Chapter 14

System Integration

CONTENTS

This page intentionally left blank.

Chapter 14

System Integration

"Like a jigsaw puzzle: you have to make the pieces fit without getting out the scissors." – Dr. Karl Maurer – On translating Greek sentences [1]

14.1 Introduction

It may have happened late on Christmas Eve, or it could have been almost any other time. You bought or received something with the *some assembly required* caveat attached. You spent untold time trying to assemble something that was supposed to be easily assembled, but couldn't see how the pieces fit together. Even after humbling yourself to the point of referring to the instruction manual, the puzzle remained a mystery, and was only solved through trial and error, repeated calls to the manufacturer, or the help of a friend who had already gone through the test. Think of the people who had to design the diabolical contraption in the first place, having to find or design each part and make all of them work together.

System integration is the successful putting together of the various components, assemblies, and subsystems of a system and having them work together to perform what the system was intended to do. It follows the coding phase in the development life cycle, as shown in Figure 14-1, and is intertwined with the testing.

Figure 14-1 Integration's Place in the Development Life Cycle

While it may sound like the final assembly of the parts of a system, successful system integration involves almost every aspect of the project and reaches from the very beginning into and through the maintenance phase of a system's life cycle. Figure 14-2 shows the actual integration, where the system comes together, many of the results of successful integration, and several of the activities that are required for successful integration.

Successful system integration results from the proper implementation of project activities shown on the left side of Figure 14-2. The primary requirement and driver is systems engineering (see Chapter 13). When systems engineering is employed throughout the project, successful system integration is one of the primary outcomes. This includes requirements definition, functional analysis, synthesis, trade studies, careful interface definition, true life cycle integration, etc. In addition to the activities associated with systems engineering, correct employment of other activities such as configuration management, design, risk management, and testing are essential to ensuring all the pieces fit together during integration.

Testing goes hand in hand with integration because it is through testing that we determine whether or not the assemblies, subsystems, and systems operate as they should after integrating them. Testing and integration are part of the development process. If there were no testing, the results of system integration would remain an unknown until acceptance testing.

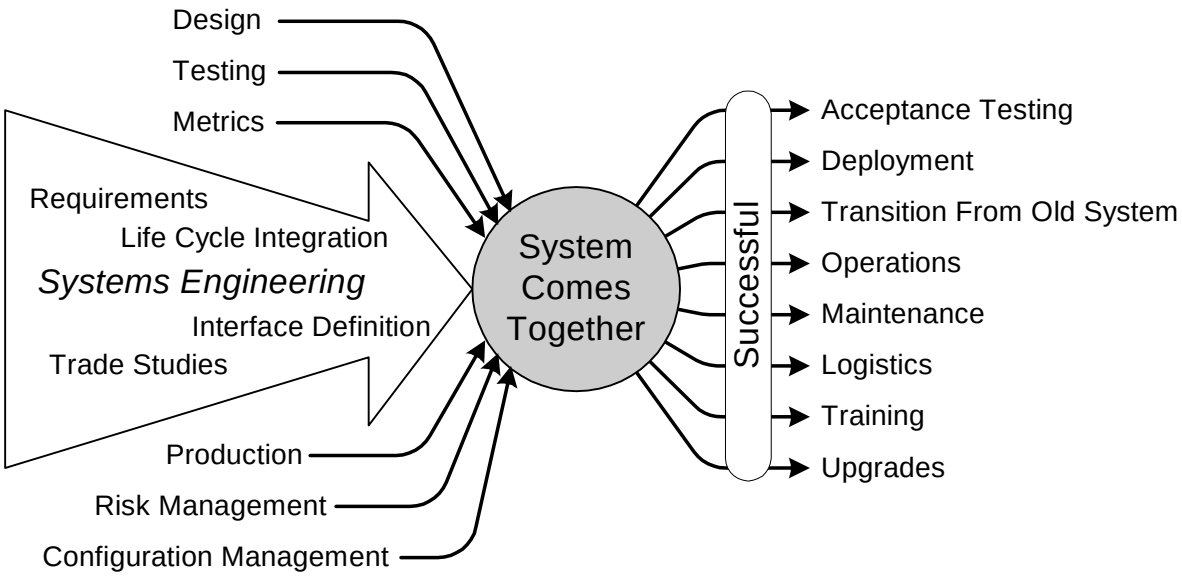

Figure 14-2 System Integration Inputs and Results

14.2 Process Description

As with almost everything else, system integration begins with planning. Because system integration is the logical consequence of systems engineering and other activities, the integration plan is usually a composite of those portions of other plans which pertain to it, as shown in Figure 14-3.

