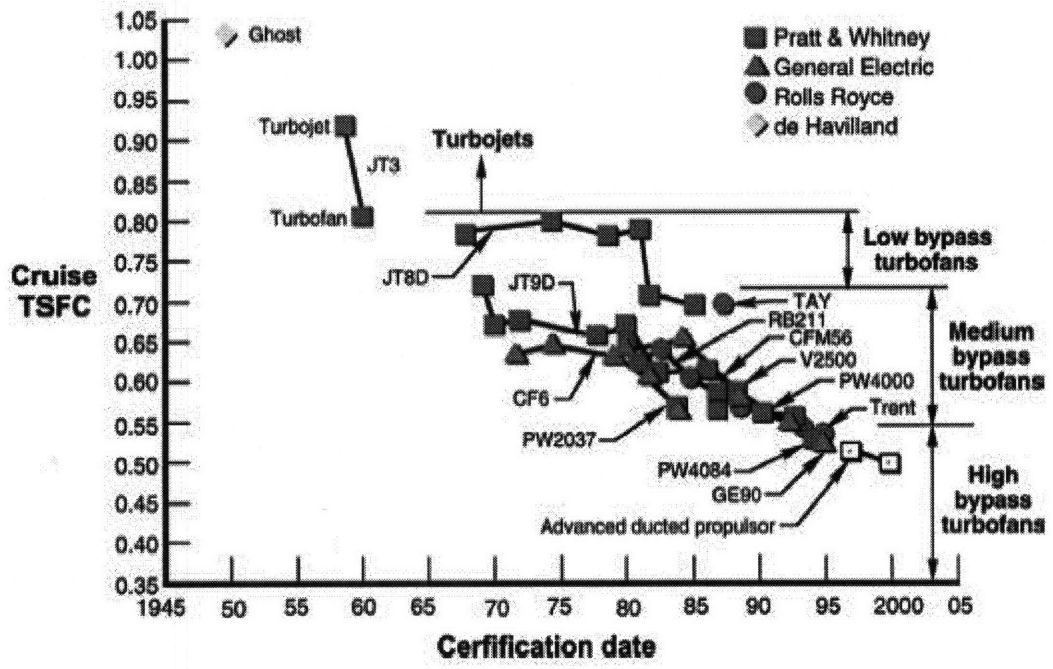


Figure 18. Trend in Thrust Specific Fuel Consumption (Koff, 2004)

Some later architecture changes to the jet engine system have been conceptually simple, but have delivered significant benefits. One example involves the two spools on a turbofan engine, which have historically spun in the same direction. Using the physics of the airflow in the turbine, the architecture was changed on some engine models so the two spools rotated in opposite directions. Gunston (1997) remarks that counter-rotating spools improved the performance and robustness so that a few turbine blades could be removed and stators could be eliminated between spools. An additional advantage is that the gyroscopic forces and engine torque are reduced.

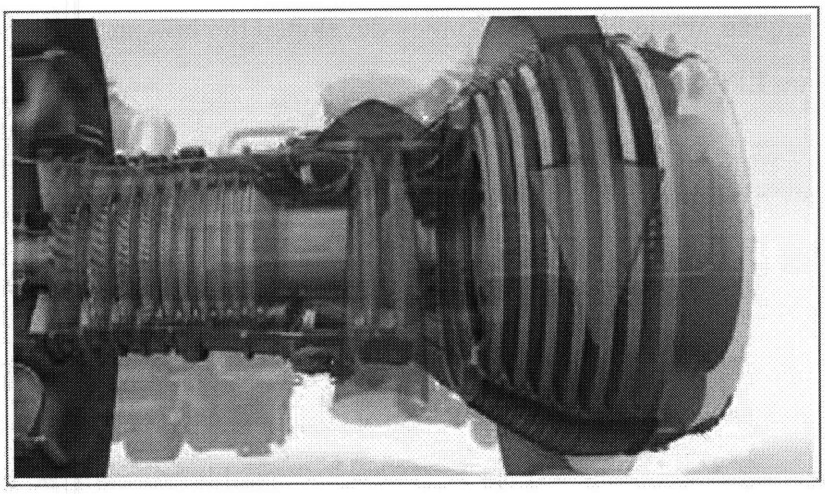


Figure 19. GEnx Engine with Counter-rotating Spools (GE, 2006)

While the early jet engines were more integral structures, subsequent iterations became more modular to facilitate design and the "ilities". Modern jet engines also appear to follow Rubinstein's principle of architecture decomposition, which requires a maximum of 7 ± 2 modules to limit the perceived complexity (Maier and Rechtin, 2002). The seven modules that comprise a typical turbofan engine are the fan, the low-pressure compressor (LPC), the high-pressure compressor (HPC), the combustor, the high-pressure turbine (HPT), the low-pressure turbine (LPT) and the nozzle. If one counts the inlet, then a total of eight modules are present. In combination, the HPC, combustor and HPT form the core of the engine, also known as the gas generator. The core architecture of the engine has become a building block for propulsion systems. A low-speed shaft is added to the core to tailor the engine for a specific application. For example, engine manufacturers have used a "common core" to create a military engine and a civil engine. In another case, the core of a turbofan engine was used to develop a marine turboshaft engine. The engine core architecture has also been scaled to create higher and lower thrust engine applications. Many modern engine projects are undertaken jointly by two or more competing companies, and in these situations the engine core concept aids in the design process. The GP7200 turbofan engine was designed and built by the Engine Alliance, which is a joint venture of two primary competitors – GE and Pratt and Whitney. This type of venture requires the technical information exchange between the companies to be absolutely minimized for proprietary and competitive reasons. In this case, GE designed

the core of the engine, while Pratt and Whitney developed the low-pressure spool including the fan, LPC and LPT.

Secondary Flow Systems

To meet growing performance and robustness demands, engine designers developed and evolved secondary flow systems for jet engines. These systems typically bleed air from the compressor or the fan to cool components, "shrink" casings for smaller clearances, purge cavities and drive the environmental control system (ECS) for the aircraft. Figure 20 is a cross-section showing the secondary flow system architecture for the GE90 turbofan engine.

Bleed valves and ducts were added early on to divert some of the airflow from the compressor to assist in engine starting and low-speed operation. At these conditions, the airflow can choke or be restricted at the aft end of the compressor because the air has been insufficiently compressed. The bleed ducts channel some of the air out of the compressor so that the rest of the flow can pass unrestricted through the aft end of the compressor.

From a performance standpoint it is desirable to maximize the temperature of the air in the combustor. However, this creates many challenges for the downstream hardware. The temperatures on a modern jet engine are well above the melting point of the blades and nozzles in the high-pressure turbine (HPT). To increase the efficiency of the engine and maintain robust hardware, cooling air is bled from the back of the compressor and delivered to the HPT blades and nozzles. Likewise, cooling air is bled from the compressor and supplied to the low-pressure turbine (LPT) to purge cavities and prevent the ingestion of hot flow path air.

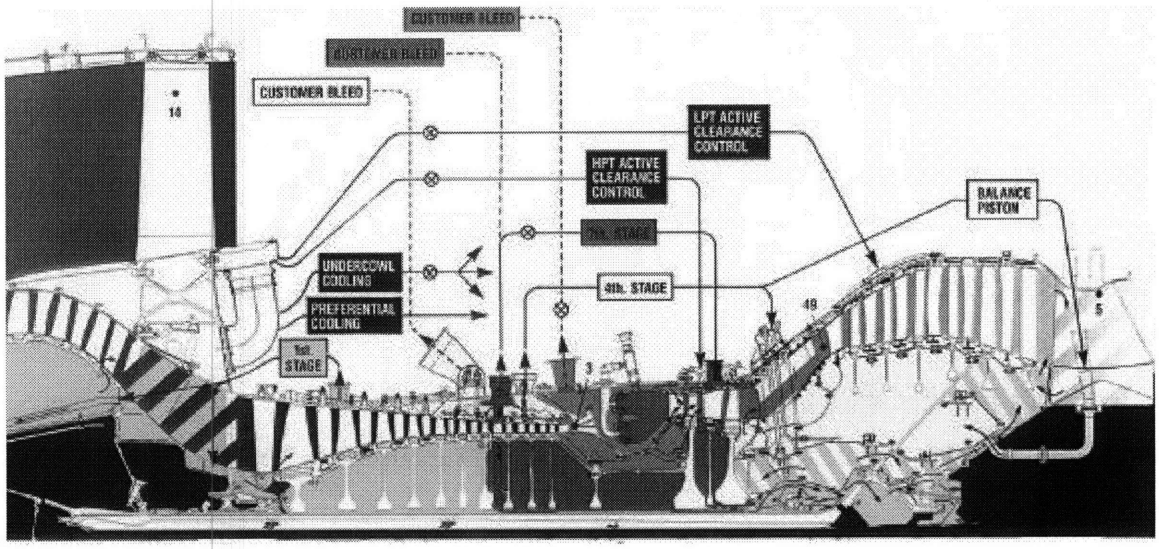


Figure 20. Secondary Flow Systems for the GE90-94B Engine (GE, 2006)

The "balance piston" is another example of the robust functionality provided by the secondary flow system. The balance piston is located on the right hand side of Figure 20. In this case, air is bled from the fourth stage of the compressor and supplied to a cavity near the aft end of the engine for pressurization purposes. The pressure creates a force on the cavity wall that partially counteracts the other forces on the low-speed shaft. This functionality reduces the lateral force on the bearings and as a result extends the bearing life.

The intent here was to show a sampling of the flow subsystems that have been added to the engine and to illustrate how they were integrated into the architecture. Secondary flow systems serve a number of other functions such as providing hot air for anti-ice systems, customer bleed air for the aircraft, and bleed air to control turbine blade tip clearances. In many cases multiple pipes are used to evenly distribute the flow and also to create redundancy for safety purposes. The evolution of the secondary flow system architecture appears to follow the TRIZ ideality principle. Ideality is the first and foremost TRIZ law, stating that all technological systems move toward increasing ideality, simplicity and benefit-to-cost ratio. Ideality guides the architect to use resources within the system and its context to solve technical conflicts instead of adding new

systems. This is seen in the jet engine evolution with the compressor and fan duct used as bleed air sources to resolve conflicts and enhance robustness.

Engine Control Systems

As depicted in Figure 21, the number of control functions in engine control systems has monotonically increased from the 1960s to present day. Requirements for improved performance, robustness and safety have driven engine manufacturers to continually create controls systems with increased functionality. In this way, the engine system can adjust and optimize engine settings for an ever-increasing number of conditions and parameters.

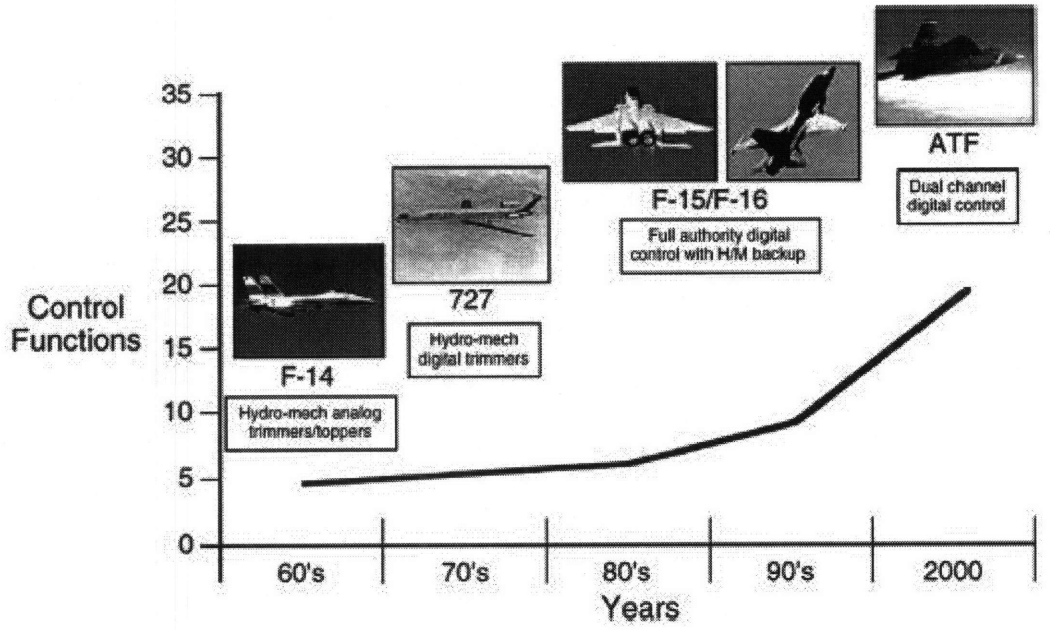


Figure 21. Number of Engine Control Functions (Koff, 2004)

The control for the first U.S. jet engine was a hydro-mechanical governor to meter fuel flow. Subsequent years brought controls for afterburners, ignition/shutdown systems and anti-ice systems. With the advent of variable compressor geometry, hydro-mechanical controls were put in place to adjust variable inlet guide vanes (VIGVs) and variable stator vanes (VSVs). As the number of controls increased, the hydro-mechanical systems soon became heavy, costly and impractical. Analog electronic control units (ECUs) were

introduced to fill this need, although they were initially used only to trim settings. ECUs were eventually given *full-authority* to control engine functions throughout the engine operating range with a backup hydro-mechanical system. The next step in the evolution was to full-authority digital engine controls (FADEC), which presented much more control capability and enabled simplified modification through software. FADEC included two digital control channels with the second channel being redundant. The FADEC systems had the ability to use advanced software models to automatically adjust engine settings, which improved performance and robustness while simplifying the complexity for the pilot. In the pursuit of enhanced control system capability, engine designers have stepped through multiple domains to select those most conducive to robustness and performance. Transitions through mechanical, electronic and informational control domains have brought greater degrees of freedom, increased sophistication and higher complexity.

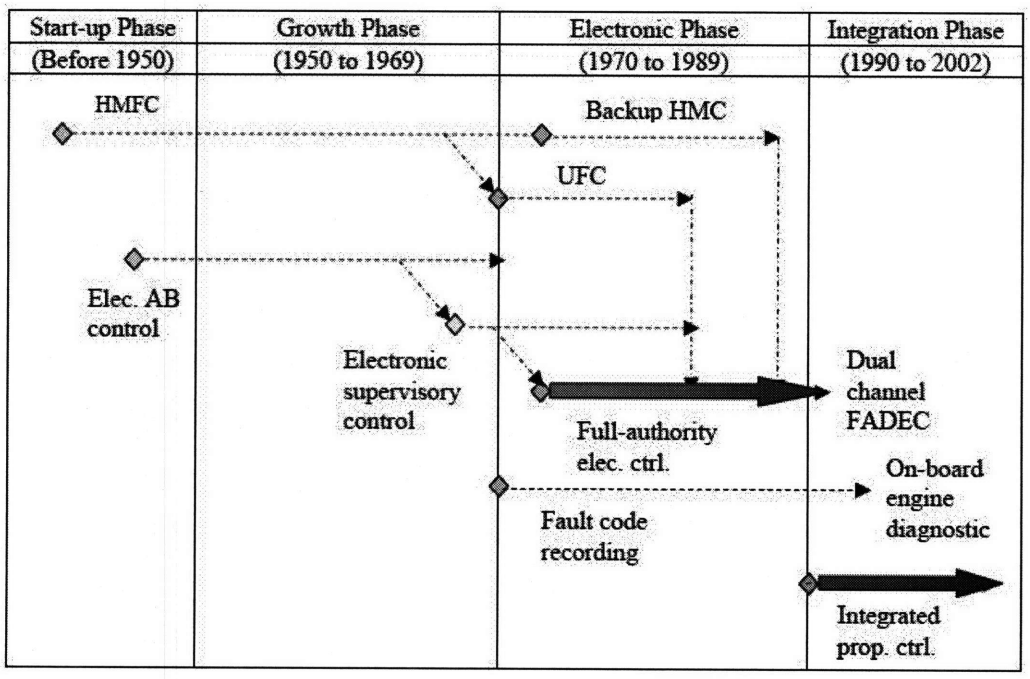


Figure 22. Evolution of Engine Controls (Jaw and Garg, 2005)

The number of sensors in a jet engine has increased in a similar manner to engine controls (see Figure 38). Starting with a few critical applications such as fuel flow rate and rotor speed, sensors have progressively been incorporated to monitor additional

aspects of engine operation. Modern engines measure the temperature of the engine at multiple axial locations. The sensors provide feedback to the engine control unit to enhance the stability and autonomy of the engine system. For example, engine inlet sensors are used to detect and adapt to changing environmental conditions, thereby improving the stability and performance of the compressor. Other sensors have been added to improve safety; temperature harnesses are placed around the outside of the engine to detect fires and activate a halon fire extinguishing system if needed. Redundant sensors are often used for safety critical functions.

Since the first turbojet-powered aircraft took flight, engine control and sensor systems have come a long way. In the beginning, pilots would monitor the gauges in the cockpit and complete the feedback loop by making adjustments as needed. The electronic systems that developed in the following decades automated these actions. Furthermore, the digital electronic control systems could perform health diagnostics on the engine. Engine data could be downloaded after each flight to search for faults or to identify trends. More recently, remote monitoring has allowed engine health data to be monitored in real-time. By reviewing engine performance trends in real-time, potential issues can be caught in the early stages, eliminating problems before they start.

Components

Since the beginning, the complexity of jet engines has increased on average due to the incorporation of additional functionality for improved performance and robustness. One indicator of the increasing complexity trend is the total part count in the engine as shown in Figure 23. Recall in section 3.3 several simplified measures for quantifying system complexity based on part count were described from work by Boothroyd and Dewhurst (1987) and Crawley (2006). The actual part count for different engine lines will be stratified in many cases depending on the size and thrust level of the engines under consideration. For example, low thrust engines utilized in business jets may have fewer parts than high thrust engines employed in large commercial aircraft. However, taken individually each engine class trends in general toward a higher part count over time. It is interesting to note that in Figure 23, the part count takes a downward step in the mid

1990s. More data is required to see if this a unique point, or if it is the beginning of a new trend. One possibility is that the complexity is being transferred from the mechanical domain to electrical and informational domains.

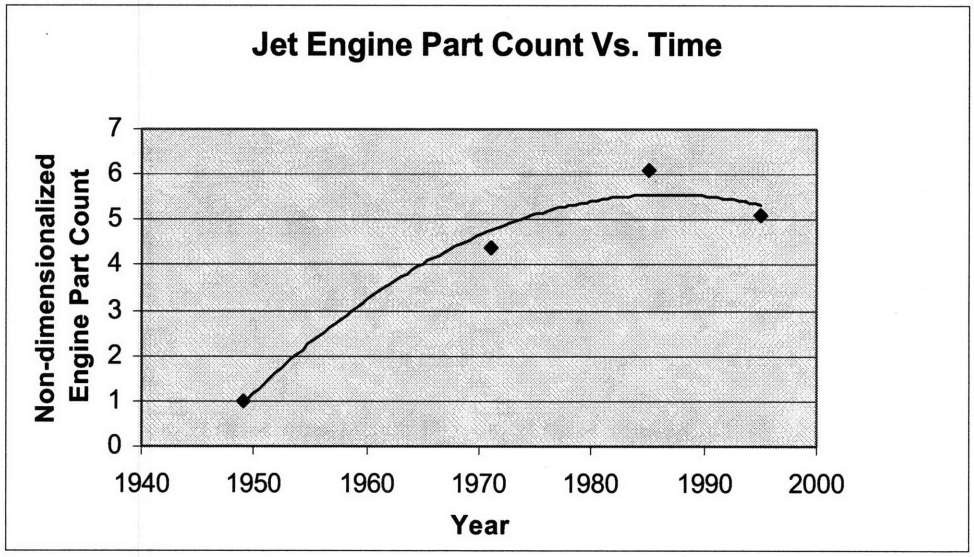


Figure 23. Jet Engine Total Part Count

In spite of the increasing system-level complexity, aspects of simplicity have emerged and been strengthened throughout the evolution of the jet engine. As one example, while the total part count for the engine has historically increased, the components and subsystems of the engine have witnessed a decrease in part count. Figure 24 shows the total fan blade count for large turbofan engines over the past four decades. The year in this plot represents the certification date for the engine. Since the 1970s, the number of fan blades has been reduced by more than 50%. This decrease in part count improves the reliability of the engine system and eliminates some failure modes.