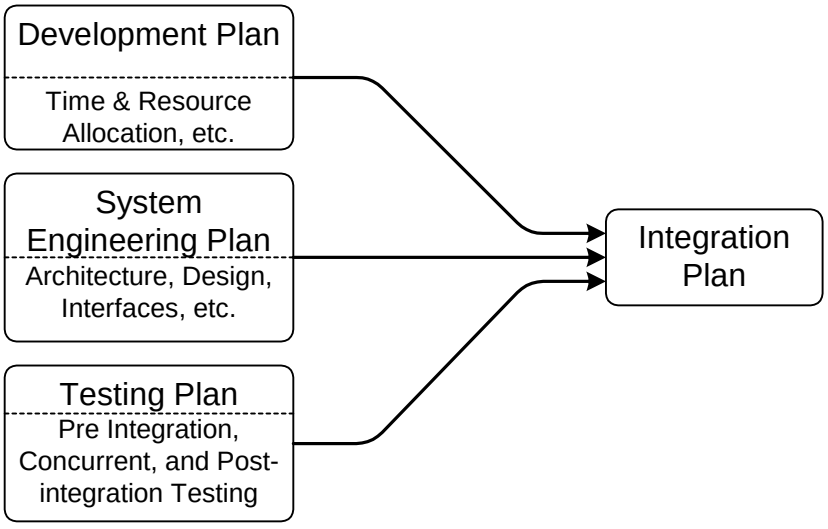

Figure 14-3 System Integration Planning

The integration process was shown in abbreviated form in conjunction with testing in Chapter 12, Figure 12-7. That drawing has been expanded in Figure 14-4 below to depict integration and testing more realistically as a series of integrations and tests. Testing is used to assure developers that the integrated product is properly functional. Integrated modules that fail testing are sent back for debugging and rework.

When tests fail, the test results are analyzed to determine the cause of failure. The components that make up the module being tested are then sent back for debugging and recoding by the developers. If there appears to be a problem with the design, and not with the coding, the module is sent back to design for resolution. When the problem appears to be solved, the integration is repeated and the module is tested again. After rework, all lower level integration and test cycles should be repeated before repeating the higher-level integration. This is part of the regression testing process and detects any new errors that may slip in due to "fixing" other problems.