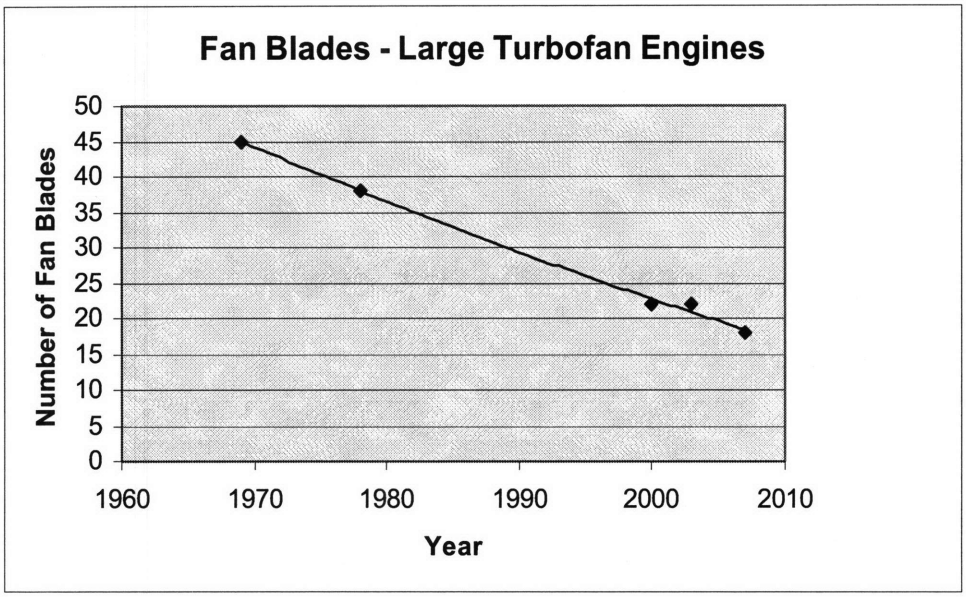


Figure 24. Number of Fan Blades for Large Turbofan Engines[1]

The compressor has also experienced simplification and part count reduction over time. Each axial stage of the compressor is comprised of numerous blades that slide onto a metal disk. The total assembly can have well over a hundred parts. However, a "blisk" architecture was introduced on some engines that integrates the blades and disk into a single piece, greatly reducing the part count. In addition to the per stage part count, the number of axial stages in the compressor has historically decreased. This is typically because new blade technologies are integrated into the system that can achieve the same pressure rise over the compressor with fewer parts. Figure 25 illustrates the number of axial compressor stages in a few large turbofan engines versus the engine certification date. In this case, the number of compressor stages has decreased over the past 40 years while the pressure rise has actually increased. Although the ramjet is not within the scope of this thesis, it offers an excellent data point for part count reduction and as such is mentioned here. The ramjet eliminates all of the compressor stages in the engine and still achieves the required pressure ratio. This is possible because the system architecture of the ramjet uses a shockwave at the inlet to decelerate and compress the air. The primary

[1] The TF39 engine is equipped with a 1.5 stage fan. The blade count for this plot only represents the full fan stage.

limitation of the ramjet is that it cannot generate thrust at a standstill; it must already be traveling at high speed to accomplish compression.

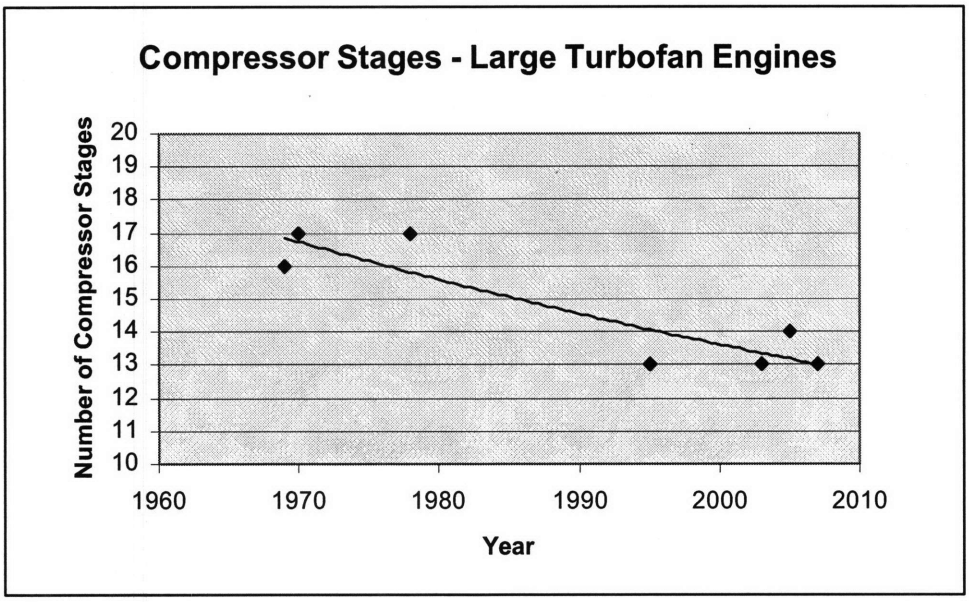


Figure 25. Number of Compressor Stages for Large Turbofan Engines[2]

As shown in Figure 26, the number of unique part numbers per engine is gradually decreasing. Simultaneously, the total number of parts is increasing, which indicates the ratio of parts to part numbers is growing. This suggests that standardization is increasing within the engine. Standardization is economically beneficial for the engine manufacturers because it means fewer parts have to be tracked and held in inventory. Additionally, standardization reduces mistakes in assembly and use of the engine.

[2] Figure 25 shows the total number of compressor stages for the engine, which includes both the low-pressure and high-pressure compressor modules.

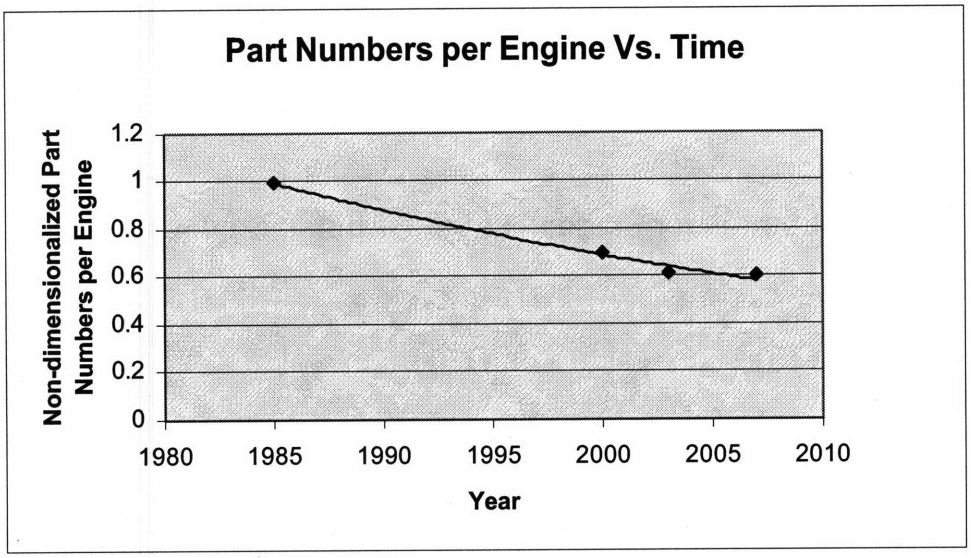


Figure 26. Unique Part Numbers per Engine

Trends and Principles

A summary and timeline of the architectural evolution of the jet engine is presented in Figure 27. Looking at the progression of changes to the engine configuration, the controls systems, the secondary flows and the components, several trends and principles became evident.

The first trend might be described as a tension between complexity and simplicity. The overall complexity of jet engines was shown to be continually increasing based on the number of parts, control functions, sensors and flow systems. However, on a component basis simplification has occurred through part consolidation and part elimination. Simplification is further enhanced because of the increasing standardization of parts. Fewer part numbers has reduced costs and eliminated some error modes in assembly, maintenance and use of the engine system. While, the complexity of the engine has increased, a simplicity principle has been followed to keep the actual complexity of the system as close as possible to the essential complexity.

As mentioned above the study of control systems and sensors showed an upward trend in complexity. Some of this complexity also came in the form of redundancy, with extra

sensors and systems put in place to improve the reliability of critical functions. However, the simplicity principle is also apparent in this evolution. The level of automation and autonomy in the control systems has dramatically increased. By incorporating feedback from numerous sensors, the control system can optimize the performance of the engine and adapt to changing environmental conditions. This minimizes the workload or apparent complexity for the pilot, thus simplifying operation. Furthermore, the control systems improved the robustness of the engine system by incorporating stability. For example, the compressor stability was enhanced by sensing inlet conditions and controlling variable geometry. From the control and sensor systems, three principles were observed: redundancy, feedback and autonomy.

Modularity has increased in the manufacture, assembly and design of jet engines, especially over the past 35 years. In some cases, these changes have been the catalyst for new dominant architectures to emerge. The two-spool turbojet helped bring about the turbofan engine, the turboshaft engine and the turboprop. The scalability and modularity of the core also supported the emergence of new dominant architectures, and allowed new market requirements to be addressed. The growing independence of the modules and functions has even allowed competitors to jointly develop engines. In other scenarios, the modularity of the structure aided in containment of error and failure conditions. For example, the nacelle that surrounds the outside of the engine is segmented into fireproof zones. These fireproof zones prevent the propagation of flames to other regions of the engine. As another example, the fan module is designed to contain a fan blade in a failure mode where the blade is released from its mounting.

Nine principles were identified to improve type I and II robustness in jet engines based on architectural changes. These are stability, modularity, feedback, standardization, redundancy, simplicity, independence, autonomy and scalability. In chapter six, each of these principles is further described, and the aspects of robustness improved by the principle are highlighted.

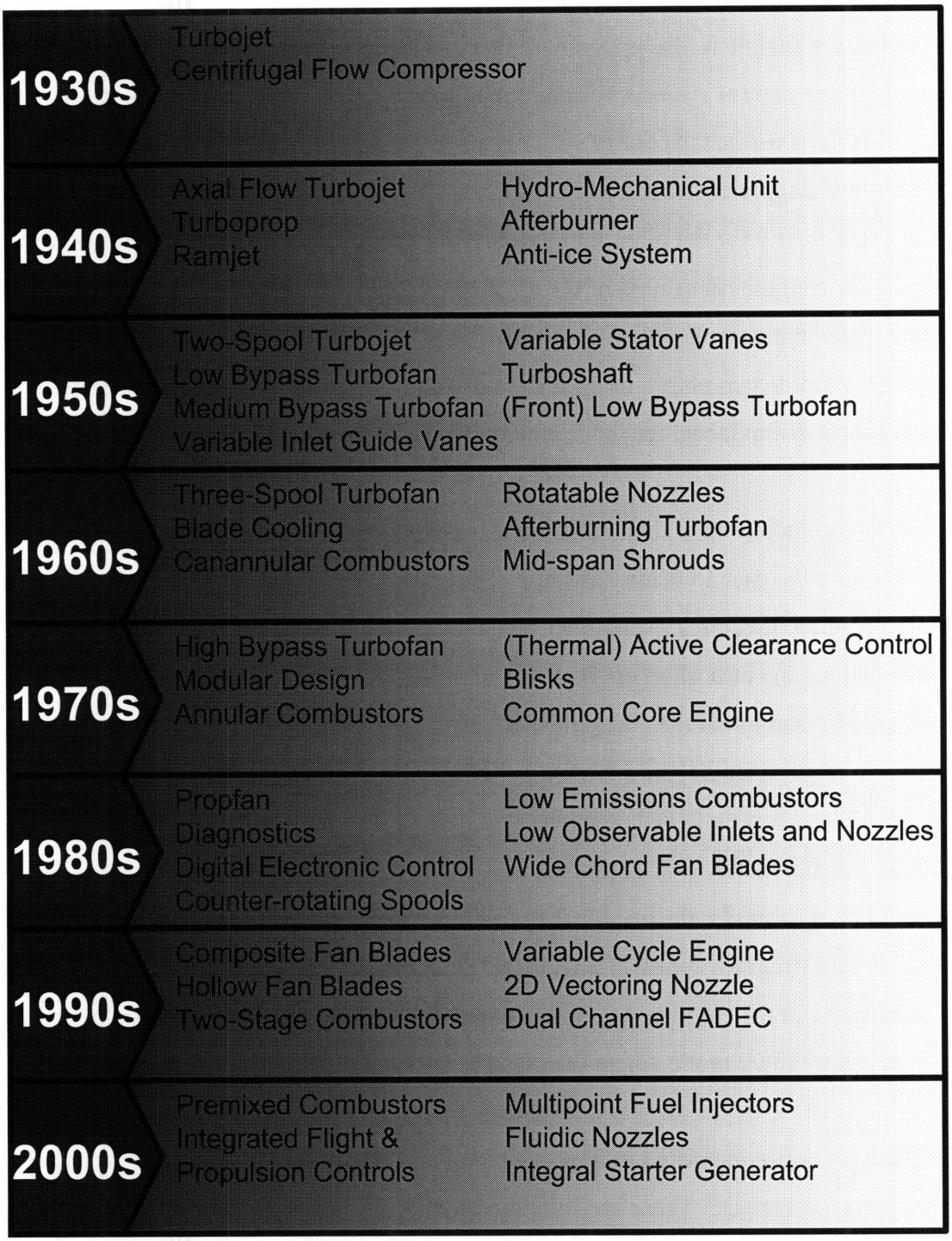


Figure 27. Architectural Evolution of Jet Engine

5.5 Summary

This chapter covered the technical overview of the jet engine and identified the specific engine types retained in the scope of this study. Architectural innovation was introduced

as a tool to frame the architectural changes that were pertinent to the research. Dominant designs have been established over the years, and these were summarized in section 5.4. The major architectural changes and innovations experienced during the history of jet engines were then placed in a timeline. Based on the analysis, trends and principles were identified that facilitated robustness during the evolution of jet engine systems. These principles will be described in greater detail in chapter six.

The trends showed that performance and robustness have been steadily increasing for jet engines. These benefits were realized through new technologies, new architectures and new functionality. Commensurate with these benefits came increases in the essential complexity of the engine system.

6. Principles of Robust System Architecture

"One insight is worth a thousand analyses."
- Charles W. Sooter

6.1 Introduction to Principles of Robust System Architecture

Tracing the evolution of the jet engine led to many possible principles of architectural robustness. To make our robustness framework more tractable, the list was narrowed down using several approaches. The first approach involved combining several closely related principles into a single principle category. For example, both redundancy and auxiliary functionality were observed to increase robustness in jet engines, but both principles fundamentally operate in the same manner – to add functionality to the system in excess of what is needed for nominal operation. As a result, they are both merged into the redundancy principle.

The second approach attempted to create hierarchies of principles to identify those at the "top-level", or those with the greatest opportunity for impact. One example hierarchy observed included modularity, segmentation, containment and encapsulation principles. Containment and encapsulation refer to the ability of a system to stop error propagation and shield elements from external noise. Segmentation suggests that a system be subdivided so that it can better react to noise factors. Modularity, however, includes all of the attributes of the other three principles, as it focuses on dividing the system into high cohesion modules with low coupling.

The remaining principles were reviewed in more detail and it became evident that two primary categories existed:

1. Principles of Robust System Architecture
2. Principles of Robust System Architecting

The first category, principles of robust system architecture, focuses on the product architecture. The second category, principles of robust system architecting, concentrates on the architecting process.

In the present chapter, I discuss the principles of robust system architecture, which are listed in Table 8. Principles of robust system architecting are reserved for future work where they can be more thoroughly developed and understood. A cursory assessment indicates that ideality, domain shift, common language, knowledge base, physics and flow, benchmarking and real options might be a few principles of robust architecting.

Many of these principles, or some attributes of the principles, have been discussed to some extent in the research literature (Altshuller, 1984; Suh, 1990; Maier and Rechtin, 2002; Fricke and Schulz, 2005; Frey and Jugulum, 2006; Sterman, 2000; Fey and Rivin, 2005; Crawley et al., 2004). However, the treatment of these principles and attributes in the literature often did not concentrate on robustness, which is precisely our intent – to generate a robustness framework for architecting complex systems.

Table 8. Robust System Architecture Principles

Principles of Robust System Architecture
Stability
Modularity
Feedback
Standardization
Redundancy
Independence
Autonomy
Simplicity
Scalability

In the next sections, an overview and definition of each principle of robust system architecture is given, and an example from the jet engine industry is used to illustrate the principle in action. Additionally, I classify the aspects of robustness that are improved by the principle utilizing the P-diagram framework. The classic elements of the P-diagram include the input signal, control factors, noise factors and system response. We add to

this error states as others have done with the P-diagram (Quality Portal, 2005). Finally, a "system context change" category is added, to cover type II robustness. A system context change would be a change in customer needs and uses, the appearance of emergent behavior and evolution of the embedding system. These are changes that reflect uncertainty in a system's control factors. At the end of the chapter, the principles are summarized again and the various aspects of robustness that are improved are highlighted.

6.2 Stability

The stability principle refers to the ability of a system to return to its normal operational state, when perturbed by noise or environmental factors, without intervention from outside the system. Anderson (1989), describing aircraft stability, states "a body is dynamically stable if, out of its own accord, it eventually returns to and remains at its equilibrium position over a period of time". In this respect, stability directly improves type I robustness. There are additional fundamental aspects of stability that enhance robustness and we will now cover these individually.

The first type, as noted above, refers to static and dynamic stability. Stability for a mechanical system[3] can be easily visualized using the three diagrams in Figure 28. The diagram on the left is analogous to stability and can be visualized as a ball sitting in a bowl. If the ball displaced in either direction, it will eventually end up at the base location. The base can be thought of as the design point or nominal operational configuration for the system. A system with these attributes will be robust to noise factors tending to move the ball. In the middle figure, we see a ball placed on a flat surface. If the ball is perturbed in this case, it will stay at the new location resulting in neutral stability. We note that the absence of stability in a complex system does not imply instability, as evidenced in this situation. Finally, the third diagram shows an unstable condition. In this case, the ball will veer sharply away from the design point if it

[3] A system does not have to be mechanical to have stability; stability applies equally well to other domains. We merely choose a mechanical system analog because it is easy to visualize and facilitates understanding.

is displaced in either direction, demonstrating a strong sensitivity to environmental factors.

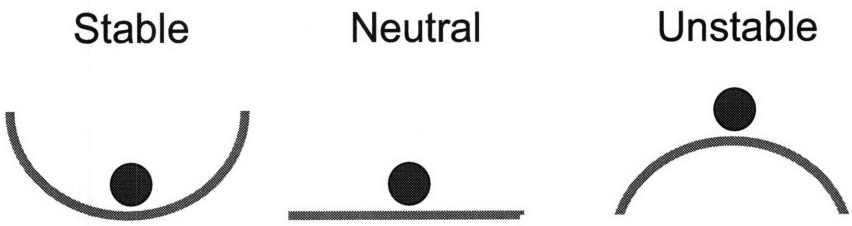


Figure 28. Visualizing Stability

System architectures can be inherently stable, as in a pendulum, or they can achieve stability through "artificial" means. For example, the F-117A fighter aircraft is inherently aerodynamically unstable, but achieves system level stability by employing an electronic fly-by-wire feedback and control system. In this context, stability can reduce the complexity and workload for the user. Furthermore, it may enhance robustness by precluding users from pushing the system past its limitations.

Clausing and Frey (2005) review a case study highlighting the benefits of inherent stability for jet engine fan blades. The fan blades of early medium and high bypass turbofan engines were susceptible to flutter vibration due to inlet flow distortions of the air entering the engine. Flutter was one of the critical failure modes for fan blades, and was originally addressed by placing spacers between the blades known as mid-span shrouds to limit lateral movement. The mid-span shrouds helped the problem, but were subject to significant wear and chatter. With an architecture change to wide-chord fan blades, the blades became more inherently stable, and this significantly improved the reliability of the fan module.

The next type of stability is compositional, meaning the elements of a system must be stable and have some level of permanence. They must exist for a length of time that is

appropriate for the intended architecture and functionality. An element lasting only a few milliseconds, such as Ununpentium (element 115), is probably not a good, stable, structural element for a long-term system. The compositional component of stability improves the robustness of a system to deterioration through time and use.

On a jet engine, parts deteriorate through wear and fatigue, and will eventually be non-functional. However, the time scale for this to occur is sufficiently long, and as a result the components are viable elements for the engine architecture.

The last type of stability implied by this principle works at the aggregate level, or a level significant to the architecture. Crawley (2005) indicates that "architecture is a representation of the stable properties of the system." The system may be in a constant state of flux, but it is this aggregate stability that gives the system its pattern of stable properties. This type of stability is commonly observed with social systems and financial markets as a few examples. Chaos theory tells us that randomness can produce stable structures. As Gleick (1987) observes, the essence of chaos is "a delicate balance between the forces of stability and forces of instability." Unreliable components can even by integrated to form reliable systems as von Neumann (1956) has demonstrated.

Jen (2003) discusses the differences between stability and robustness and comments that there are common aspects of stability and robustness, but robustness is much broader. She goes on to describe robustness as feature persistence in complex systems. I also acknowledge that robustness is much more encompassing than stability, but contend that stability is a contributor to achieving robustness goals.

Aspects of Robustness that are Improved by Stability

In terms of the P-diagram framework, stability improves the robustness of the system through the use of control factors. The control factors are established so that the system has inherent or augmented stability and is therefore desensitized to noise. This stabilization also helps to contain error states and provide a level of fault-tolerance. The stability of the elements that comprise the complex system minimizes internal variation,

deterioration and failure modes. By maintaining stability at an aggregate level with the environment, the effect of noise factors on the system are, on a net basis, minimized. Lastly, stability improves the response of the system to emergence and system context changes by providing stable properties and interfaces for evolution and adaptation.

Case Study – Gas Turbine Load Reduction Device (LRD)

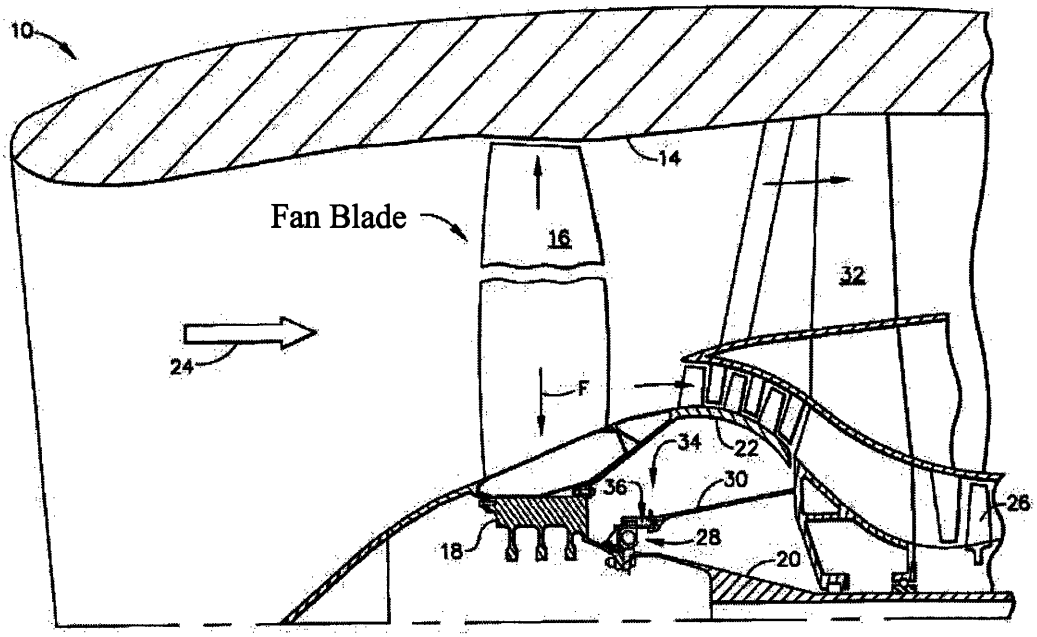


Figure 29. Engine Cross-Section Showing Fan Blades and Bearing Support (Doerflein et al., 2004)

Jet engines may be exposed to bird strikes and fan bladeout (FBO) failure events wherein a fan blade or piece of fan blade is released from its mounting. The fan imbalance in these conditions results in substantial structural loads on the rotor shaft and surrounding hardware. The imbalance of the fan causes it to hunt for a new rotation axis that reflects its unbalanced center of gravity.

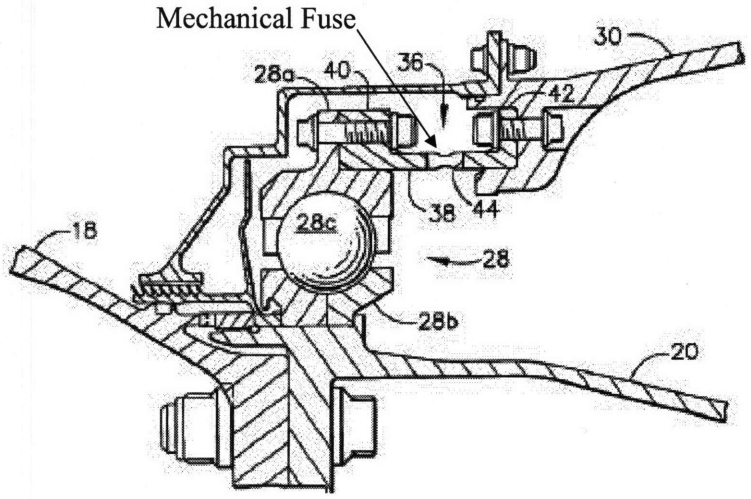


Figure 30. Close-up of Bearing Support

To improve the stability of the fan in these conditions, engineers have developed a load reduction device (Gerez, 1999; Kastl et al., 2002; Doerflein et al., 2004). The load reduction device incorporates mechanical fuses into the bearing support structure, which shear under pre-determined loads. The shearing of the mechanical fuses allows the shaft to move radially and stabilize at a new position. The shaft is still held in place by bearing supports aft of the fan, but the elimination of the fan bearing support in combination with the inherent flexibility of the shaft allows it to accommodate a center of rotation change. This architecture decouples the fan rotor and support system, greatly reducing the load on the engine structure and simultaneously helping to contain the damage. Moreover, with sufficient rotor clearances, the fan shaft can continue to rotate so the engine can "windmill" home to the airport.

6.3 Modularity

In chapter five, it was mentioned that as a new system enters the marketplace, much competition and experimentation occurs, with a dominant design eventually being established. In some cases, these dominant designs, or architectures, start out as very integral structures. This is a direct consequence of the system being designed for a specific purpose or to solve a particular problem. The system may satisfactorily fulfill the intended purpose, but it may be functionally limited as its use environment changes or customers demand more features. Integral architectures are difficult to upgrade as

customers' needs change and technologies evolve because subsystems cannot be easily removed or changed. The same holds true for maintenance, repair and other "ilities", in that parts cannot be quickly replaced without detrimentally affecting other components and functionality in the system. As failure scenarios arise, they might not be contained, which means small or avoidable problems could be seriously exacerbated.

Over time, architectures frequently become increasingly modular to address life cycle robustness. Modularity allows the system to be decomposed into subsystems for easier design and testing. Related functions can be bundled in a cohesive manner into standalone modules, and the coupling between modules can be kept very low. In this respect, the effect of environmental changes and technology evolution may be limited to a few modules that can be swapped out with upgraded modules. Internal variation, such as part deterioration, can be addressed with replacement or refurbished components. If the modules are uncoupled, or loosely coupled, the additional benefit of error and failure containment may be had.

Several types of modularity exist and have been by described by Ulrich and Eppinger (2004), Ulrich (1995), and Baldwin and Clark (2000) among others. Ulrich divides the modular architectures into three categories based on the organization of the component interactions and interfaces (see Figure 31). In the slot-modular architecture, each of the modules has a unique interface and as a result cannot be interchanged with other components. The bus-modular architecture incorporates a common bus to which all modules attach using a standardized interface. A sectional-modular architecture uses a standardized interface, but the components connect directly to each other and there is no common bus.

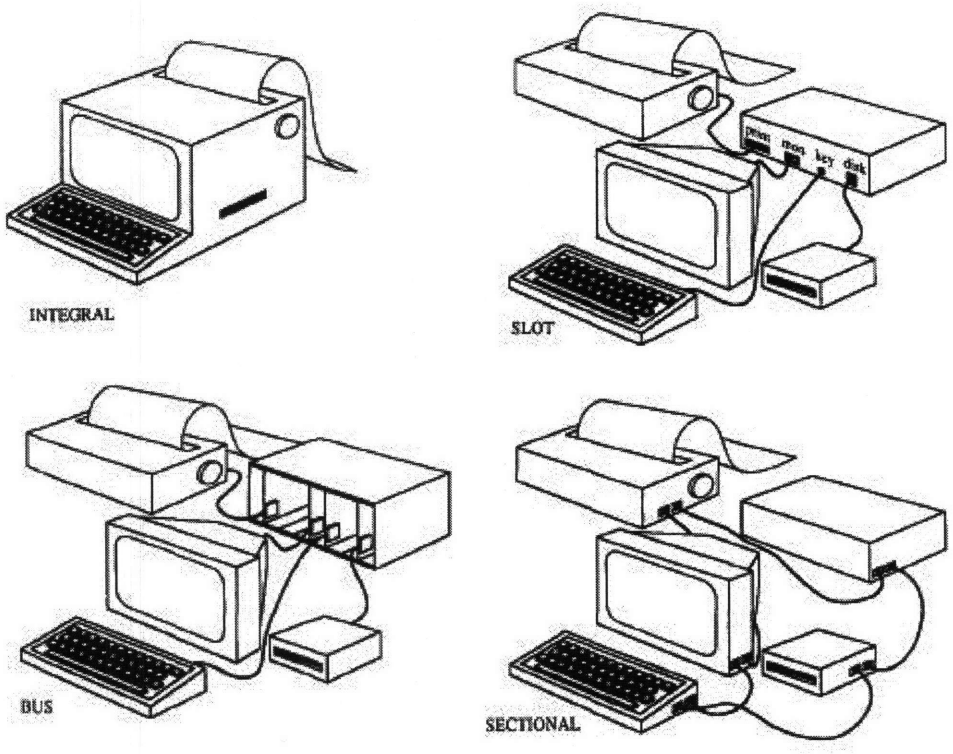


Figure 31. Types of Modularity (Ulrich, 1995)

The "ilities" are directly impacted by the modular architecture of the jet engine. For example, jet engines utilize a modular architecture with components that can be quickly added or replaced on the assembly line as well as on wing on the aircraft. These modular components are known as line-replaceable units (LRUs). Typical LRUs include engine control units (ECUs) and other electronic components. Because LRUs can be replaced on wing, repair time and cost are significantly reduced.

A second example of modular architecture helping the "ilities" involves the GE90-115B turbofan engine. The GE90-115B has an exceptionally large fan diameter of 135 inches, which means the Autonov An-124 is the only aircraft that can transport the engine in assembled form (Antonov, 2006). Transportability is significantly improved through the use of a modular propulsor concept (see Figure 32) that allows the engine to be separated into fan and propulsor modules. With this change, the jet engine architecture becomes substantially more flexible to transportability because it can be shipped on 22 types of aircraft (GE, 2006). The ability to store and handle the engines is also facilitated.

Maintainability and repairability are enhanced with the propulsor concept because the fan module, which requires very little maintenance, can be left on the aircraft while the propulsor, which requires periodic maintenance for life-limited parts, is removed and simply replaced. The replacement can be a new or overhauled propulsor unit. This offers the airline time flexibility in that they can wait for a given engine to be repaired or purchase spares and swap them out as needed to quickly return to service. GE indicates that 60% of the spare GE90 engines sold are propulsor modules only, which reduces acquisition costs by as much as 15%.

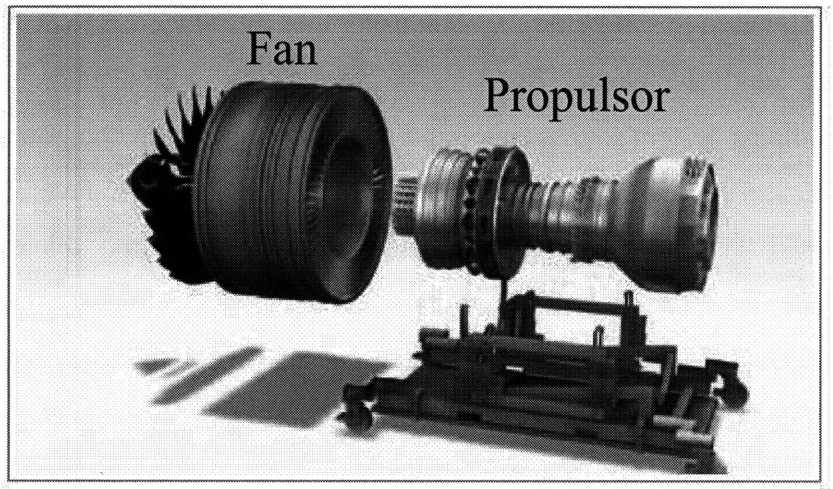


Figure 32. Propulsor Concept (GE, 2006)

Modularity allows a system to be evolvable and upgradeable as its system context changes over time. This is especially important for complex systems with long life cycles such as infrastructures (Crawley et al., 2004). A common change is that the component technology improves or is replaced with a new technology that achieves the same functionality. These noise variables are unavoidable as a system ages and its constituent technologies move along their respective "S" curves. Modularity allows the components or modules to be replaced to improve the system's competitiveness. In this fashion, the architect can use modularity to help prepare for unknown customer and market needs of the future. With increasing fuel prices, airlines have become increasingly sensitive to the fuel burn, or specific fuel consumption of the engines. To improve the efficiency of the jet engine, the customer has the ability to replace key

modules thanks to the modular decomposition. Some current upgrades available in the aircraft engine market include (Joyce and Rosario, 2002):

- CFM56-3 Core Upgrade
- CFM56-5C/P 3D Aerodynamic Core
- GE90-94B 3D Aerodynamic Compressor
- CF6-80C2 High-Pressure Turbine Durability

Aspects of Robustness Improved by Modularity

Modularity can be used to subdivide a system using the control factors to achieve enhanced failure mode avoidance. By using the physics of operation, an architect can segment the complex system so that coupling is minimized between components or modules, which allows each module to adjust independently and more favorably to noise factors. Because of the loose coupling between modules, the system is able to contain and prevent the transmittal of noise or error states to other parts of the system. Modular systems are easier to maintain and repair because the system can be disassembled and inspected, with components being replaced as needed. Modular systems are also easier to update, evolve and expand, which improves adaptability to system context changes.

Case Study – J57 Turbojet Engine

As an example of modularity improving robustness, the development of the P&W J57 turbojet engine is reviewed. The J57, known internally as the JT3, began development in 1949 and was part of P&W's attempt to catch up to and get substantially ahead of the competition. The following quote from P&W's Leonard Hobb (Lippencott, 1985) describes the situation at the time:

> We faced a mighty tough situation. Not only were we five years behind the other companies, but some of them could draw upon years of experience in the steam turbine field. We were running a poor race. We decided that it would not be enough to match their designs; that to get back into the race we must 'leap-frog' them – come up with something far in advance of what they were thinking about.

Engines developed prior to the J57 turbojet were designed and optimized for cruise operation. At conditions of lower and higher engine power, performance was degraded and the engine was subject to violent stalls. A stall in a compressor is when the airflow detaches from the rotor blades and as a result compression ceases to take place. The air can potentially surge out of the front of the engine damaging hardware and shutting the engine down. This problem at lower and higher power conditions is known as the "off-design dilemma".

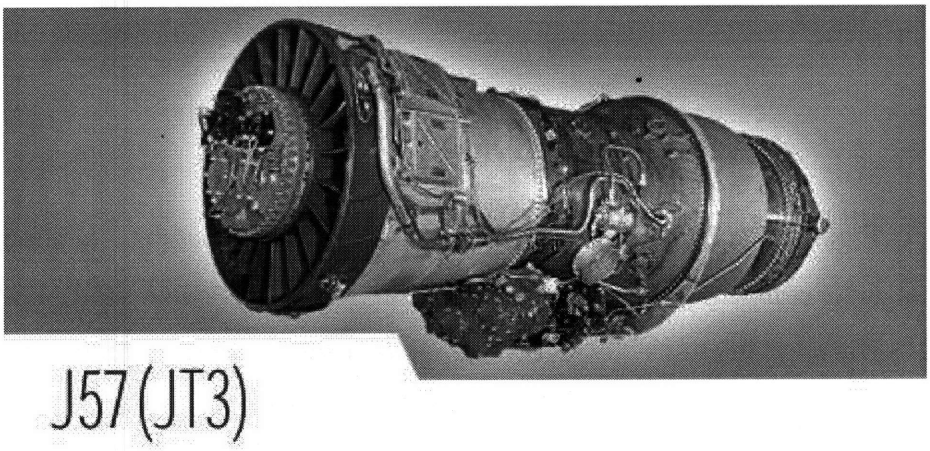


Figure 33. P&W J57 Turbojet Engine (Pratt & Whitney, 2006)

During normal operation, the air is compressed as it travels through the compressor so that the density (ρ) and pressure (p) are continuously increasing (see Figure 34). The design intent is to maintain constant velocity (V) through the compressor, so the flow area is reduced as the air moves aft to compensate for the increasing density. At low speed, the air velocity is too low in the front and too high in the back. This causes the flow to choke at the aft end of the compressor and the rotor blades to stall.

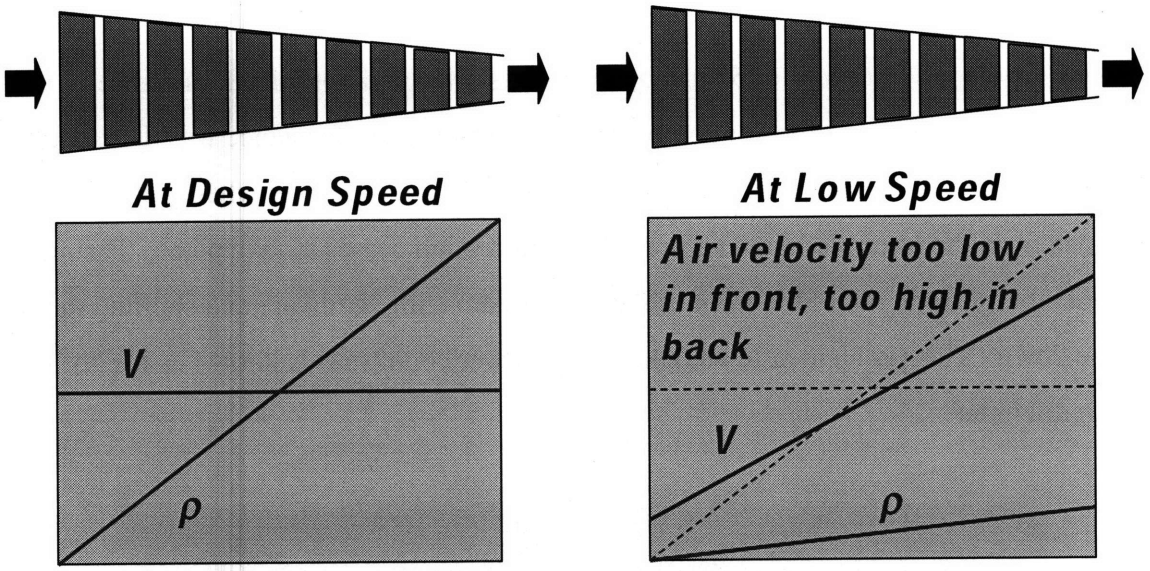
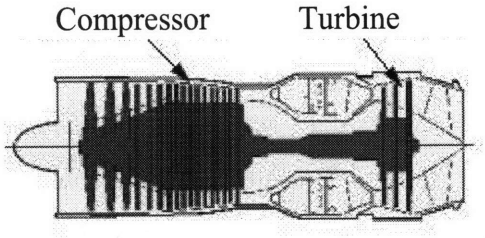
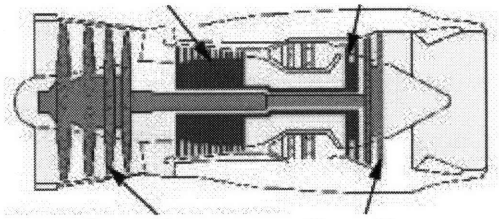


Figure 34. The Off-Design Dilemma (Blanton, 2004)

As shown in Figure 35, a conventional turbojet had a single spool architecture meaning there was a single compressor and a single turbine all connected to one shaft. P&W solved the "off-design dilemma" by switching to a two-spool architecture that split both the compressor and turbine into low-pressure and high-pressure modules. The low-pressure compressor and turbine were connected to one spool, while the high-pressure compressor and turbine were connected to a second spool. This architecture change decoupled the forward and aft sections of the compressor, allowing each spool to individually adjust to the airflow and rotate at a different speed. Not only did this change improve robustness to stall and surge by improving stall and surge margin (see Figure 36), it also improved engine performance. Moreover, it was an enabling architecture and technology for the turbofan jet engine. The J57 was one of P&W's most successful engines of all time, and the company received the prestigious Collier Trophy for their innovative efforts (St. Peter, 1999).



Single-Spool Turbojet



Two-Spool Turbojet

Figure 35. Two-Spool Vs. One-Spool Architecture

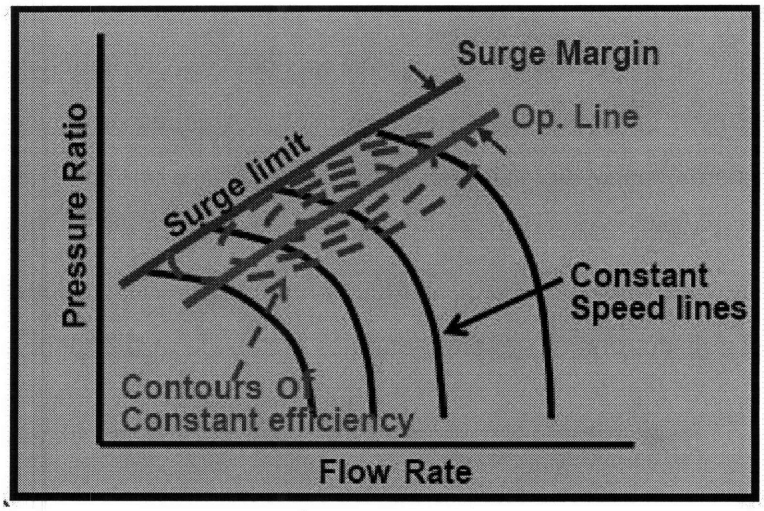


Figure 36. Compressor Stall Map / Margin (Prasad, 2005)

6.4 Feedback

For a sufficiently complex system to be robust and adaptable, it is desirable to have a mechanism for detecting the system's own operation as well as that of the environment. Feedback is the process of obtaining information about the system's environment and the output from the system, and utilizing this information to revise the system response. Because the output of the system is fed back into the input, this process is called a feedback loop. For a feedback loop, it is insufficient to only monitor certain attributes of the system. The information must be used by the system (or a person) to make adjustments as needed to the system response. The feedback process can occur at all levels within a complex system. It might take place from subsystem to subsystem or from the total system to its embedding system. Feedback improves type I robustness by responding to environmental noise, but it also enhances type II because of the increased adaptability.