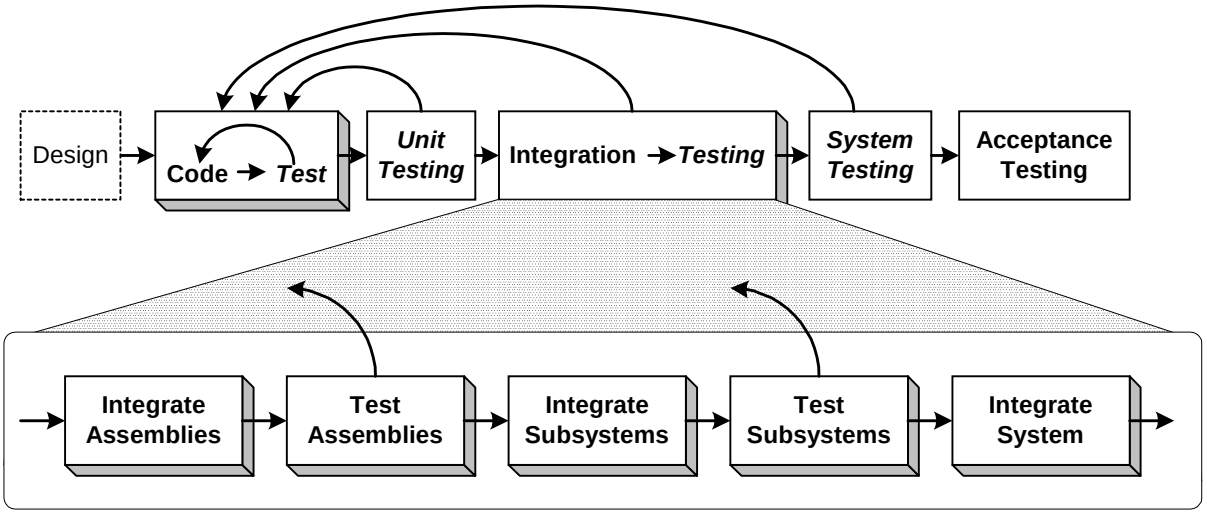

Figure 14-4 System Integration Process

Integration is iterative and progressive, with each level of integration building from and on top of the previous level of integration. This iterative, progressive nature is shown in Figure 14-5. Components are integrated into assemblies, and the assemblies are tested for functionality. Successful testing is followed by the integration of subsystems, which are also tested for correct functionality. Finally, the subsystems are integrated into the complete system, which is then tested for functionality. While three levels of integration are shown in Figures 14-4 and 14-5, it is only representative and in a real project there will probably be additional iterations of integration and testing, depending on the complexity of the system.

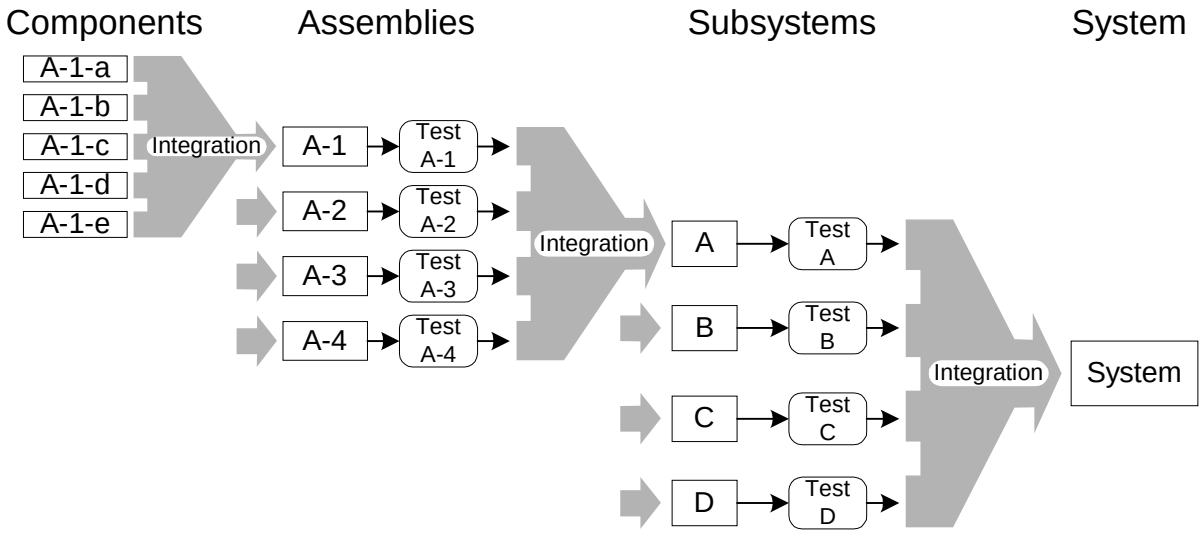

Figure 14-5 Iterative, Progressive Nature of Integration

Integration and testing are part of the development process and are used to ensure all the various pieces work together in performing their higher-level functions.

14.2.1 Interfaces

An absolute essential to any integration effort is complete knowledge of all interfaces. This includes interfaces between components, assemblies, subsystems, and between the system and other systems it will need to work with. This is depicted in Figure 14-6. Defining interfaces and maintaining those definitions is a primary responsibility of systems engineering.