Open and Closed Systems. A system that does not utilize feedback is known as an open system. In contrast, a closed system is one that leverages feedback loops and incorporates this information into the determination of the system response.

Feedback loops. Two types of loops can be used to describe all feedback interactions – positive loops and negative loops (Sterman, 2000). A positive feedback loop reinforces or amplifies a signal or perturbation. A negative feedback loop counteracts a signal or perturbation. This type of loop is also known as a self-correcting or balancing loop. In large complex systems, many thousands of feedback loops are operating simultaneously and the net effect of all the loops drives the dynamic behavior of the system. System Dynamics offers an excellent toolset for understanding and analyzing complex feedback systems, however a discussion of system dynamics is outside of the scope of this thesis.

In simple systems, a human frequently receives the feedback and provides the adjustments. As systems become more and more complex it becomes increasingly important to incorporate methods of determining the operational status of the system. Simultaneously, if the system is governed by a human, then cognitive limits will soon be

reached. At this point, the system must take on many of the interim controls and adjustments as a means to simplify the system for better usability. This simplification is the focus of two other principles (autonomy, and simplicity) that we discuss later in the chapter.

Feedback in ever increasing real-time. As a system evolves to higher levels of robustness, feedback becomes increasingly faster to the point of real-time information and eventually to prognostics. This aspect of the feedback principle might be referred to as feedback in ever increasing real-time. By reducing the latency and lag in the feedback loops, more accurate assessments can be made of the system and environmental responses. Similarly, Fey and Rivin (2005) have remarked, in the context of TRIZ, that technological systems evolve from real-time feedback to forward sensing in order to "guarantee good performance in the presence of uncertainty". Other authors have discussed the classic aspects of feedback, however here it is suggested that feedback is an effective means to increase robustness and additionally that feedback occurs at faster rates as systems evolve and this contributes positively to robustness.

Figure 37. Trend toward Faster Feedback (Modified from Fey and Rivin, 2005)

Aspects of Robustness Improved by Feedback

Feedback can improve robustness through the use of control factors by sensing noise factors and the operational response of the system, and subsequently reconfiguring the system as needed. Using the sensed feedback, a system can adapt the form and function of the system to avoid failure and minimize the effect of noise factors. With feedback, a complex system can also sense environmental or internal noise and make adjustments to the signal to maintain a desired response. Frey and Jugulum (2006) note that a noise signal could be measured and cancelled using destructive interference. The impact of error states can be minimized because the system can detect the errors and compensate

using other functional parts of the system. With both autonomy and feedback, the system may also be able to correct the error state. Feedback improves the ability of the system to adapt to system context changes by sensing and responding to environmental changes and emergent behavior.

Case Study – Engine Sensors

As Table 9 (Jaw and Garg, 2005) illustrates, the number of sensors in a jet engine has been steadily increasing on average since 1942. The trend for military engines is steeper than commercial engines as can be seen in Figure 38, but both types are increasing. These sensors provide feedback to the engine that significantly improves the system robustness, even though in some cases the sensor may be a lower reliability component (St. Peter, 1999). A basic fuel control system can be used to illustrate how feedback is utilized in the control system. A pilot sets the throttle to a desired power setting. The engine control system then works to keep this power setting by taking readings from the various rotor speed, pressure and temperature sensors and adjusting the fuel flow rate as needed. For example, as the ambient environment changes with altitude, weather and inlet distortions, the control system adjusts the fuel flow to compensate for these noise factors.

Table 9. Evolution of the Number of Engine Sensors

Engine model	Year (flt) tested	Number of engine sensors
GE I-A	1942	3
GE J47	1948	5
GE J79	1954	5
GE TF39	1966	6
GE CF6	1968	6
PW F100	1970	8
AR TFE731	1972	5
GE F101	1972	7
CFM56	1974	6
GE YF120	1989	11
PW F119	1990	10
GE90	1993	7

Additional sensors have been added over the past few decades to further improve system robustness. For example, temperature harnesses are added in the nacelle region to detect possible fires between the engine and nacelle. Other sensors are used to detect temperature exceedances in the core of the engine or to identify clogging in oil lines. This information is then used by the engine control system or by the pilot to take corrective action. The architecture change to include real-time feedback has greatly improved the robustness of jet engines by reducing the effect of noise factors and by preventing problems from occurring in the first place.

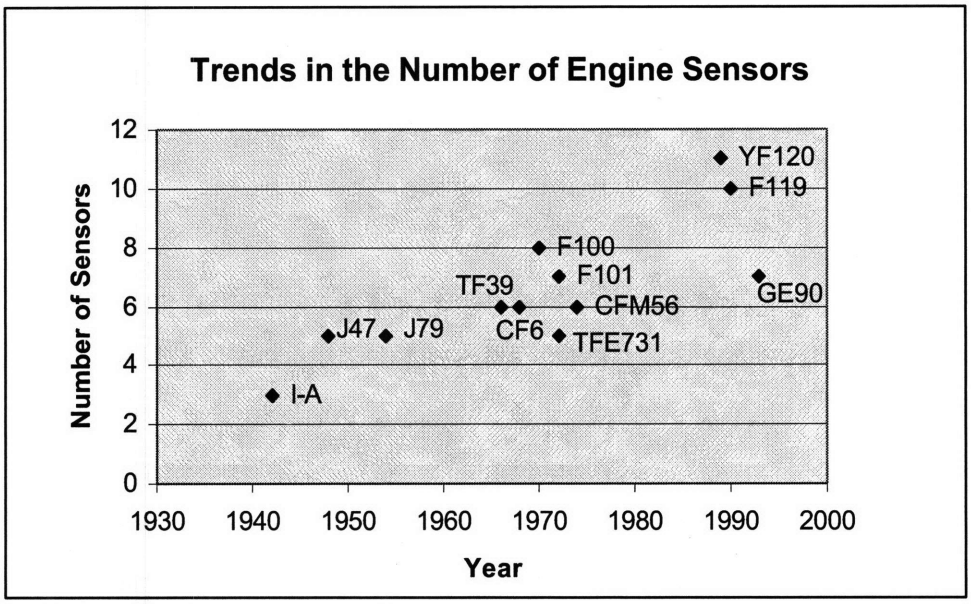


Figure 38. Trends in the Number of Engine Sensors

6.5 Standardization

The standardization principle implies that standard interfaces, protocols, and properties should be incorporated into an architecture wherever possible. With standards, all parties understand the specified aspects of a component and this reduces uncertainty throughout the system life cycle. Standardization also reduces complexity by minimizing the number of types of elements in a system and minimizing the amount of information to specify interfaces. All of these factors make the architect's job easier, but they also facilitate product assembly, maintenance and repair because components can simply be replaced with another standard unit.

Doyle et al. (2002) use LEGO bricks as an example of protocol or interface standardization. The snap-together LEGO interface is used on all of the bricks, and this feature provides robustness in assembly as well as adaptability in use. This example also shows the counter-intuitive result that standardization can simultaneously promote both robustness and creativity in the architecting process. LEGOs have been around for many decades and yet children are still entranced by these objects and generate original architectures.

Standards also establish common processes and communication protocols that facilitate robustness. These processes typically incorporate the lessons learned of the past to preclude repeat failures and mistakes. In this respect, the architect "builds-in" some nominal level of robustness simply by following the standards.

Since the turbojet's inception, standardization has walked its way through an ever increasing amount of the engine. The Federal Aviation Authority (FAA) maintains standards in the form of Federal Aviation Regulations. For example, airworthiness standards for aircraft engines are covered primarily by part 33. These standards guide the design, testing and certification of aircraft, and incorporate the collective wisdom of the past to avoid repeat failure events. As new safety issues and failure events emerge, the standards are updated to capture and lock-in the knowledge, while preventing future occurrences. Since the first FARs were established, the system has grown increasingly more complex and comprehensive. Currently there are five volumes that include approximately 1400 "parts". Furthermore, aircraft engine systems are governed by material, component and process standards such as MIL-SPECs and MIL-STDs. Standardization in the aircraft industry has improved numerous facets of robustness and as a result it continues to grow in new directions. The Boeing 787, currently in development, will be the first commercial aircraft to offer a standardized interface for its jet engines (Boeing, 2006). The customer will be able to switch between the two engine manufacturers, namely GE and Rolls-Royce, at any point in time.

Aspects of Robustness Improved by Standardization

Standardization reduces noise factors during the production process by establishing standard parts and assembly techniques. The same can be said for operation of the system; standard components and operating principles facilitate better understanding, thus eliminating errors in use and maintenance. By leveraging standard interfaces and protocols, standardization improves the ability of a system to be modified, updated and evolved in parallel with changes to the system context.

Case Study – Engine Maintenance

In-flight shutdowns (IFSDs) of jet engines are periodically caused by production and maintenance errors. Cheldelin et al. (2005), referencing two separate studies of FAA data, show that using the wrong parts is one of the top causes of maintenance related IFSD events. A 1991 FAA study, based on 120 events for the Boeing 747, listed wrong parts as the second largest contributor, while a 1993 study (see Figure 39) shows it as the third largest factor.

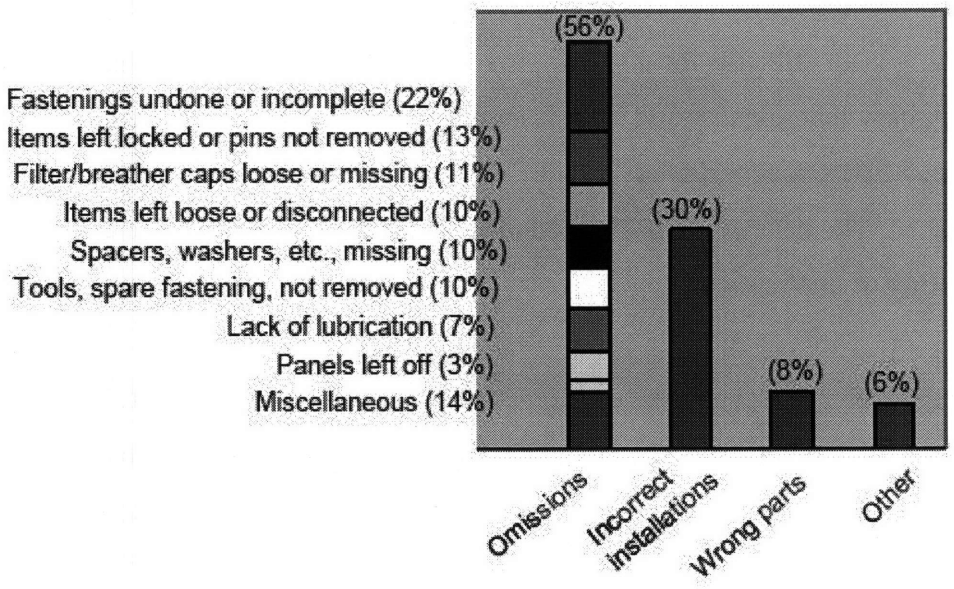


Figure 39. Boeing Study on 122 Maintenance Errors Occurring over 3 Years (Boeing, 1993)

The standardization of system elements significantly diminishes the failure mode of using the wrong parts. One step is to combine parts that are similar and are subject to

confusion. For example, standard bolt diameters or lengths can be used to avoid fastener "mix-ups". In this manner, it will be visually apparent if the technician retrieves the wrong fastener because of the large step size between standard diameters. Standardization improves the life cycle robustness of jet engines from production to maintenance through simplification and elimination of error states. The T700 turboshaft engine, for example, made extensive use of standardization in its design. The result was an engine that only needed a twelve-piece toolkit for maintenance and repair (St. Peter, 1999).

6.6 Redundancy

Redundancy refers to duplication of functionality, form (structure) or resources in a complex system. More specifically, the duplication is in excess of what is needed for nominal product or system operation. A typical implementation of redundancy is to include two or more identical parts such as a spare tire in a car. Redundancy effectively improves robustness by providing a backup in failure conditions and a level of fault-tolerance for the system. This improvement is somewhat counterbalanced by additional complexity and failure modes associated with the additional structure. While redundant functionality is an effective means to improve robustness, jet evolution indicates that the use of auxiliary functionality and form is a much better approach. Auxiliary functionality implies that different concepts or structure are used to produce the same desired functionality. The benefit of this method of redundancy is that it eliminates the susceptibility of duplication.

In 2004, the Defense Advanced Research Projects Agency (DARPA) established a robotic competition known as the Grand Challenge. Engineers from across the country competed to design an autonomous robot that could traverse 175 miles across the harsh desert environment in the least amount of time. In addition, the winner had to finish in less than ten hours to claim victory. At the end of the first race, the best robotic vehicle had traveled a mere 7.5 miles. William Fredkin, Professor of Robotics at Carnegie Mellon University, described his top three lessons learned from the first competition, with the number one being "sensing robustness counts" (Design News, 2005). He goes

on to say, "You need multiple modes of sensing. In a barrage of dust, one sensor might do well while another fails." The architect should aim to create alternate paths to the same functionality for select, critical functions.

Aspects of Robustness Improved by Redundancy

The redundancy principle improves the robustness of a complex system through several mechanisms. Control factors are leveraged to build multiple paths in the system to accomplish the desire functionality. In this way, redundancy provides the system with backup systems that can take over in the event of a failure or error condition. The architect must be cognizant that adding redundant systems may also introduce new failure modes. In autonomous systems with redundancy, a voting rule can be established to identify the correct signal or response from the various redundant sub-systems. Redundancy reduces the criticality of noise because, with the implementation of auxiliary functionality, noise affecting one sub-system may not affect another. Along these lines, the system response may be improved because the response can be carried out in different ways. In the event of a system context change, redundancy gives the architect or user more options for change. If one sub-system or interface does not meet the new needs, then one of the redundant options may.

Case Studies – FADEC Control System, Turbine Cooling and Engine Starting

On a modern turbofan engine such as the GE90, the Full Authority Digital Engine Control (FADEC) unit is the primary computer that controls all functions during operation. Redundancy is built into the FADEC via a dual channel system. All components connecting to the FADEC, such as temperature and pressure sensors, can operate on a channel A or a channel B. Each flight, the channel is switched to ensure that both channels are consistently used and problems can be identified before takeoff. In the event of a problem, the channel can be switched to the backup in order to resume normal operation.

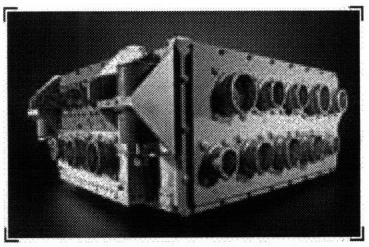


Figure 40. Redundancy in the FADEC III Control Unit (Hispano-Suiza, 2006)

Turbine Vane Cooling

Another example of classic redundancy is the use of multiple compressor bleed tubes for turbine stator cooling. While there is the additional intent of providing a uniform flow and pressure distribution around the high-pressure turbine, the use of multiple tubes protects the system in the event of a broken pipe during operation. The flow from the remaining pipes will redistribute around the turbine so that the vanes are sufficiently cooled.

Engine Starting

It is critical that a jet engine have starting capability at all points during its operation. To preclude potentially dangerous events during flight, multiple modes of starting have been established should an in-flight shutdown (IFSD) engine event occur. As a result, redundant starting mechanisms have been architected into the system. The techniques vary somewhat depending on the aircraft, so our system of interest here is a Boeing 777 aircraft equipped with GE90 engines. On the tarmac, the pilot has the option of using ground air from the airport, or by bleeding high-pressure air from the auxiliary power unit (APU). During a flight, the engine can be started with either the APU or by employing cross-bleed from the other engine (777 has two engines). These two options are in addition to the basic functionality of doing a windmill start.

Engine Starting Techniques
Windmill start
Ground air
Bleed air from APU
Cross-bleed from another engine

Figure 41. Aircraft Engine Starting Techniques

The next four principles (independence, autonomy, simplicity and scalability) have been discussed at length in the literature as mentioned in the introduction to chapter six, and were also observed to improve jet engine robustness. They are included here to complete the robustness framework. Other principles will likely be codified in the future and should also be added to the framework.

6.7 Simplicity

The simplicity principle is oriented towards making a complex system as simple as possible while still meeting all of its functional requirements. This concept differs from simplicity implied by the Information Axiom of Axiomatic Design, which requires the information content of a complex system to be minimized (Suh, 1990). For the Information Axiom, the relationship between the information content and the probability of success is given as:

$$I = -\log_2(P_S)$$

Suh (1990) additionally writes that for the more general case of an uncoupled design with multiple (n) functional requirements, information content is:

$$I = \sum_{i=1}^{n} [\log_2 \frac{1}{P_i}]$$

Therefore, in order to maximize the probability of success (p), the information content must be minimized. The simplicity principle differs in that it addresses descriptive complexity.

From a qualitative standpoint, the simplicity principle has a few other important goals. As discussed in chapter three, Crawley (2004) indicates that the essential complexity must be kept to a minimum. However, he also notes that the actual complexity of the system should be as close as possible to the essential complexity. The simplicity principle assists the architect in minimizing actual complexity to essential complexity. Another imperative is that the perceived complexity must be minimized and maintained within the limits of human cognition. This eliminates complicated systems, helping engineers to better understand the system and avoid mistakes during the design process. Additionally, it decreases customer use mistakes out in the field.

The simplicity principle may appear to be in conflict with the "robust functionality drives essential complexity" principle introduced in chapter three. The system complexity as a whole will increase over time as new functionality is incorporated, but simplicity guides the actual complexity of the system to be minimized to the essential complexity. Simplicity improves robustness by guiding the architect to create simple, understandable and perhaps elegant architecture. It incorporates all required functionality, while minimizing the form. There are no excess components that could fail and cause downstream damage. As the saying goes, a part that is not there cannot fail.

Aspects of Robustness Improved by Simplicity

The simplicity principle effectuates robustness by minimizing the number of control factors in the system, which in turn diminishes the number of failure modes. Furthermore, simple systems are easy to understand, which helps cut down on production and operator errors. As the system context changes, these well-understood and minimally complex systems are readily evolvable, making them robust to the "ilities".

Case Study – T700 Turboshaft Engine

The T700 is a turboshaft engine developed by GE in the mid 1970s for use in the UH-60A Black Hawk helicopter (GE, 2006). At the start of development, the Army identified reliability and simplicity as two of the key design requirements.

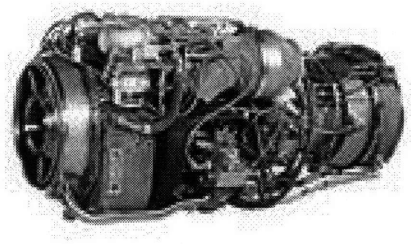


Figure 42. GE T700 Turboshaft Engine (GE, 2006)

The compressor configuration for this engine used five axial stages followed by one centrifugal stage. Typically axial compressor stages are comprised of numerous blades that slide onto a large disk, leading to a high part count. For the T700, integral blade and disks, known as blisks, were introduced into the design to reduce part count and improve simplicity (see Figure 43). The architecture change from independent blades to single piece, monolithic blisks greatly reduced the compressor part count from 300 on the demonstrator to 12 pieces on the production engine (St. Peter, 1999). In addition to simplifying the design, the blisk architecture also eliminated a potential failure mode. On a conventional compressor stage, the base of the blade, called the dovetail, rubs against the rim of the disk causing wear on both components. As a single piece structure, the blisk stops relative motion between the blade and disk, which eliminates dovetail and disk wear.

Figure 43. T700 Compressor Blisk (DRS, 2006)

6.8 Independence

The independence principle flows directly from Suh's (1990) Independence Axiom, which states that the architect must maintain the independence of the functional requirements (FRs) of a system. An axiom can be thought of as a fundamental truth that does not have exceptions. If the ideal Independence Axiom criteria are met, then the system design is uncoupled, meaning that an adjustment of one control parameter will not affect multiple system functions. Each design parameter addresses a single functional requirement in this scenario. An uncoupled design corresponds with a diagonal matrix at shown in Figure 44, where the rows represent the functional requirements and the columns represent the design parameters (DPs). Individual design parameters can be refined independently to optimize the robustness of a single function. A system can also be decoupled and still satisfy the Independence Axiom if the design parameters are changed in the correct order. A decoupled design will have a triangular matrix as shown in the middle picture of Figure 44. The third type of design is a coupled system, which is undesirable because it violates the independence axiom and creates unnecessary complexity.

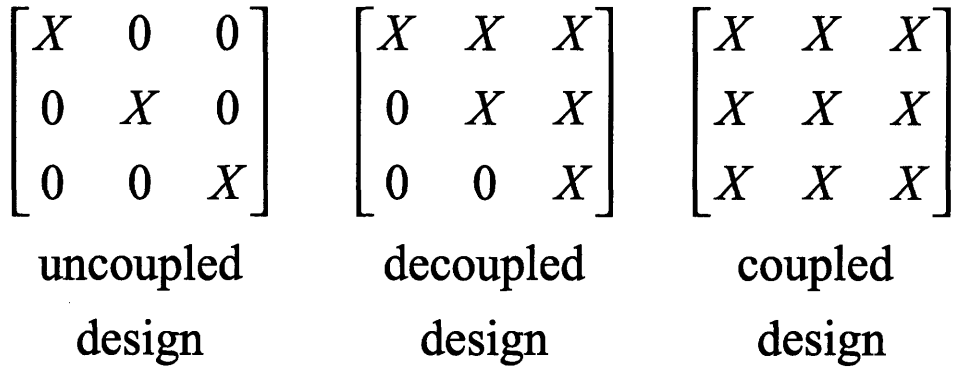


Figure 44. Types of Design Matrices (Suh, 1990)

Independence facilitates architectural robustness in several ways. Suh (1998) argues that it improves the controllability and stability of the architecture, so the system performs reliably in the presence of external noise. Frey (2005) observes that it simplifies system operation, making the architecture more transparent and easier to understand. Changes can be made quickly and more easily to the system lending to its adaptability and maintainability. Another benefit is that during the architecture and design processes, the system can be subdivided and developed in parallel. In jet engine design it has been challenging to create an uncoupled architecture because each of the primary gas path components is linked to another via a long spool or shaft. The architectural progression, however, shows that the coupling is being reduced over time. For example, jet engines started off with a single spool architecture and then progressed to a two-spool design. Currently, Rolls-Royce offers Trent turbofan engines with three spools.

Aspects of Robustness Improved by Independence

This principle minimizes the effect of noise due to the independence of the functions and their associated control parameters. Error states will tend to be contained, diminishing their impact, because of the very low coupling between system functions. If new needs or uses arise in the field, the system functionality can be adjusted independently to address the new requirements and maintain robustness.

6.9 Autonomy

The autonomy principle implies that a complex system is self-governing and can act without external control. It is tied intimately to the feedback principle, because a system must have feedback mechanisms, such as sensors, to perform autonomously. A control system must accompany the sensors to close the loop and put the feedback to good use.

Autonomy enhances architectural robustness for sufficiently complex systems in several key ways. First, it allows a system to adapt to new environmental conditions and factors. Second, with autonomy the system can manage some non-fatal error states and continue to operate as intended. It may also have self-healing capability. In this capacity, it improves both type I and II robustness. Because the system can act without external control, it is particularly useful in applications where the system is very remotely located or inaccessible. The system might automatically perform self-checks or other pre-determined actions at select intervals.

One of the disadvantages with the addition of control systems, feedback systems and actuators needed to create autonomy is that it also increases complexity. Taken individually, these components may have lower reliability or introduce new failure modes, so the system must be integrated in a manner that optimizes overall robustness and reliability.

Fey and Riven (2005) suggest that autonomy is part of the natural technological evolution of complex systems. They indicate that systems progress from passive systems, to operator-controlled systems and finally to active-adaptive systems. Autonomy is an active area in jet engine research, with programs like Autonomous Propulsion System Technology (APST) and the intelligent engine well underway.

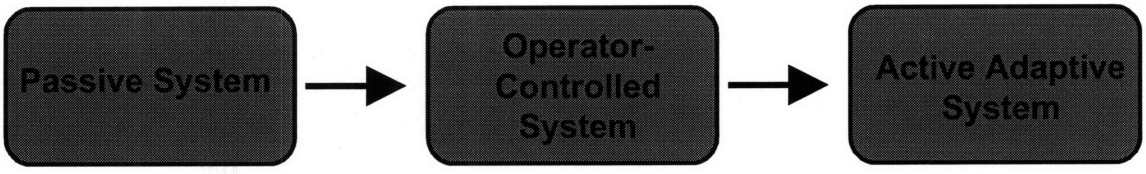


Figure 45. Line of Transition to Active Adaptive Systems

Aspects of Robustness Improved by Autonomy

The autonomy principle improves robustness through many different means. Autonomous systems can reduce the sensitivity of the system to noise by changing control parameters. This is accomplished by the system reconfiguring itself or invoking alternate operating modes, sometimes at pre-ordained intervals and in other cases by using sensed feedback to guide the process. Autonomous complex systems can compensate for incorrect or sub-optimal signal settings. Alternately, it can sense environmental change and preemptively make adjustments to the signal to maintain a response. The autonomy principle, in tandem with the feedback principle, can act to eliminate noise factors or counteract their effects. A system with autonomy can detect and possibly perform self-repair as error states occur. Additionally, the system may have default "workarounds" and safe modes for failure modes. The relative autonomy of the modules in an architecture acts to limit error propagation. The response variable can be used to improve robustness as the system can adjust its output until the desired response is achieved. If the system does not have sensing, it can cycle through and utilize different operating actions to increase the chances of the response matching the intent. Autonomy is especially effective at improving robustness to system context changes because of its adaptability. The system can readily detect, reconfigure and adapt itself to emergent behavior.

6.10 Scalability

Scalability is the ability of a system architecture to be expanded, contracted or scaled to meet stakeholder needs as well as changes in the system context. Crawley et al. (2004) define it as "the ability of a system to maintain its performance and function, and retain all its desired properties when its scale is increased greatly, without causing a corresponding increase in the system's complexity." The scalable architecture can have self-similar attributes similar to those observed in fractals. The robustness of a system to changing customer needs is greatly enhanced with the flexibility afforded by a scalable architecture.

As Figures 48 and 49 illustrate, there at least four methods to increase or decrease the scale of a system. The first category of methods operates on the elements themselves. Within this category, the first type treats the baseline architecture as a modular building block. Elements can be added together to create larger or higher level systems much as a bricklayer creates a new house. In this respect, a system can be built in an accretive manner. If the system is already comprised of multiple modules, then conversely one or more can be removed to reduce the size and scope of the system. The second type is geometric in nature with the attributes of a single architecture or module being scaled upwards or downwards.

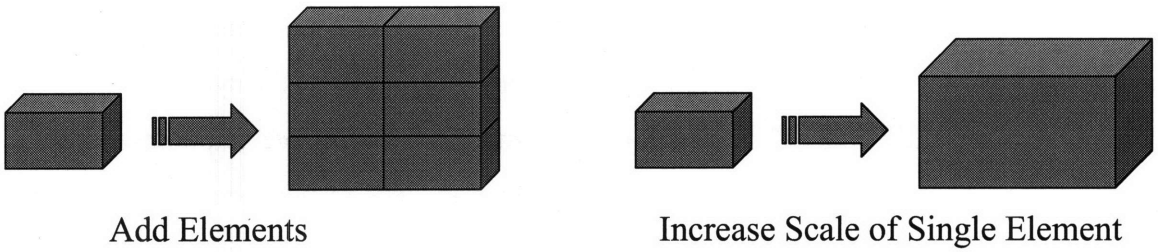


Figure 46. Types of Scaling for Elements

The second category of methods operates on the connections between elements (see Figure 47). Scaling can be realized by increasing the number of connections between elements or alternately by increasing the bandwidth of the existing connections. All of the techniques mentioned in this section can be used individually or in combination.

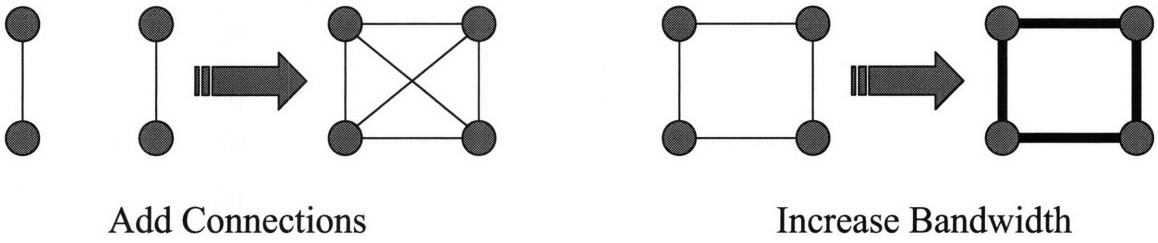


Figure 47. Types of Scaling for Connections

GE's product portfolio of commercial engine platforms is illustrated in Figure 48. Scalability of engine architecture is evident as one views the four existing platforms, with each platform covering a different thrust range. The approximate thrust ranges for each engine are shown on the ordinate axis. The proven GE90 engine architecture was scaled upwards in the early 2000s to create an entirely new, higher thrust engine in the 115,000 lbf class. More recently, the GE90 core[4] architecture was also scaled down to create entirely new product platforms for the GP7000 and the GENX engine product lines.

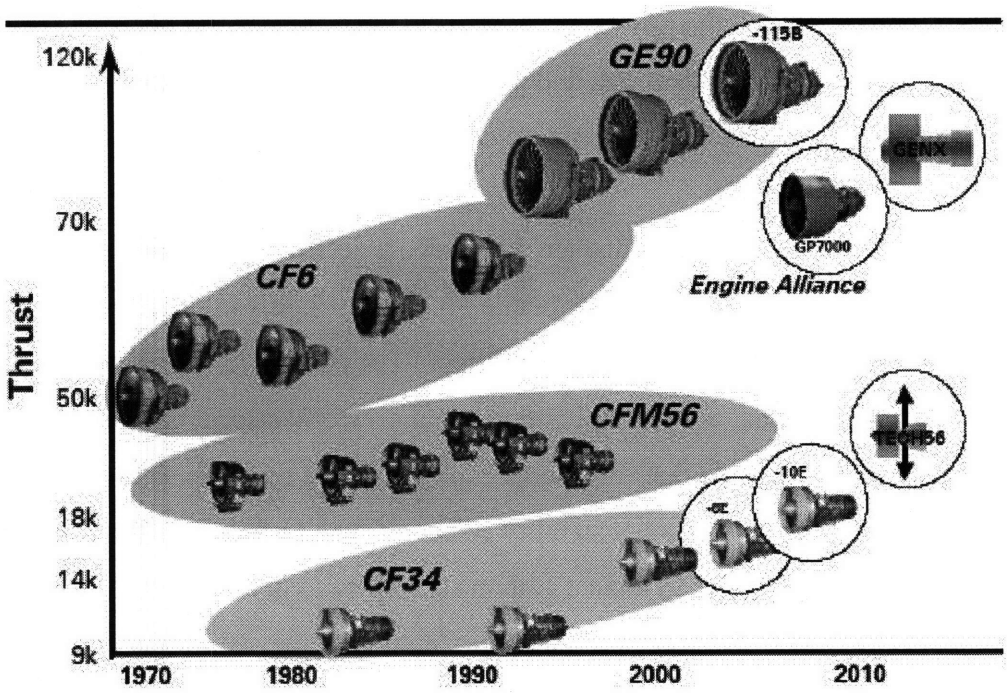


Figure 48. GE Commercial Engine Development (Joyce and Rosario, 2002)

Aspects of Robustness Improved by Scalability

The scalability principle primarily contributes to life cycle and type II robustness. Scalability imparts adaptability into the system so that changing market and customer needs can be met. The system control factors can be scaled to increase or decrease functionality as needed to meet new requirement and uses providing a robust system foundation that can be evolved over time.

---

[4] The core of a jet engine is known as the gas generator and includes the high-pressure compressor, combustor and high-pressure turbine.

6.11 Summary and Framework

The principles of robust system architecture have been summarized again in Table 10. Frey and Jugulum (2006) have previously suggested a Parameter Diagram framework to organize robustness inventions, and this framework is used here to categorize the principles. The P-diagram is especially useful because it is known by engineers practicing robust design in the field and is well established in the literature. The categories of the framework show which aspects of robustness can be improved for a given principle. Alternately the categories may be viewed as the mechanism by which robustness is enhanced with the principle. An error states category has been added to the standard diagram for type I robustness, and additionally a "system context change" category is included to address the expanded scope of type II robustness.

For each principle reviewed in this chapter, I included a detailed description of the aspects of robustness improved by the principle. In Table 10, this information is condensed into a useful framework for the architect. The convention for the table is as follows: a "+" indicates that the principle improves the specific aspect of robustness and a "0" indicates that the principle does not significantly improve or diminish the parameter. For example, the first row shows that stability enhances type I robustness through the manipulation of control factors to reduce system sensitivity to noise and stability also reduces the level of noise. Moreover, the implementation of the stability principle in a complex system improves type II robustness.

Table 10. Principles in P-Diagram Framework

	Type I Robustness					*Type II Robustness*
Principle	*Control Factor*	*Signal Factor*	*Noise Factor*	*Error States*	*Response Variable*	*System Context Change*
Stability	+	0	+	+	0	+
Modularity	+	0	+	+	0	+
Feedback	+	+	+	+	+	+
Standardization	0	0	+	+	0	+
Simplicity	+	0	0	+	0	+
Independence	0	0	+	+	0	+
Autonomy	+	+	+	+	+	+
Redundancy	+	0	+	+	+	+
Scalability	0	0	0	0	0	+

In this chapter, I reviewed the principles of robust architecture and provided a detailed explanation of each principle. After listing the aspects of robustness that were improved, case studies were examined showing the principles in action. Lastly, a useful framework was presented showing the robustness benefits. In chapter seven, I show how the architect and engineer can integrate this framework into the established robust design and product development processes.

7. The Product Development Process

"Hell, there are no rules here – we're trying to accomplish something."
- Thomas A. Edison

The processes of a business organization can all be categorized as either product acquisition, supply chain, sales and marketing, or integration and direction (Clausing and Fey, 2004). Of the four enterprise processes, this chapter will examine product acquisition, or more specifically the product development process (PDP) in more detail.

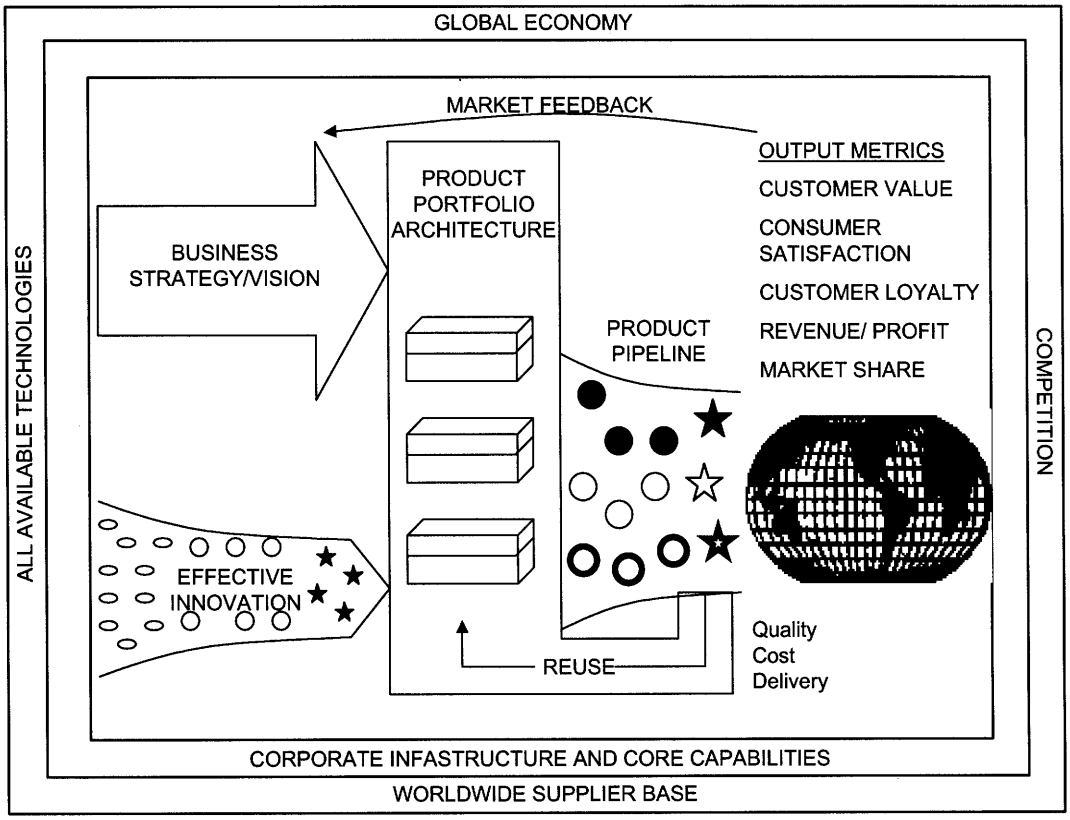


Figure 49. The Product Acquisition Process (CIPD, 2004)

Section 7.1 walks through the product development process and illustrates how system architecting fits in to the overall procedure. In section 7.2, I propose a robust system architecting methodology and show how it can be integrated with the existing robust design processes. I then discuss a common business implementation of robust design in

section 7.3, namely design for six sigma (DFSS). Finally, the chapter is concluded by walking through an enhanced DFSS process that incorporates the robustness framework generated in chapter six.

7.1 The Product Development Process

All businesses follow some form of PDP, although many different blends and varieties are in use depending on the particular industry and organization. At a higher level of abstraction, a typical PDP includes conceive, design, implement and operate phases as shown in Figure 50 (MIT, 2006).

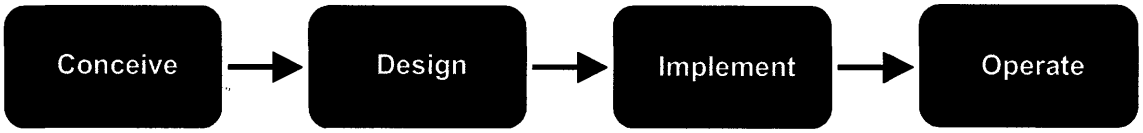


Figure 50. Generic Product Development Process

There are two primary embodiments of the generic PDP, and we will begin with the spiral product development process. The spiral development process (see Figure 51) is used in what Ulrich and Eppinger (2004) call "quick-build" products. These are products, such as some types of software and electronics, which require rapid iteration and a more flexible process.

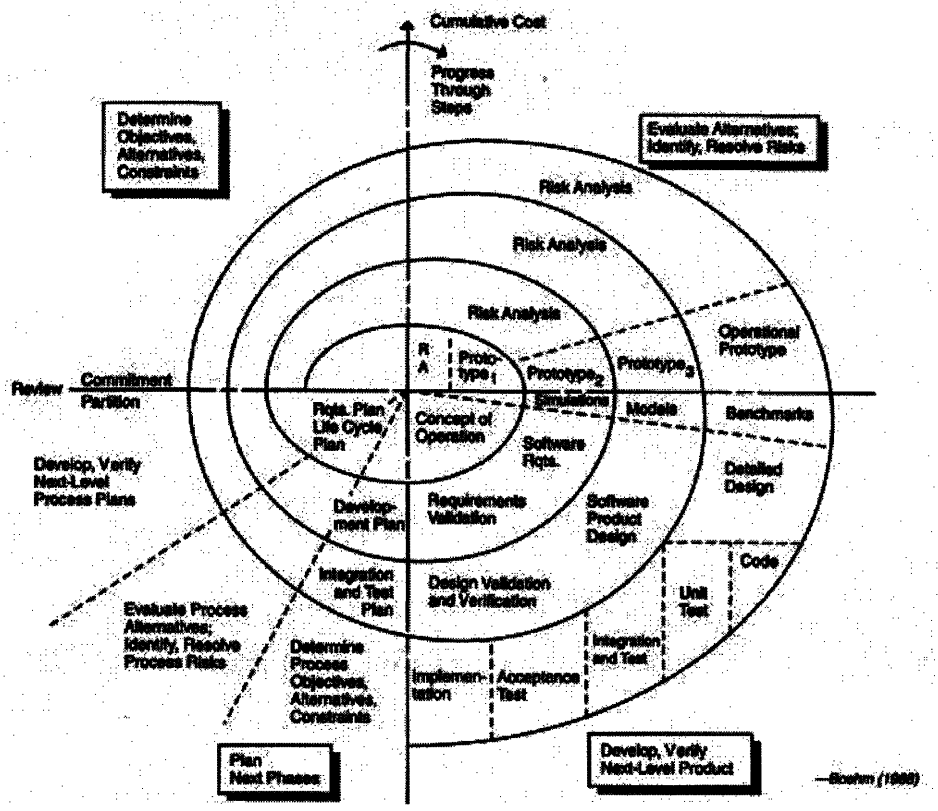


Figure 51. Spiral Development (Boehm, 1988)

In developing complex systems, several changes are evident in the PDP as compared to spiral development. Much more emphasis is placed on conceptual and system level design. Different top-level product architectures are evaluated and the system is subsequently decomposed into subsystems and components. In addition, substantial efforts must be made during the system integration, and testing / validation phases. We will focus on this more detailed process as our interest lies in complex engineering systems. Figure 52 shows the additional detail in the PDP, with the conceive phase split into mission (sometimes called definition) and conceptual design phases. Beneath the development process, we see a breakdown of the various tasks to complete at each stage.