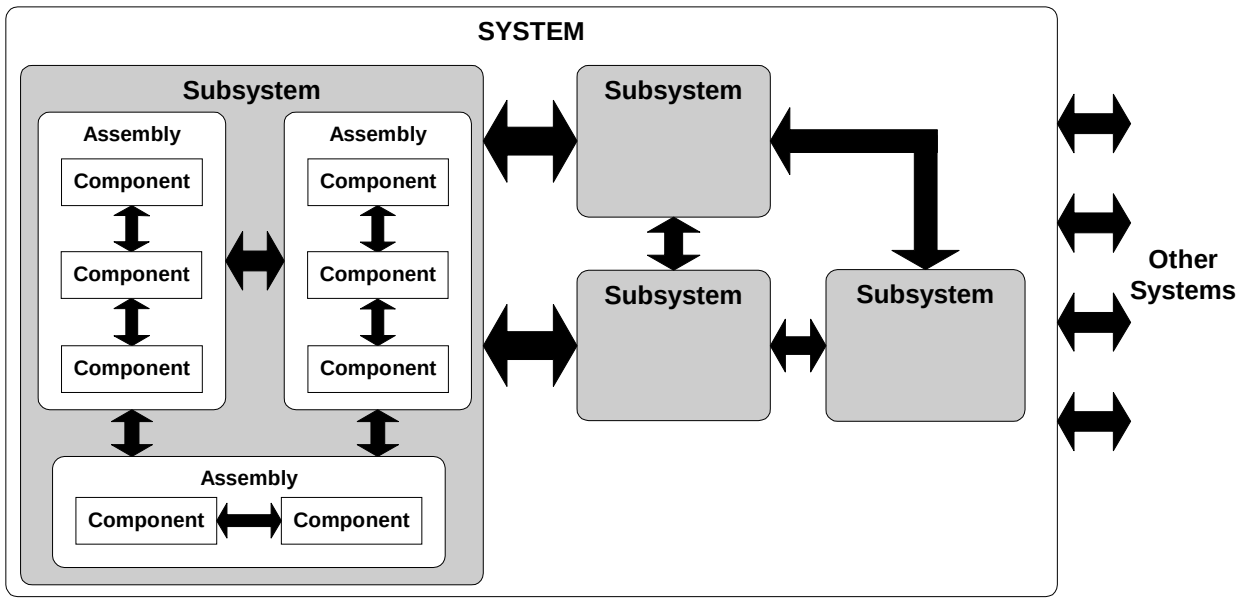

Figure 14-6 Interfaces Between System Parts and Between Systems

14.2.2 Complete System Integration

Most systems consist of both hardware and software. These two are sometimes looked at as complete systems in and of themselves, but they cannot function independently of each other. While they may be called the hardware and software systems, in the system level view they should both be considered as elements of the real, complete system. Development of these two elements may proceed concurrently, with their integration also proceeding concurrently. However, it may be necessary for the hardware to already be in place and operational before the software can be developed, integrated, and tested. Ideally, both elements will be ready for integration into the final system at the same time. If one has to wait on the other, there will likely be problems with schedule, funding, and manpower. The integration of software and hardware elements into a complete system is shown in Figure 14-7.

Figure 14-7 also shows two other system elements: people and support systems. While these other elements may not need to be in place during the development integration, they nonetheless are part of the complete system. For a system to be successfully implemented and used, these other elements must be in place and functioning correctly. The system integration plan must also consider these oft forgotten parts and monitor their establishment. They cannot be left as a follow-on effort. Failure to consider and prepare for all system elements from the beginning will leave the new system crippled or useless.

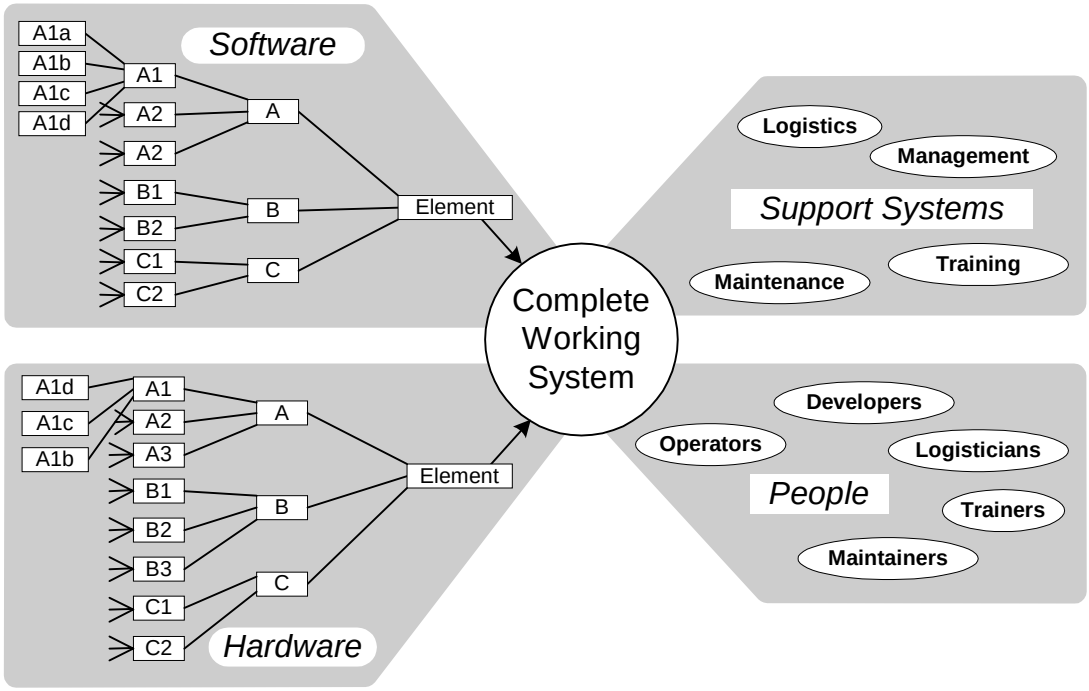

Figure 14-7 System Elements

14.3 System Integration Checklist

This checklist is provided to assist you in understanding the system integration issues of your project. If you cannot answer a question affirmatively, you should carefully examine the situation and take appropriate action.

14.3.1 Before Starting

- ❑ 1. Have you implemented systems engineering as an integrated life cycle effort (see Chapter 13)?
- ❑ 2. Do your test plans include and support integration efforts?
- ❑ 3. Does your development plan allocate adequate time and resources for system integration efforts, including rework time?
- ❑ 4. Are the interfaces between components, assemblies, subsystems, and systems defined in adequate detail?
- ❑ 5. Will hardware be available for testing software during integration?
- ❑ 6. Is there a contingency plan if the schedule slips if and the integration schedule is compressed?
- ❑ 7. Are all elements of the system included in the integration plan?
- ❑ 8. Is all documentation current and available for reference?

14.3.2 During Integration

- ❑ 9. Is there an efficient rework cycle in place to fix problems found during integration testing?
- ❑ 10. Are “fixed” modules or components integrated and retested at all levels of integration up to the level where the problem was found?
- ❑ 11. Is the people element (operators, maintainers, logisticians, trainers, etc.) being prepared to work with the system when it is deployed?
- ❑ 12. Is the support systems element (logistics, maintenance, training, etc.) being prepared to support the new system when it is deployed?

- ❑ 13. Are you following an iterative, progressive integration process?
- ❑ 14. Are experienced integrators involved with the integration?
- ❑ 15. Are area/subject matter experts involved with the integration?
- ❑ 16. Is adequate time being allowed for integration, testing, rework, reintegration, and retesting?
- ❑ 17. Are all necessary resources being made available for integration?
- ❑ 18. Is adequate testing being performed on integrated units (assemblies, subsystems, elements, system) to ensure that there are no surprises during acceptance testing?
- ❑ 19. Are you updating documentation during rework?
- ❑ 20. Are integration and system test errors being traced back to requirements and design? And if so, are the requirements and design being updated?

14.4 References

[1] Maurer, Dr. Karl, "Quotes From Greek Class": www.angelfire.com/ga/dracodraconis/greekquotes.html

14.5 Resources

Department of Energy (DOE) *Software Engineering Methodology*, Chapter 8: http://cio.doe.gov/sqse/sem_toc.htm

Crosstalk Magazine: www.stsc.hill.af.mil/crosstalk/

- "C++ Component Integration Obstacles": www.stsc.hill.af.mil/crosstalk/1997/05/swanson.asp

Guide to Software Engineering Body of Knowledge, especially Appendix D: www.swebok.org

NASA Systems Engineering Handbook: http://ldcm.gsfc.nasa.gov/library/library.htm

Software Engineering Institute: www.sei.cmu.edu

System Engineering Fundamentals, 2001, Defense Acquisition University, download at: www.dau.mil/pubs/gdbks/sys_eng_fund.asp

Systems Engineering Guide, Version 1.1, 5 April 1996, ASC/EN -- SMC/SD: http://web1.deskbook.osd.mil/reflib/DAF/073GZ/001/073GZ001DOC.HTM