The architect's domain is at the front of the PDP as Figure 52 illustrates, with responsibilities in the mission definition and conceptual design phases. The architecting process touches nearly all steps in the conceive phase with the only exception being strategy development. Crawley (2005) defines the role of the architect as follows:

- To define the system boundaries, goals and functions
- To create the system concept
- To allocate functionality
- To define the interfaces and abstractions
- To manage the evolution of complexity

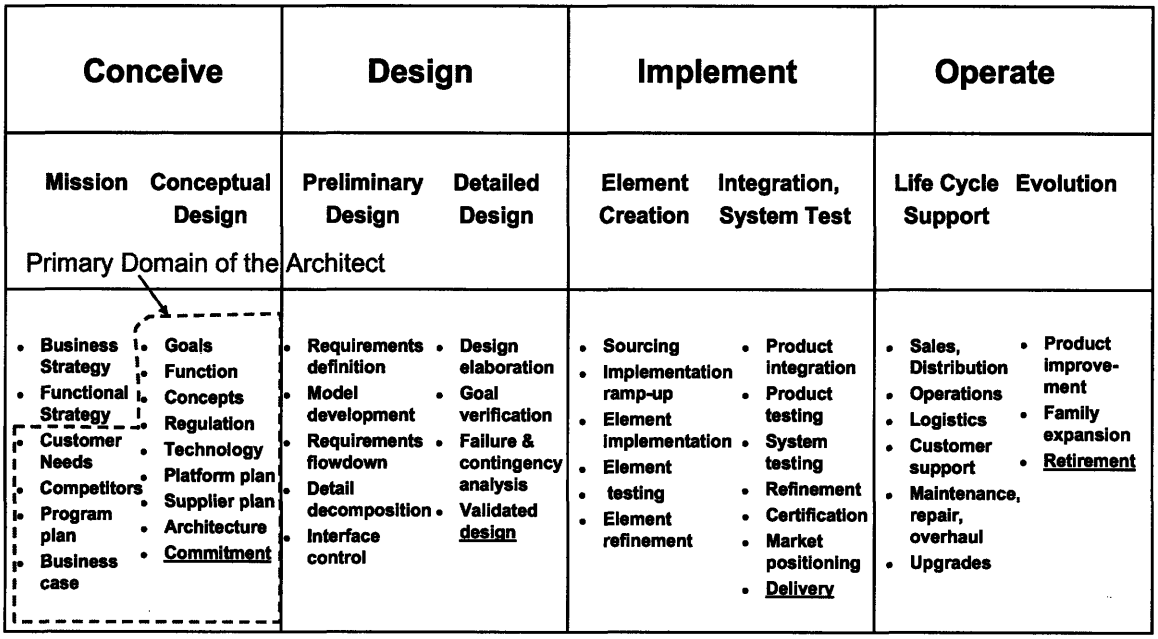


Figure 52. Generic Product Development Process (Crawley, 2005)

7.2 Robust System Architecting Methodology

To develop robust complex systems, a robust architecting methodology can be utilized that joins synergistically with existing robust design methods (see Figure 53). Our discussion will focus on the conceive phase of the product development process as parameter design and tolerance design are reviewed extensively in the literature and are outside the scope of this thesis. We will now step through the methodology at a high level to give the reader a feel for the process.

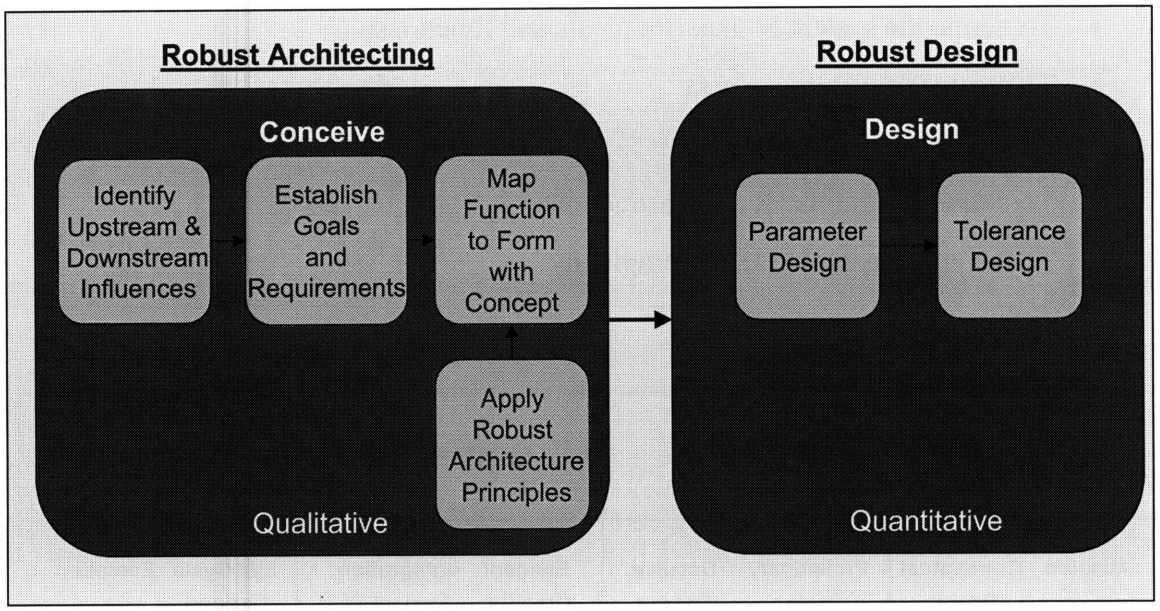


Figure 53. Robust Architecting Methodology

Robust architecting methodology incorporates several frameworks, the first of which is Crawley's (2005) system architecture framework for upstream influences. Looking at the system architecture framework in Figure 54, we see many factors affecting the goals and a few affecting the architecture directly. The architect must elicit needs from the customers and stakeholders to begin the goal setting process. Once the customer needs are established, the architect can synthesize this information with the other upstream influences, such as corporate strategy and regulations, to begin establishing goals with a high probability of being achieved.

The next framework introduced is Crawley's (2005) system architecture framework for downstream influences. Before an architecture is generated, the downstream influences must be taken into consideration as they will place additional requirements and constraints on the form. The downstream influences include implementation, operation, evolution and design.

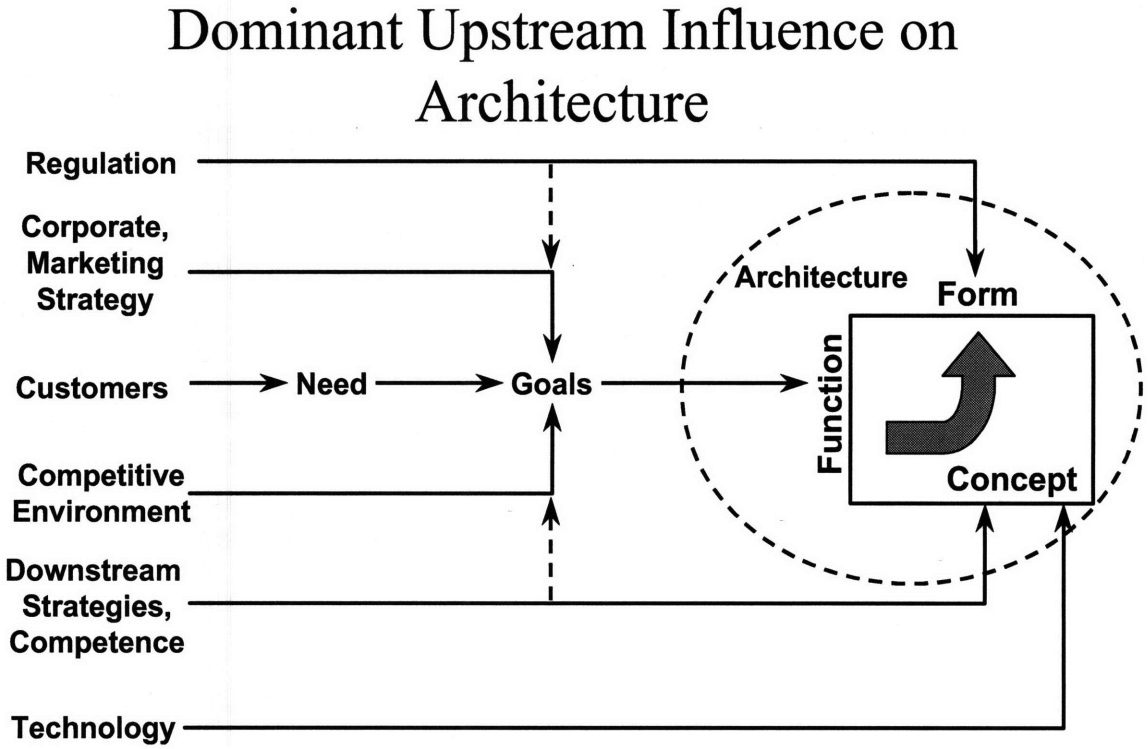


Figure 54. System Architecture Framework for Upstream Influences (Crawley, 2005)

Implementation refers to the production of the system and takes into account the manufacture and assembly. This is a good early opportunity to consider any high-level production issues that may affect robustness or variation. Operational influences are items such as the use of the system, the training of the user and life cycle cost. The evolution influence reminds us that architectures may need to change in future years to stay competitive or incorporate new technologies. The architecture must be designed to bring the product system to fruition. To this end, the architect must establish the architecture so the engineering design phases are simplified, manageable and efficient.

Once the upstream and downstream influences are fully considered, the architect should compile a comprehensive list of goals and constraints for the system architecture. These goals must be complete and attainable, and defined to provide significant value to the stakeholders at an acceptable cost. At this point, the goals can be translated into functional requirements for the top-level (i.e. zero level) system architecture. The

requirements should be in solution neutral form and answer the question "what" not "how".

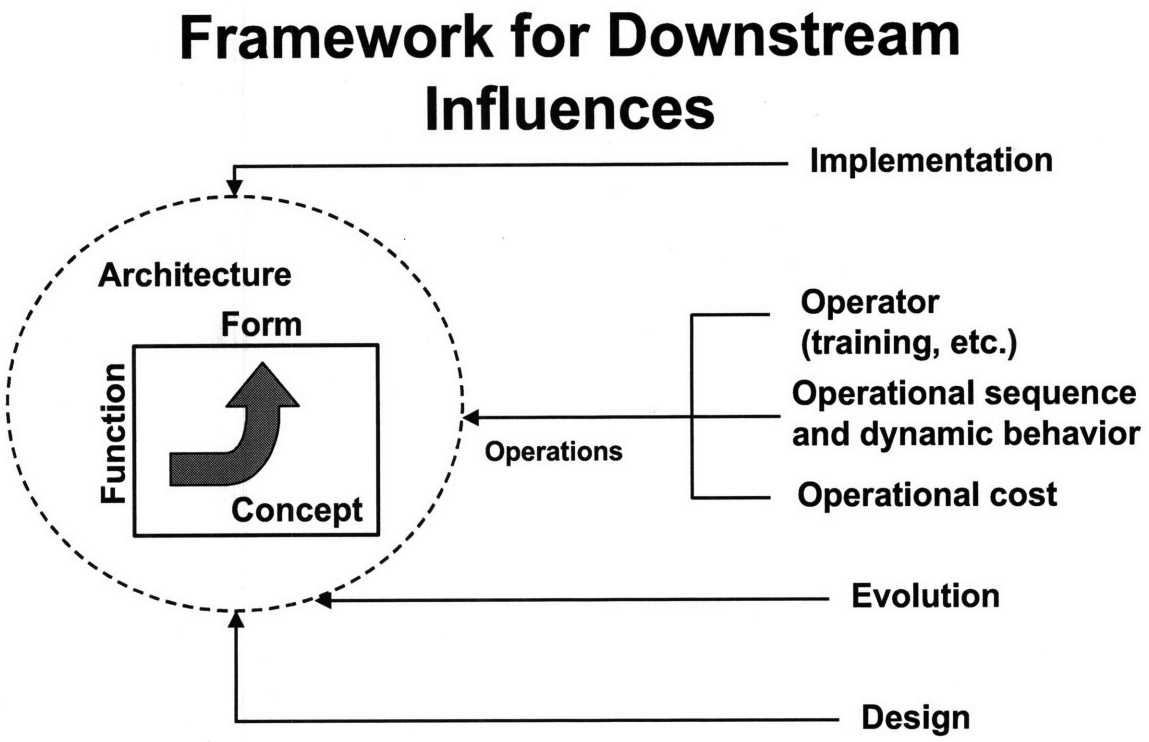


Figure 55. System Architecture Framework for Downstream Influences (Crawley, 2005)

The next step in the architecting process is to generate concepts that can be used to guide the mapping of function to form. There are numerous techniques available to aid in concept generation with some examples being TRIZ, mindmapping and brainstorming. TRIZ is especially helpful with its laws and lines of technological evolution that drive a system towards ideality.

The robust system architecture principle framework should be applied during concept generation to guide the process. The principles can be used to develop new concepts, modify existing concepts or eliminate ineffective ones. By introducing the principles at this point in the PDP, we give the architect the greatest opportunity to achieve robustness. For example, the simplicity principle guides the architect to eliminate excessively complex concepts that have a low likelihood of being robust. Other robust concept

design approaches may be used here to complement the framework including "robustness inventions" (Frey and Jugulum, 2006) and failure-mode avoidance (Clausing and Frey, 2005). The principle framework can also be applied in tandem with Pugh concept selection to systematically expand the concept space and then down-select to a single system concept.

After the generic concept is identified and decomposed to the second level, the architect begins flowing the system level goals to the subsystem level. With the goals established, the functional requirements are decomposed into subsystem requirements. The concept specifies the actual function and form used to meet the requirements. In combination, the function, form and concept constitute the architecture of the complex system. The above process is repeated, cascading the goals, requirements, concept and form to lower and lower subsystem levels until the complete architecture is established. Throughout the process, the robust principles framework is continually applied to ensure robust architecture at each level of decomposition.

The development of robust architecture from the last step signals the end of the conceive phase, and the beginning of the design phase of the PDP. At this point, we switch from robust architecting to robust design methodology and continue the development of a high quality system.

7.3 Design for Six Sigma

In chapter two, I gave a short overview of six sigma and its two constituent processes DMAIC and DMADOV. DMAIC, as mentioned earlier, is focused on improving existing products and processes in the business. Within the product development process, we find the majority of DMAIC robustness-oriented activity taking place in the implement and operate phases. Realizing the benefits of moving the robustness activity forward in the development process, companies have begun using design for six sigma (DFSS) processes. DFSS is a quality process that incorporates robust design methodologies into the detailed design phase of product development. In contrast to DMAIC, DFSS takes a systems engineering approach to developing *new* products and

processes. One of the common implementations of DFSS is the DMADOV process. Figure 56 highlights the relative positioning of DMAIC and DMADOV in the product development process.

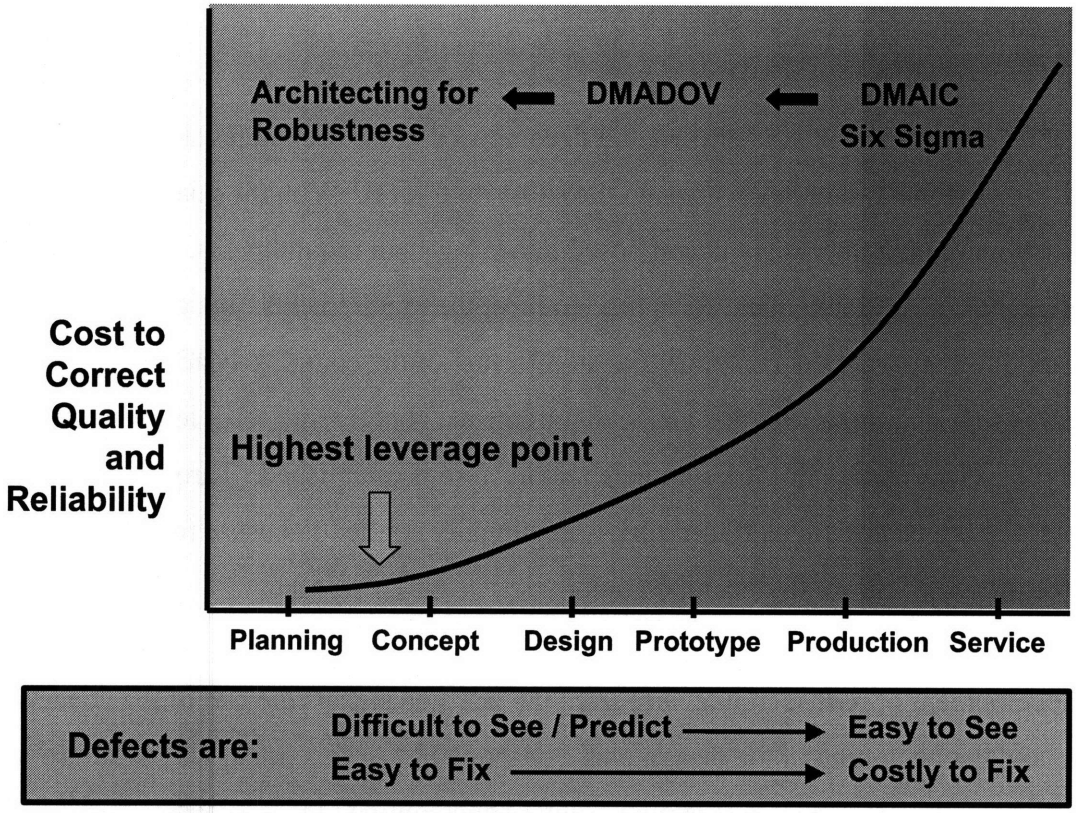


Figure 56. Incentive to Architect for Robustness (modified from Soman, 2004)

As this figure shows, there are many benefits to the business organization by addressing robustness at the design phase with DMADOV. As compared to DMAIC, the cost to correct quality and reliability issues is significantly less. While defects can be more difficult to predict at this stage, the net benefits make this a very profitable endeavor.

Referring back to Figure 56, we see that infusing robustness into the conceive phase of PDP offers the highest leverage of all. As described in section 7.2, this is exactly what the robust system architecting methodology accomplishes. Because the potential benefits to the business organization are large, it is proposed here that the system architecting methodology be integrated with the existing DMADOV process. By updating an existing

process, we create a product that is more easily adopted by the business world, which allows us to have a faster impact.

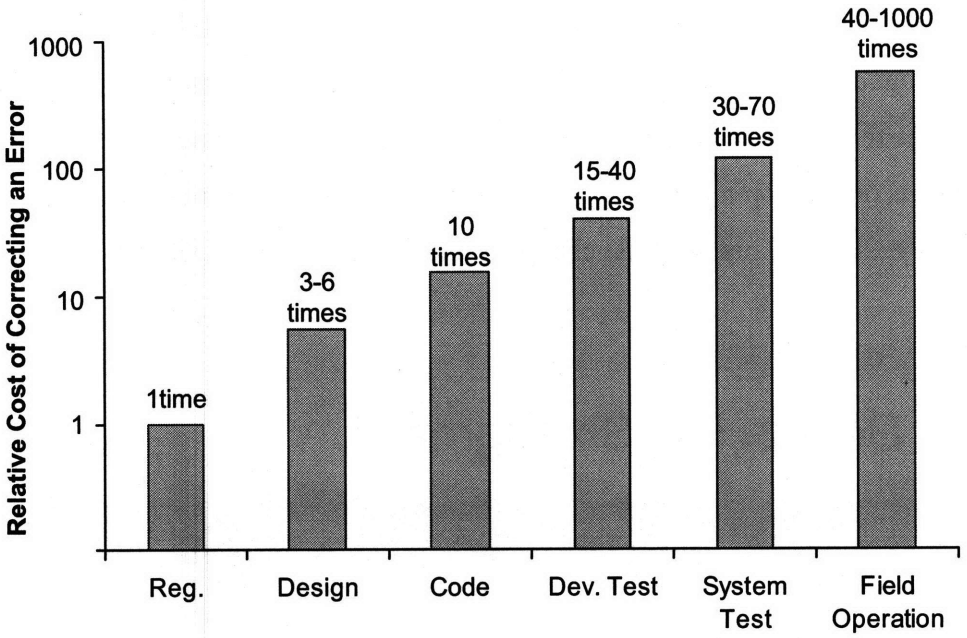


Figure 57. Relative Cost of Correcting an Error (Frey, 2005)

The steps for GE's DMADOV process are laid out in Figure 58. As can be seen, the DMADOV name is an acronym for the process steps: define, measure, analyze, develop, optimize and verify. Some implementations of DMADOV do not explicitly include the optimization phase, and use the shorter acronym DMADV.

In Figure 59, an updated design for six sigma DMADOV process is shown that integrates robust system architecting methodology as described in section 7.2. Several steps have been added before the define phase, and they reflect the inclusion of the upstream and downstream influences framework. The first step is the actual identification of the upstream and downstream influences. This information is then used to define the needs and subsequently to create a value delivery proposition that fulfills the needs.

The define step has several changes, with the existing two tasks "set quality goals" and "flow down requirements" being replaced with "set system level goals and functional

requirements." This change is necessary because the functional requirements cannot be decomposed to all the subsystem levels until the concept has been established.

No changes are proposed for the measure step, but one thought is offered for consideration. Taguchi has claimed that time should not be spent making precise assessments of the reliability of current products, but instead on improving the robustness of fielded products or developing entirely new products with enhanced robustness (Clausing and Fey, 2004). This potential change in the process is left as an item for future work.

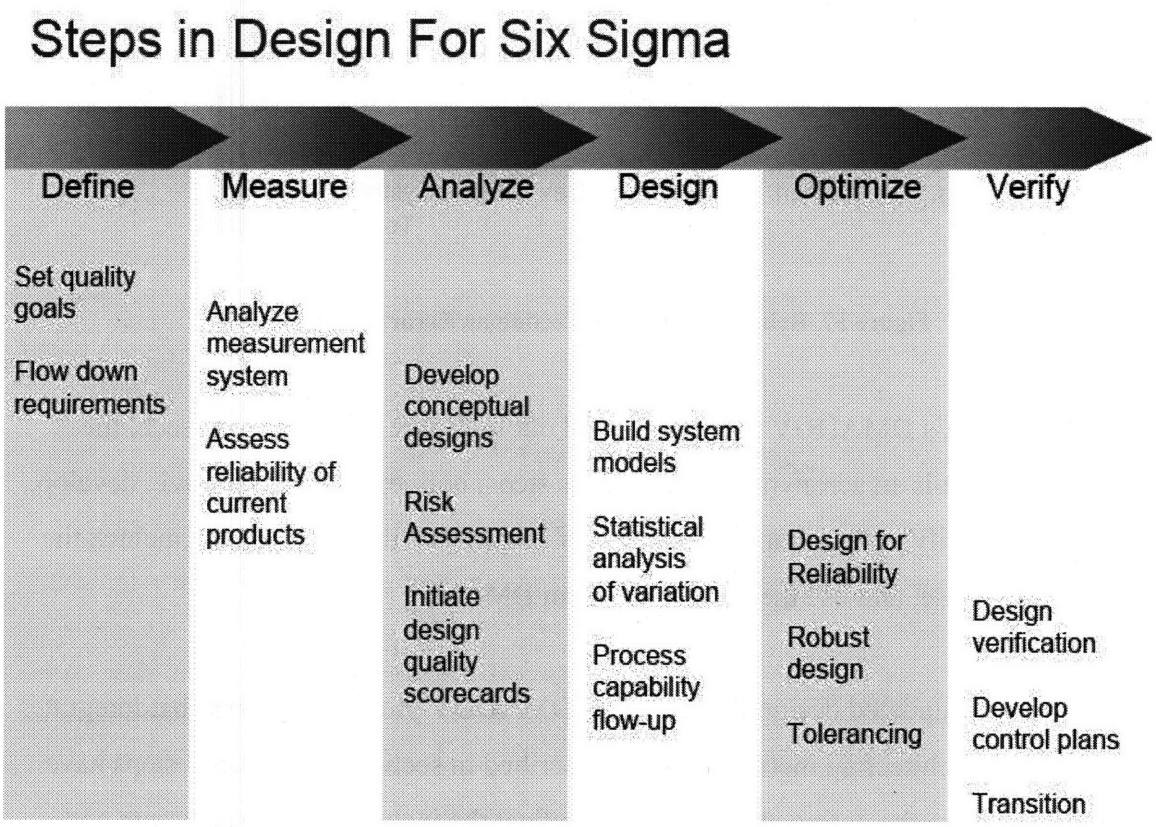


Figure 58. GE Design for Six Sigma Process (Soman, 2004)

During the analyze step, the robust principles framework be applied. The framework should be using during the concept generation phase to guide the architect, and additionally during the mapping of function to form at all levels of the architecture. The principles not only guide the process, they also act as an early evaluation tool for the

architecture. As mentioned in section 7.2, other techniques such as robustness inventions and failure mode avoidance techniques can also be leveraged. The last change in the analyze step is to cascade down the functional requirements to the subsystem levels as these levels of decomposition are resolved by the concept.

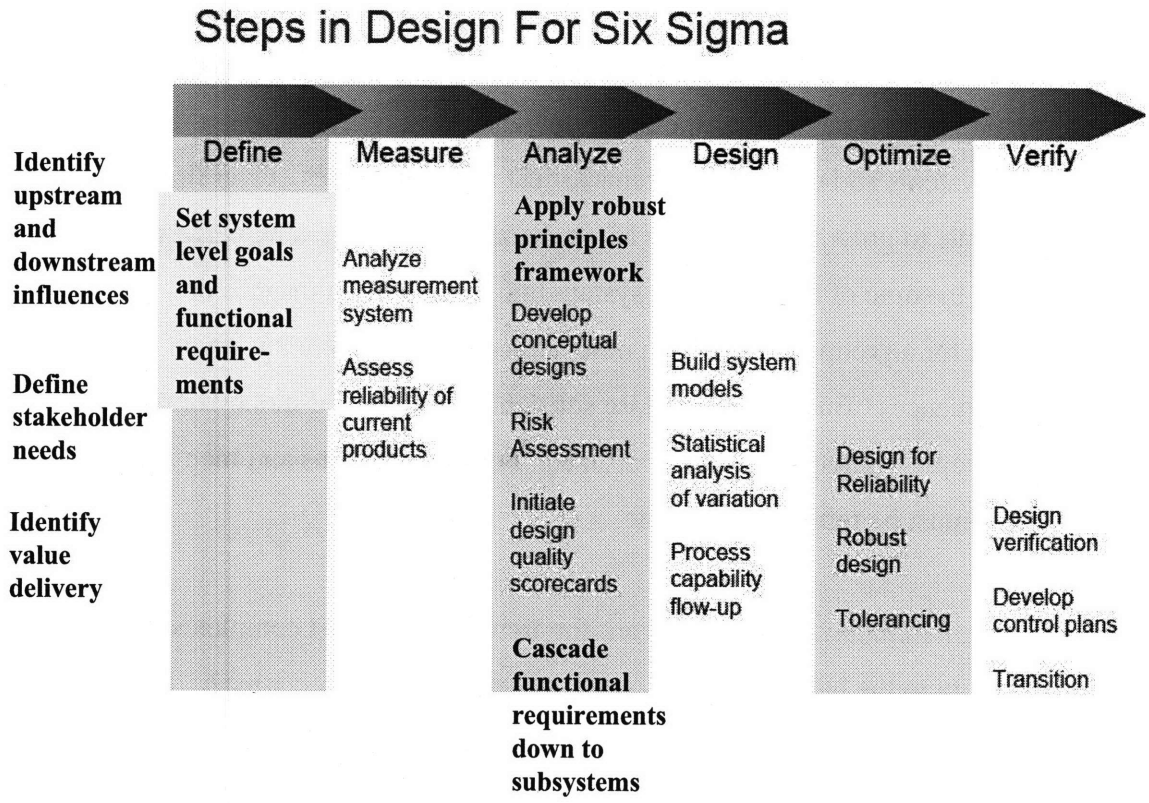


Figure 59. Updated DFSS Process Incorporating Robust Architecting Methodology

In this chapter, we examined the product development process and identified the role of the architect within this process. After this overview, we pieced together a robust architecting methodology that leverages the robust principle framework from chapter six. We next delved into the DFSS process and reviewed the cost and quality benefits of applying robustness methodologies earlier in the PDP. Lastly, a new DFSS process was proposed that addresses robustness at the system architecture level for maximum benefit to the business.

8. Conclusions and Future Work

"It is not necessary to change. Survival is not mandatory."
- W. Edwards Denning

Complex systems are used by each of us to improve our effectiveness, warm our houses, keep us healthy and transport us around the world. These are our infrastructure systems, our information technology systems, our national defense systems, our medical systems, our business systems and our space systems as just a few examples. Over time, these systems continue to grow in complexity as we demand ever-increasing performance and functionality. In some of these cases, a system failure may be a nuisance or inconvenience for a group of people. In other cases, system failure has much more serious consequences. Businesses can lose substantial amounts of money or more importantly people's safety may be jeopardized. For these reasons and many others, these systems must be robust and reliable.

In this chapter, I summarize the framework for architecting robust complex systems and discuss its utility to the business organization, system architect or engineer. I also review the context of the thesis and make note of any limitations.

8.1 Conclusions

Robustness and reliability, as mentioned above, are key properties for complex systems. In order to be reliable, the system must be robust, and mistakes must be avoided in the design and use of the system. Traditionally, robustness has been designed into complex systems using robust design methodology. Robust design methodology has proven to be very effective, but it operates on the later stages of the product development process. Recent research has sought to move the robustness focus forward in the product development process to the concept design phase because it offers substantially higher leverage. At the concept design phase, both the cost committed and the cost to correct quality mistakes are very low. The research in this thesis most closely complements these robust concept design methodologies that have arisen.

The objective for this research was primarily to develop a framework for architecting robust complex systems. I sought to accomplish this by identifying principles that have historically improved architectural robustness. In support of this objective, the architectural evolution of the jet engine was explored to see if any insights could be gleaned. Jet engines were selected as the system of study for several reasons. First, they are extremely complex systems that integrate multiple domains, including mechanical, electrical and informational. Second, since their inception they have experienced numerous architectural changes and at the same time have demonstrated exponentially increasing reliability.

The research for this work was performed over the course of approximately six months while the primary author was a student in the System Design and Management program at the Massachusetts Institute of Technology. Concurrently, the primary author was also working full time as an Engineering Manager at GE Aviation in Cincinnati, OH. One benefit of this arrangement was the author having access to jet engine hardware from many different periods in jet engine history.

Using primarily qualitative and some empirical data, it was observed that certain architectural changes did have a positive effect on robustness. The observations based on empirical data alone, however, must be tempered because the data is the net result of many changes. For example, jet engine reliability has continuously improved over the past 50 years, but this is attributable to better modeling techniques, superior materials, improved pilot training, et cetera as well as architectural changes.

By retrospectively examining a large number of the architectural innovations and changes in jet engine history, a set of principles was identified that improved architectural robustness. The attributes of robustness that are improved by the principles have been categorized using the P-diagram, and this categorization includes type I and type II robustness. The principles of robust architecture framework can be used by the architect or engineer to guide the development of robust system architecture.

The research next showed how the robust principles framework could be applied in the larger context of the system architecting process. To generate a more immediate benefit for the business community, the design for six sigma DMADOV process has been updated to incorporate the robust system architecting methodology and the robust principles framework. The robust architecting methodology combines synergistically with robust design to form a single development process that adds robustness activity at the conceptual design stage. While additional studies should be performed, this research suggests the updated process can be used by the business to achieve higher levels of robustness at lower cost.

8.2 Future Work

In the course of completing this thesis, I have identified limitations as well as additional research opportunities that should be further explored. The principles in the architecting for robustness framework have been drawn from, or verified by, an evolutionary study of jet engines. It would be useful to study other industries and technology evolutions to see if the framework is equally valid in those cases and to gauge effectiveness. The secondary benefit is that other domains may have reliability and robustness data more abundantly available that could supplement the empirical data used in this thesis. As mentioned in the literature search, the aircraft engine business is exceptionally competitive and as a result the companies keep most information proprietary. Furthermore, other industries should be examined to see if additional principles could be identified. It is suggested that a diverse set of domains be researched including additional mechanical domains, information systems, electrical systems and natural systems. Studying varying levels of complexity would also be beneficial. The architectural principles generated herein are not intended to be a comprehensive set, but rather a starting point.

A number of questions still need to be answered with the set of principles that have been established. Some effort was made to capture the "top-level" principles as explained in chapter six. In one case, it was determined that the containment and segmentation principles fall under the more powerful modularity principle category. Research should

be conducted to determine if there is a hierarchy to the principles and if so, what form it would take. The laws and lines of evolution introduced by TRIZ might be a guide in accomplishing this objective.

While the principles of robust system architecture have been covered in this thesis, another area of interest lies in researching the principles for robust architecting. A few possible principles of this type were mentioned at the beginning of chapter six without substantiation. These include ideality, domain shift, common language, knowledge base, physics and flow, benchmarking and real options. A few of these principles, such as ideality and real options, have been the subject of case studies. However, additional research should be conducted on all of the proposed principles in the context of the robustness framework suggested in this thesis.

The P-diagram is a valuable tool in robust design and this categorization was applied to the robust architecture principles framework. It may be equally applicable and effective at categorizing the principles governing the robust architecting process.

The robust architecture principles have been verified with jet engine architecture and reliability, but the framework does not dictate the extent to which each principle should be applied. It is conceivable that the extent will vary based on domain and perhaps complexity level. These items might be further researched as well as any possible detrimental effects or negative interactions with other principles.

A final area for potential research is the management and dispersion of innovation with respect to new robust design methodologies. We have walked through the implementation of the architecting for robustness framework in the product development process and highlighted the benefits to the organization. The extension of robust design to the architecture and concept design phases, however, will present challenges to the business. Organizational leaders will wrestle with a transition from highly quantifiable techniques, such as parameter design, to the qualitative approach that this author and other robust conceptual design researchers recommend. Research should be conducted

on the best approaches for dispersing the new robust design framework to maximize both effectiveness and traction.

Bibliography

Alexander, Christopher, et al. A Pattern Language: Towns, Buildings, Construction. New York: Oxford University Press, 1977.

Altshuller, G. S. Creativity as an Exact Science. Amsterdam: Gordon and Breach Publishers Inc., 1984.

Altshuller, Genrich. The Innovation Algorithm: TRIZ Systematic Innovation and Technical Creativity. Worchester, MA: Technical Innovation Center, Inc., 2000.

Anderson, John D. Introduction to Flight. 3rd. New York, NY: McGraw-Hill, Inc., 1989.

"Antonov News." Antonov. 2006. Antonov ASTC. 3 Nov 2006 <http://www.antonov.com>.

Baldwin, Carliss Y., and Kim B. Clark. Design Rules, Vol. 1: The Power of Modularity. Cambridge: The MIT Press, 2000.

Ballal, Dilip R. and Joseph Zelina. "Progress in Aeroengine Technology (1939-2003)." Journal of Aircraft 41(2004): 43-50.

Benzakein, J. J. "The Future of the Jet Engine." University of Dayton. Dayton, OH. 25 May 2006.

Blanton, John. "History of the Jet Engine: GE Aircraft Engines Perspective." GE Aircraft Engines. Evendale Plant, Evendale, OH. 05 Sep 2004.

Blanton, John. Personal interview. 12 September 2006.

Boeing, "Commercial Airplanes." 20 Aug 2006. <http://www.boeing.com/commercial/>.

Boothroyd, G., and P. Dewhurst. Product Design for Assembly. Wakefield, RI: Boothroyd & Dewhurst, Inc., 1987.

Carlson, J. M., and John Doyle. "Highly Optimized Tolerance: Robustness and Design in Complex Systems." Physical Review Letters 84(2000): 2529-2532.

Century of Flight, "The Jet Age." 6 Aug 2006 <http://www.century-of-flight.net>.

Cessna, "Our Aircraft." Cessna.com. 5 Dec 2006. Cessna, a Textron Company. 22 Dec 2006 <http://cessna.com/aircraft/>.

"Chapter 10.5.2. Technology Maturity and Technology Readiness Assessments." Defense Acquisition Guidebook. Version 1.6 (07/24/2006). U. S. Department of Defense. 26 Aug 2006 <http://akss.dau.mil/dag/welcome.asp>.

Chen, W., et al. "A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors." ASME Journal of Mechanical Design 118(1996): 478-485.

Christensen, Clayton M. The Inventor's Dilemma: The Revolutionary Book that Will Change the Way You Do Business. New York: Collins Business Essentials, 2003.

Clausing, Don and Daniel D. Frey. "Improving system reliability by failure-mode avoidance including four concept design strategies ." Systems Engineering 8(2005): 245- 261.

Clausing, Don. Personal Interview. 19 July 2006.

Crawley, Edward, Daniel Frey, and Benjamin Koo. "Robust Design at a System Architecture Level – A Proposal to BMW". Unpublished proposal, 2005.

Crawley, Edward, et al. Engineering Systems Monograph: The Influence of Architecture in Engineering Systems. Proceedings of the MIT Engineering Systems Symposium, 29-31 March. Cambridge, MA: MIT Engineering Systems Division (ESD), 2004.

Crawley, Edward. "ESD.34 System Architecture Lectures." MIT Engineering Systems Division. Cambridge. Fall 2005.

Crawley, Edward. "ESD.34 System Architecture Lectures." MIT Engineering Systems Division. Cambridge. Fall 2006.

Creveling, C. M., J. L. Slutsky and D. Antis, Jr. Design for Six Sigma in Technology and Product Development. Upper Saddle River, NJ: Prentice Hall PTR, 2002.

Daita, Lakshmi. "P-Diagram." The Quality Portal. 12 July 2006. The Quality Portal. 12 July 2006 <http://thequalityportal.com/p_diagram.htm>.

DeNardo, Adrian. United States. Air Force Research Laboratory.Engines for Change. Dayton, OH: 2003.

Doerflein, Thomas, Stephen Wilton, et al. Method and apparatus for supporting rotor assemblies during unbalances. General Electric Company, assignee. Patent 6,783,319. 31 Aug 2004.

Donlan, Charles J. United States. NACA. NACA RM L541716 An Assessment of the Airplane Drag Problem at Transonic and Supersonic Speeds. 1954.

Dori, Dov. Object-Process Methodology: A Holistic Systems Paradigm. New York, NY: Springer-Verlag, 2002.

Doyle, John, and J. M. Carlson. "Power Laws, Highly Optimized Tolerance, and Generalized Source Coding." Physical Review Letters 84(2000): 5656-5659.

DRS Technologies, "Military Parts Reinvention Network." DRS Technologies. 2 Dec 2006 <http://www.milparts.net/index.html>.

Ducharme, Eric. Personal interview. 26 Oct 2006.

Fabrycky, W. J. and B. S. Blanchard. Life-cycle Cost and Economic Analysis. Englewood Cliffs, NJ: Prentice Hall, 1991.

Federal Aviation Administration. 8 Nov 2006 <http://www.faa.com>.

Fey, Victor, and Eugene Rivin. Innovation on Demand: New Product Development Using TRIZ. New York: Cambridge University Press, 2005.

Fey, Victor. Personal Interview. 13 August 2006.

Foster, John. "Gas Turbines." Aircraft Engine Historical Society. 30 Oct 2006. Aircraft Engine Historical Society, Inc. 30 Nov 2006 <http://www.enginehistory.org/gas_turbines.htm>.

Fowlkes, William Y. and Clyde M. Creveling. Engineering Methods for Robust Product Design: Using Taguchi Methods in Technology and Product Development. Reading, MA: Addison-Wesley Publishing Co., 1995.

Frey, Daniel D. "ESD.937 Systems Engineering Lectures." MIT Engineering Systems Division. Cambridge. Summer 2005.

Frey, D. D., and R. Jugulum. "Robustness Through Invention." Journal of the Quality Engineering Society (Japan) 12(2004): 116-122.

Frey, D. D., J. Palladino, J. P. Sullivan, and M. Atherton. "Part Count and Design of Robust Systems." Submitted to Journal of Systems Engineering (INCOSE). 2006.

Fricke, Ernst, and Schulz. "Design for Changeability (DfC): Principles to Enable Changes in Systems Throughout Their Entire Lifecycle." Systems Engineering 8(2005): 342-359.

Fry, Ronald S. "A Century of Ramjet Propulsion Technology Evolution." AIAA Journal of Propulsion and Power 20(2004): 27-58.

Garg, Sanjay. Controls and Health Management Technologies for Intelligent Aerospace Propulsion Systems. 42nd AIAA Aerospace Sciences Meeting and Exhibit. Reno, Nevada: AIAA, 2004.

GE Aviation. 2 Nov 2006 <http://www.geae.com/>.

GE, "GE90-115B: GE's Best-Ever New Jet Engine Entry Into Airline Service." GE Aviation. 17 July 2006. General Electric. 10 Dec 2006 <http://www.geae.com/aboutgeae/presscenter/ge90/ge90_20060717a.html>.

GE, Eight Decades of Progress: A Heritage of Aircraft Turbine Technology. Cincinnati, OH: GE Aircraft Engines, 1990.

"GE." What is Six Sigma? The Roadmap to Customer Impact. 15 July 2006. General Electric Company. 15 July 2006 <http://www.ge.com/sixsigma/SixSigma.pdf>

Gerez, Valerio. Method for enabling operation of an aircraft turbo-engine with rotor unbalance. Sciete National d'Etude et de Construction de Moteurs d'Aviation "Snecma", assignee. Patent 5,974,782. 2 Nov 1999.

Gleick, James. Chaos: Making a New Science. New York, NY: Viking Penguin Inc., 1987.

Gleick, James. Faster: The Acceleration of Everything. Australia: Warner Books, 1999.

Gunston, Bill. The Development of Jet and Turbine Aero Engines. Second Edition. Gloucestershire, United Kingdom: Sutton Publishing, 1997.

Hamel, Gary. Leading the Revolution. Boston: Harvard Business School Press, 2000.

Hari, A., and M. P. Weiss. CFMA - An Effective FMEA Tool for Analysis and Selection of the Concept for a New Product. Proceedings of the ASME Design Engineering Technical Conference, DTM. Las Vegas, NV: ASME, 1999.

Hawking, Stephen. The Universe in a Nutshell. United Kingdom: Bantam Dell Publishing, 2001.

Henderson, Rebecca M. The failure of established firms in the face of technical change: A study of photolithographic alignment equipment. Diss. Harvard University, 1988.

Henderson, Rebecca and Kim B. Clark. "Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms - Technology, Organizations, and Innovation." Administrative Science Quarterly (1990).

Hispano-Suiza, "Digital Engine Control." Hispano-Suiza, SAFRAN Group. 14 Sep 2006 <http://www.hispano-suiza-sa.com/article.php3?id_article=62&lang=en>.

Horowitz, Ron. Creative Design Methodology and the SIT Method. 1997 ASME Design Engineering Technical Conference Sept 14-17. American Society of Mechanical Engineers, 1997.

How to Achieve a Higher Plane. CFM: The Power of Flight. 05 Oct 2006 <http://www.cfm56.com/>.

INCOSE, "Systems Architecture Working Group." International Council on Systems Engineering. 29 June 2006. INCOSE. 27 Dec 2006 <http://www.incose.org/practice/techactivities/wg/sysarch/>.

Ishii, Kosuke, and Whitfield Fowler. "Design for System Reliability Research Update: GEAE and Stanford University." GEH DFSR Practices. 8 Nov 2005.

Jane's. "Jane's Aero-Engines." Jane's: Intelligence and Insight You Can Trust.. 13 Jan 2007 <http://jae.janes.com/public/jae/index.shtml>.

Jaw, L. C., and S. Garg. Propulsion Control Technology Development in the U. S. – A Historical Perspective. Proceedings of the International Symposium on Air-breathing Engines. Cleveland, OH: ISABE 2003-1186, 2003.

Jen, Erica. "Stable or Robust? What's the difference?" Complexity 8(2003): 12-18.

Jen, Erica. "Working Definitions of Robustness RS-2001-009." Robustness in Natural, Engineering, and Social Systems. 10 Oct 2001. Santa Fe Institute. 7 Aug 2006 <http://discuss.santafe.edu/robustness/>.

Joyce, David and Jeanne Rosario. "Investing for Sustained Technology Leadership." Technology Leadership Meeting. GE Analyst Meeting, Fairfield, CT. June 2002.

Jugulum, R. and D. D. Frey. "Toward a Taxonomy of Concept Designs for Improved Robustness." Accepted to Journal of Engineering Design. 2006.

Kastl, John, Randy Marinus, et al. Fan decoupling fuse. General Electric Company, assignee. Patent 6,402,469. 11 Jun 2002.

Koff, Bernard L. "Gas Turbine Technology Evolution: A Designer's Perspective." AIAA International Air and Space Symposium and Exposition: The Next 100 Years. Dayton, OH. 14 July 2003.

Koff, Bernard L. "Spanning the Globe with Jet Propulsion." AIAA Annual Meeting and Exhibit. Arlington, VA. 30 Apr 1991.

Kurzweil, Ray. The Singularity is Near: When Humans Transend Technology. New York: Viking, 2005.

Lippencott, Harvey H. "P&W Enters the Jet Age." Casting About 1985.

Loftin, Laurence K. "Quest for Performance: The Evolution of Modern Aircraft." NASA History Division. 2006. National Aeronautics and Space Administration. 12 May 2006 <http://www.hq.nasa.gov/office/pao/History/SP-468/contents.htm>.

Maclin, Harvey and Clay Haubert. Fifty Years Down - Fifty Years to Go. Proceedings of the AIAA/ICAS International Air and Space Symposium and Exposition: The Next 100 Years 14-17 July 2003. Dayton, OH: AIAA, 2003.

Maier, Mark W., and Eberhardt Rechtin. The Art of System Architecting. Second Edition. New York: CRC Press, 2002.

Meier, Nathan. "Civil Turbojet / Turbofan Specifications." Jet Engine Specification Database. 03 Apr 2005. 13 Jan 2007 <http://www.jet-engine.net/civtfspec.html>.

Merriam-Webster, Inc., "Merriam-Webster Online Search." Merriam-Webster Online. Merriam-Webster, Inc. 22 Jun 2006 <http://www.m-w.com/>.

Meyer, Marc H., and Alvin P. Lehnerd. The Power of Product Platforms. New York: The Free Press, 1997.

Morrison, Brad. "ESD.74 System Dynamics for Engineers Class Lectures." MIT Sloan School of Management. Cambridge. Summer 2006.

"Motorola University." About Motorola University: The Inventors of Six Sigma . 12 November 2006. Motorola. 12 Nov 2006 <http://www.motorola.com/content.jsp?globalObjectId=3079>.

Murman, Earll, et al. Lean Enterprise Value: Insights from MIT's Lean Aerospace Initiative . New York: Palgrave McMillan, 2002.

Murray, Charles J. "Driverless Vehicles Take on the Desert." Design News 05 Sep 2005: 67-74.

Petroski, Henry. To Engineer is Human: The Role of Failure in Successful Design. New York: St. Martin's Press, 1985.

Phadke, Madhav S. Quality Engineering Using Robust Design. Englewood Cliffs, NJ: Prentice Hall PTR, 1989.

Pratt and Whitney: A United Technologies Company. 06 Nov 2006. Pratt and Whitney. 8 Nov 2006 <http://www.pratt-whitney.com/>.

Rechtin, E. Systems Architecting, Creating & Building Complex Systems. Englewood Cliffs, NJ: Prentice-Hall, 1991.

Rolls-Royce. Rolls-Royce. 20 Oct 2006 <http://www.rolls-royce.com>.

Sahal, Devendra. "Technological Guideposts and Innovation Avenues." Research Policy 14(1985): 61-82.

Schulz, Armin, and Ernst Fricke. Incorporating Flexibility, Agility, Robustness, and Adaptability within the Design of Integrated Systems - Key to Success? 18th Digital Avionics Systems Conference. St. Louis, MO: IEEE, 1999.

Simon, Herbert (2000): Can There Be a Science of Complex Systems? Unifying Themes in Complex Systems proceedings of the First International Conference on Complex Systems 1997, 3-14. Perseus Books Group (CCC).

Slagle, Jason. "Remote Diagnostics and the GE90-115 Engine." Report for MIT Innovation in Marketing Class. 27 Mar 2005.

Slagle, Jason, Aspi Wadia, et al. Booster Compressor Deicer. General Electric Company, assignee. Patent 6,561,760. 13 May 2003.

Smith, Jonathan, and John Clarkson. "Design Concept Modeling to Improve Reliability." Journal of Engineering Design 16(2005): 473-492.

Soman, Narendra. "Role of Monte Carlo Simulation in Design for Six Sigma (DFSS)." 2004 Crystal Ball User Conference General Electric Global Research. Marriott Denver City Center, Denver, CO. 17 June 2004.

St. Peter, James. The History of Aircraft Gas Turbine Engine Development in the United States: A Tradition of Excellence. first edition. New York: American Society if Mechanical Engineers, 2000.

Sterman, John D. Business Dynamics: Systems Thinking and Modeling for a Complex World. Boston: Irwin McGraw-Hill, 2000.

Suh, N. P. The Principle of Design. New York: Oxford University Press, 1990.

Taguchi, G. "The Development of Quality Engineering." The ASI Journal 1(1988): 5:29.

Taguchi, G. Taguchi on Robust Technology Development. New York, NY: ASME Press, 1993.

Ulrich, Carl, and Steven Eppinger. Product Design and Development. Third Edition. Columbus: McGraw-Hill/Irwin, 2004.

Ulrich, Karl. "The Role of Product Architecture in the Manufacturing Firm." Research Policy 24(1995): 419- 440.

United States. Office of Technology Assessment.Safe Skies for Tomorrow PB89-114318. Washington, DC: U.S. Government Printing Office, 1988.

Utterback, James. "15.365J Disruptive Technologies: Predator or Prey?". MIT Sloan School of Management. Cambridge. Spring 2006.

Utterback, James M. Mastering the Dynamics of Innovation. Boston: Harvard Business Press, 1996.

von Neumann, John. "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." Automata Studies. Ed. C. Shannon and J. McCarthy. Princeton, NJ: Princeton University Press, 1956.

Whittle, Ian. "The Early Development of the Turbojet." AIAA. General Electric Aircraft Engines Headquarters, Evendale, OH. 29 April 2006.

Wiggs, Gene. Personal Interview. 24 August 2006.

Wisman, M. W., and T. Guo. An Investigation of Life Extending Control Techniques for Gas Turbine Engines. Proceedings of the american Control Conference June 25-27. Arlington, VA: 2001.

Womack, James P., and Daniel T. Jones. Lean Thinking: Banish Waste and Create Wealth in Your Corporation. New York: Simon and Schuster Adult Publishing Group, 2003.

Wu, C. F., and M. Hamada. Experiments: Planning, Analysis, and Parameter Design Optimization. NY: Wiley and Sons, Inc., 2000.

Younossi, Obaid, et al. "Military Jet Engine Acquisition: Technology Basics and Cost-Estimating Methodology." RAND. (2002).

Appendix A. Technology Readiness Levels

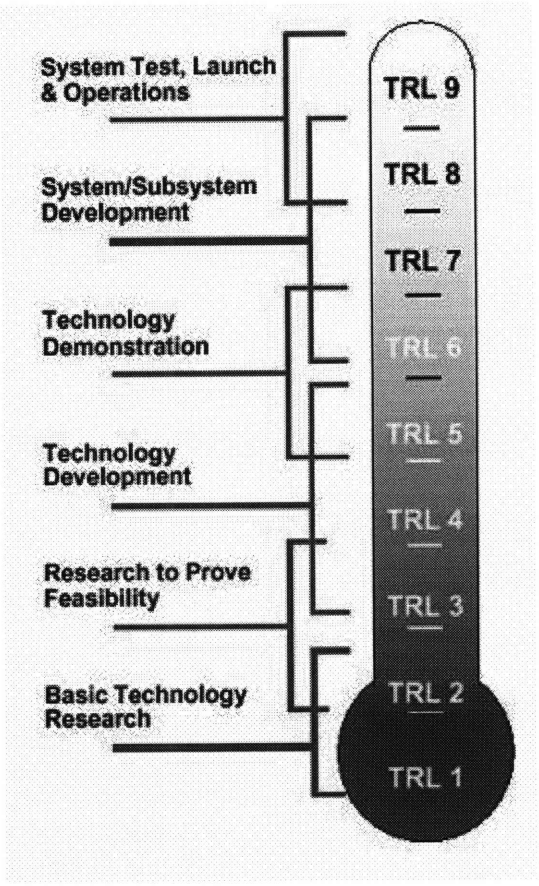


Figure 58. Technology Readiness Levels (NASA, 2006)

Table 11. Description of Technology Readiness Levels (DoD, 2006)

Technology Readiness Level	Description
1. Basic principles observed and reported.	Lowest level of technology readiness. Scientific research begins to be translated into applied research and development. Examples might include paper studies of a technology's basic properties.
2. Technology concept and/or application formulated.	Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.
3. Analytical and experimental critical function and/or characteristic proof of concept.	Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.

4. Component and/or breadboard validation in laboratory environment.	Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared to the eventual system. Examples include integration of "ad hoc" hardware in the laboratory.
5. Component and/or breadboard validation in relevant environment.	Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so it can be tested in a simulated environment. Examples include "high fidelity" laboratory integration of components.
6. System/subsystem model or prototype demonstration in a relevant environment.	Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high-fidelity laboratory environment or in simulated operational environment.
7. System prototype demonstration in an operational environment.	Prototype near, or at, planned operational system. Represents a major step up from TRL 6, requiring demonstration of an actual system prototype in an operational environment such as an aircraft, vehicle, or space. Examples include testing the prototype in a test bed aircraft.
8. Actual system completed and qualified through test and demonstration.	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation of the system in its intended weapon system to determine if it meets design specifications.
9. Actual system proven through successful mission operations.	Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation. Examples include using the system under operational mission conditions.

Appendix B. Invention Levels

Invention Levels (Fey and Riven, 2005)

1. A component intended for the task is used. No system conflicts are resolved. (32%)
2. Existing system is slightly modified. System conflicts are resolved by the transfer of a solution from a similar system. (45%)
3. System conflicts are resolved by radically changing or eliminating at least one principal system's component. Solution resides within one engineering discipline. (19%)
4. System conflicts are resolved and a new system is developed using interdisciplinary approaches. (<4%)

TechNav Process (Fey and Rivin, 2005)

Phase 1 – Analysis of the past and current system's evolution
Phase 2 – Determination of high-potential innovations
Phase 3 – Concept development
Phase 4 – Concept selection and technology plan

Technology plan – Investments in three key areas:

1. Product planning
2. Competitive analysis (present or potential know-how being developed by competitors, which is different from the company's technology).
3. Advanced technology (technology where a design is not yet defined, but thought to be the potentially key to future products or subsystems).

Appendix C. Jet Engines List

Jet Engine	Certification / In Service / Test Date
W.U.	1937
He S 1	1937
He S 3B	1939
W.1.	1941
I-A	1942
I-16	1943
TG-100/T31	1943
J33	1944
I-40 (J-33)	1945
J35	1946
J35	1946
J42	1948
J47	1948
J48	1949
T34	1950
T40	1950
J57 (JT3)	1951
J79	1955
J85	1956
J85	1956
T58	1957
CJ805-3	1958
CT58-100-1	1959
CJ805-23	1960
CJ805-3B	1960
CT58-100-2	1960
CJ610-1	1961
CJ610-2B	1961
CJ805-23B	1961
CJ805-23C	1961
CJ805-3A	1961
CT58-110-1	1961
J79	1962
PT6A-6	1963
T64	1963
CF700	1964
CF700-2B	1964
CJ610-4	1964
J58	1964
TF30	1964
J93	1964
CF700-2C	1965
CT58-140-1	1965
CT64-410-1	1965
CT64-610-1	1965
CT64-810-1	1965

T64-GE-10	1965
PT6A-6A	1965
PT6B-9	1965
PT6A-20	1965
CJ610-5	1966
CJ610-6	1966
Conway 505	1967
CT64-820-1	1967
CT64-820-2	1967
PT6A-6B	1967
PT6A-27	1967
CF700-2D	1968
CJ610-8	1968
CJ610-9	1968
CT58-110-2	1968
CT58-140-2	1968
CT64-630-1	1968
T64-GE-16	1968
PT6A-29	1968
JT9D	1968
VIPER MKS 521	1968
VIPER MKS 522	1968
VIPER MKS 526	1968
CF700-2D-2	1969
PT6A-28	1969
TF39	1969
JT9D-3A	1970
CF6-6D	1970
CF6-6	1971
CT64-630-1A	1971
JT9D-7	1971
PT6A-34	1971
CF6-6D1	1971
JT9D-7A	1972
JT9D-20	1972
CF6-50A	1972
CF6-50D	1972
F100	1972
TYNE 506	1972
TYNE 512	1972
TYNE 515	1972
VIPER MK 601-22	1972
TF34	1972
CT64-820-3	1973
PT6A-20A	1973
PT6A-20B	1973
PT6A-6/C20	1973
PT6A-36	1973
CF6-50C	1973
CF6-50E	1973

CF6-6H	1973
CF6-50H	1973
CT64-820-4	1974
JT9D-7H	1974
JT9D-7AH	1974
JT9D-7F	1974
PT6A-21	1974
CF6-50E1	1975
F107	1975
JT9D-7J	1976
PT6A-25	1976
PT6A-25A	1976
PT6A-34B	1976
CF6-50C1	1976
CJ610-8A	1977
PT6A-11	1977
PT6A-34AG	1977
CT7-1	1977
CF6-50CA	1977
CF6-45A	1977
CF6-45B	1977
PT6A-15AG	1978
CT7-2	1978
T700/T1C	1978
CF6-6D1A	1978
CF6-50C2	1978
CF6-50E2	1978
CF6-45A2	1978
CF6-45B2	1978
CFM56-2	1979
JT8D-209	1979
PT6A-11AG	1979
CF6-50C2B	1979
CF6-50E2B	1979
JT8D-217	1980
JT8D-217A	1981
T700/T2C	1981
CT7-2A	1981
CF6-6K	1981
F108	1981
CFM56-2B	1982
PT6B-35F	1982
PW2037	1983
CT7-2B	1983
CT7-5A	1983
CT7-7	1983
CT7-7E	1983
F404	1983
CFM56-3	1984
CFM56-3B	1984

PW115	1984
PW120	1984
PW120A	1984
CT7-5A1	1984
CT7-7A	1984
CT7-7E1	1984
CF6-6K2	1984
CF6-80C2	1985
CFM56-2A	1985
JT8D-219	1985
CT7-5A2	1985
CT7-2D	1985
F101	1985
F103 / CF6	1985
CFM56-3C	1986
JT8D-217C	1986
JT9D-20J	1986
PW4056	1986
PW4156	1986
PW4152	1986
PW118	1986
F110	1986
CFM56-5	1987
PW4052	1987
PW2037M	1987
PW2040	1987
PW118A	1987
PW4060	1988
PW4160	1988
PW4460	1988
PW4158	1988
F117-PW-100	1988
PW121	1988
PW123	1988
PW124	1988
PW124A	1988
PW125B	1988
CT7-9B	1988
CT7-9C	1988
CT7-6	1988
CT7-6A	1988
CF6-50C2D	1988
RB211-524G-19	1988
RB211-524G-19	1988
PW4050	1989
PW126A	1989
CT7-5A3	1989
CT7-7A1	1989
CT7-2D1	1989
RB211-524G2-19	1989

RB211-524G3-19	1989
RB211-524H-36	1989
RB211-524G2-19	1989
RB211-524G3-19	1989
RB211-524H-36	1989
CFM56-5A2	1990
CFM56-5A3	1990
PW4060A	1990
PW124B	1990
PT6A-25C	1990
RB211-524H2-19	1990
RB211-524H2-19	1990
CFM56-5C	1991
PW4156A	1991
PW305	1991
CFM56-5-A1/F	1992
PW4062	1992
PW4462	1992
PW4060C	1992
PW2240	1992
PW2337	1992
PW127	1992
CT7-9B1	1992
CT7-9B2	1992
CT7-9D	1992
PW119B	1993
PW305A	1993
PW305B	1993
F118	1993
GE90-76B	1995
GE90-85B	1995
PW4650	1995
PW2043	1995
PW2643	1995
PW2143	1995
PW121A	1995
PW123B	1995
PW123C	1995
PW123D	1995
PW123E	1995
PW127E	1995
RB211-877-17	1995
RB211-884-17	1995
RB211-875-17	1995
RB211-890-17	1995
CFM56-5A4	1996
CFM56-5A4/F	1996
CFM56-5A5	1996
CFM56-5A5/F	1996
GE90-90B	1996

GE90-77B	1996
PW530A	1996
PW118B	1996
PW119C	1996
PW127F	1996
RB211-524G2-T-19	1997
RB211-524G3-T-19	1997
RB211-524H-T-36	1997
RB211-524H2-T-19	1997
RB211-524G2-T-19	1997
RB211-524G3-T-19	1997
RB211-524H-T-36	1997
RB211-524H2-T-19	1997
RB211-892-17	1997
RB211-892B-17	1997
PW306A	1998
PW306B	1998
CT7-9C3	1998
CT7-9D2	1998
F414	1998
PW535A	1999
PW123AF	1999
PW127G	1999
RB211-895-17	1999
GE90-94B	2000
CT7-8	2000
250-B15A	2000
250-B15E	2000
250-B15G	2000
250-B17	2000
250-B17B	2000
250-B17C	2000
250-B17D	2000
250-B17E	2000
250-B17F	2000
250-B17F/1	2000
250-B17F/2	2000
PW4062A	2002
PW6122-1D	2002
PW6124	2002
PW306C	2002
PT6A-35	2002
GE90-110B1	2003
GE90-113B	2003
GE90-115B	2003
PW2040D	2003
PW2037D	2003
PW6122A	2004
PW6124A	2004
CT7-8A	2004

CT7-8A5	2004
CT7-8B	2004
CT7-8B5	2004
CT7-8E	2004
CT7-8E5	2004
CT7-8F	2004
CT7-8F5	2004
GP7200	2005
GEnx	*2007*

Note: For commercial engines, the listed date refers to engine certification. For military engines, the listed date refers to the in service date. If this information was not available, then the first flight test date is used. For technology demonstration engines such as the W.U., the date of first test is used.

Vita

Jason Slagle is currently an engineering manager at GE Aviation within the Advanced Technology department. In this role, he is responsible for the thermal design and the secondary flow system design of the GE90 product line, the world's largest and most powerful jet engines. Over the past decade, Jason has made significant contributions to the design and certification of the GE90-94B and GE90-115B engines. Additionally, Jason has helped define state-of-the-art technologies for propulsion systems, and has several patents / patents pending for these efforts.

At GE, Jason holds the controlled title of Senior Engineer. He is also a Senior Member of the American Institute of Aeronautics and Astronautics (AIAA). As an active member of the Dayton-Cincinnati section, he currently supports the group as Membership Chair.

Prior to these roles, Jason worked at Middle River Aircraft Systems (formerly Lockheed-Martin Aerostructures), Cessna Aircraft Company and Consol Energy.

Jason graduated from the Pennsylvania State University in 1997 with a Bachelors of Science degree in Aerospace Engineering. He then completed graduate studies in Mechanical Engineering at the Johns Hopkins University. Jason is currently a System Design and Management Fellow, and the present work completes the degree requirements for a Masters in Engineering and Management from Massachusetts Institute of Technology.